Joaquin Garcia-Alfaro
Guillermo Navarro-Arribas
Alessandro Aldini
Fabio Martinelli
Neeraj Suri (Eds.)

LNCS 9481

Data Privacy Management, and Security Assurance

10th International Workshop, DPM 2015
and 4th International Workshop, QASA 2015
Vienna, Austria, September 21–22, 2015
Revised Selected Papers

Springer

Joaquin Garcia-Alfaro · Guillermo Navarro-Arribas
Alessandro Aldini · Fabio Martinelli
Neeraj Suri (Eds.)

Data Privacy Management, and Security Assurance

10th International Workshop, DPM 2015
and 4th International Workshop, QASA 2015
Vienna, Austria, September 21–22, 2015
Revised Selected Papers

Springer

Editors
Joaquin Garcia-Alfaro
Telecom SudParis
Evry
France

Guillermo Navarro-Arribas
Universitat Autònoma de Barcelona
Bellaterra
Spain

Alessandro Aldini
University of Urbino
Urbino
Italy

Fabio Martinelli
National Research Council - C.N.R.
Pisa
Italy

Neeraj Suri
Department of Computer Science
TU Darmstadt
Darmstadt
Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-29882-5 ISBN 978-3-319-29883-2 (eBook)
DOI 10.1007/978-3-319-29883-2

Library of Congress Control Number: 2016930820

LNCS Sublibrary: SL4 – Security and Cryptology

Foreword from the DPM 2015 Program Chairs

This volume contains the proceedings of the 10th Data Privacy Management International Workshop (DPM 2015), held in Vienna, Austria, during September 21–22, 2015, in conjunction with the 20th annual European Symposium on Research in Computer Security (ESORICS 2015). The DPM series started in 2005 when the first workshop took place in Tokyo (Japan). Since then, the event has been held every year in different venues: Atlanta, USA (2006), Istanbul, Turkey (2007), Saint Malo, France (2009), Athens, Greece (2010), Leuven, Belgium (2011), Pisa, Italy (2012), Egham, UK (2013), and Wroclaw, Poland (2014).

The aim of DPM is to promote and stimulate international collaboration and research exchange in areas related to the management of privacy-sensitive information. This is a very critical and important issue for organizations and end-users. It poses several challenging problems, such as translation of high-level business goals into system-level privacy policies, administration of sensitive identifiers, data integration and privacy engineering, among others.

In this workshop edition, 39 submissions were received and each of them was evaluated on the basis of significance, novelty, and technical quality. The Program Committee, comprising 40 members, performed an excellent task and with the help of an additional 22 referees all submissions went through a careful anonymous review process (three or more reviews per submission). In the end, eight full papers, accompanied by six short papers and two position papers were presented at the event. The program was completed with two keynote talks given by Pierangela Samarati (Università degli Studi di Milano) and Dieter Gollman (Technischen Universität Hamburg).

We would like to thank everyone who helped organize the event, including all the members of the Organizing Committee of both ESORICS and DPM 2015. In particular, we would like to highlight and acknowledge all the efforts of the team from SBA Research, for all their help and support. Our gratitude also goes to Pierangela Samarati, steering committee chair of the ESORICS Symposium, for all her arrangements that made possible the satellite events, and Javier Lopez, the workshops chair of ESORICS 2015. Our special thanks to the general chairs of DPM 2015, Josep Domingo-Ferrer and Vicenç Torra. Last but, by no means least, we thank all the DPM 2015 Program Committee members, additional reviewers, all the authors who submitted papers, and all the workshop attendees.

Finally, we want to acknowledge the support received from the sponsors of the workshop: Institut Mines-Telecom (Telecom SudParis), CNRS Samovar UMR 5157 (R3S team), UNESCO Chair in Data Privacy, Universitat Autonoma de Barcelona, Internet Interdisciplinary Institute (IN3), Open University of Catalonia (UOC), and projects TIN2011-27076-C03-02 CO-PRIVACY and TIN2014-55243-P from the Spanish MINECO.

January 2016

Joaquin Garcia-Alfaro
Guillermo Navarro-Arribas

10th International Workshop on Data Privacy Management — DPM 2015

Steering Committee

Additional Reviewers

Foreword from the QASA 2015 Chairs

This proceedings volume contains the revised versions of papers presented at QASA2015: 4th International Workshop in Quantitative Aspects in Security Assurance, held during September 21–22, 2015, in Vienna, as an affiliated event of ESORICS 2015 and in cooperation with DPM.

The QASA workshop series responds to the increasing demand for techniques to deal with quantitative aspects of security assurance at several levels of the development life-cycle of systems and services, from requirements elicitation to run-time operation and maintenance. The aim of QASA is to bring together researchers and practitioners interested in these research topics with a particular emphasis on the techniques for service-oriented architectures. The scope of the workshop is intended to be broad, including aspects as dependability, privacy, risk, and trust.

QASA2015 received 11 submissions, each one reviewed by at least three Program Committee members. The committee decided to accept four papers (after two rounds of evaluations) for the proceedings. The program also includes one invited talk, given by Pierangela Samarati on data protection (in cooperation with DPM).

The presentations and the discussions during the workshop have shown that the area of quantitative security, in its many facets, is an active and interesting field of research.

We would like to thank the invited speakers, the authors of submitted papers, the members of the Program Committee, the external reviewers, and the sponsors, which are the EU projects NeCS and SPECS and the IFIP WG 11.14 (NESSoS) on Secure Engineering. We are also grateful for the use of the EasyChair platform, which offered an effective and clear way of managing the entire review process as well as the proceedings production. Finally, we are also grateful to the SBA-Research and Technology University of Vienna for providing the venue for QASA2015.

January 2016

Alessandro Aldini
Fabio Martinelli
Neeraj Suri

4th International Workshop on Quantitative Aspects in Security Assurance — QASA 2015

General Chairs

Alessandro Aldini	University of Urbino, Italy
Fabio Martinelli	IIT-CNR, Italy
Neeraj Suri	TU Darmstadt, Germany

Program Committee

Andrea Bondavalli	University of Florence, Italy
Tom Clothia	University of Birmingham, UK
Jorge Cuellar	Siemens, Germany
Frédéric Cuppens	Télécom Bretagne, France
Joaquin Garcia-Alfaro	Télécom SudParis, France
Javier Lopez	University of Malaga, Spain
Jesus Luna Garcia	Cloud Security Alliance, UK
Catherine Meadows	Naval Research Laboratory, USA
Charles Morisset	Newcastle University, UK
Pierangela Samarati	University of Milan, Italy
Ketil Stoelen	SINTEF, Norway
Lorenzo Strigini	City London University, UK
Herbert Wiklicky	Imperial College London, UK

Additional Reviewer

Gencer Erdogan	SINTEF, Norway

Contents

Keynote Address

Data Protection in Cloud Scenarios

Sabrina De Capitani di Vimercati(✉), Sara Foresti, and Pierangela Samarati

Computer Science Department, Università degli Studi di Milano, 26013 Crema, Italy
{sabrina.decapitani,sara.foresti,pierangela.samarati}@unimi.it

Abstract. We present a brief overview of the main challenges related to data protection that need to be addressed when data are stored, processed, or managed in the cloud. We also discuss emerging approaches and directions to address such challenges.

Data security and privacy in the cloud

The 'cloud' has emerged as a successful paradigm enabling users and companies to have access to a virtually unlimited amount of resources to store, manage, and process data in a reliable and dependable infrastructure, even with economic advantages with respect to 'in-house' solutions. Together with considerable evident convenience, the cloud also introduces novel security and privacy issues. In fact, when storing or processing data in the cloud, data owners lose control over their data, leaving them potentially exposed to unauthorized parties, including the provider itself that might be not fully trusted. While typically cloud providers may be considered reliable for guaranteeing basic security protection (such as protection from unauthorized accesses to data and resources by third parties), they might not be considered trusted for the confidentiality (i.e., authorized to know the content) - or guaranteeing integrity - of the data they store or process.

Many are the challenges that need to be addressed to guarantee proper security and privacy in the cloud. In this paper, we focus in particular on the challenges specifically related to data management [18,19,21,34,35]. Of course, there are also other security and privacy issues that characterize a cloud scenario (e.g., multi-tenancy and virtualization, fault-tolerance management [26–28]) on which we do not elaborate.

Protection of data at rest. Protection of data at rest concerns the security and confidentiality of data in storage. Data stored at an external cloud provider need to be protected from unauthorized accesses by third parties as well as from the cloud provider itself, which might be not trusted for knowing the content of the data it stores or the accesses performed on them (*honest-but-curios* provider). The protection of the confidentiality of stored data typically relies on encryption. In cloud scenarios, protecting data from the providers' eyes requires keeping the encryption key within the client's trust boundary. In other words, encryption should work at the client side, encrypting data before moving them to the

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 3–10, 2016.
DOI: 10.1007/978-3-319-29883-2_1

cloud. Since encryption makes query evaluation and application execution more expensive or not always possible (see next challenge on 'fine-grained access to outsourced data'), alternative *fragmentation-based* solutions have been also proposed (e.g., [1,6–10]). Fragmentation allows departing from encryption whenever what is sensitive are not the data values singularly taken but their association (e.g., in a medical database, patients' names and illnesses might be considered public, while the specific association between the name of each patient and her illness might be considered sensitive). In this case, instead of encrypting the values, the sensitive association can be protected by storing values that are sensitive in association in different fragments so to break the association itself impeding its visibility to non authorized parties. For instance, with reference to our example, the patients' names can be stored in one fragment and illnesses in another one. To ensure that the sensitive associations protected by fragments cannot be reconstructed, fragments can be stored at independent (and non communicating) providers, or fragments must be guaranteed to be unlinkable. Fragmentation limits encryption to values that are sensitive by themselves, or even completely departs from encryption in cases (e.g., hybrid cloud) where the availability of a trusted party can be assumed for some storage/computation support. The advantage of using fragmentation is the availability of data in the clear, which enables evaluation of conditions on them and therefore better support for query processing by the cloud provider. In addition to data confidentiality, data integrity (i.e., authenticity and integrity of the stored data) and availability (i.e., the satisfaction by cloud providers of the data storage and access requirements that users may wish to enforce) are two further aspects that need to be addressed [3,29].

Fine-grained access to outsourced data. As noted above, cloud providers cannot have full access to the data they store, which might be either encrypted or fragmented. Also, when data are encrypted, the encryption key should remain within the client's trust boundary to ensure data remain confidential even with respect to the storing and processing provider itself. Providers cannot then decrypt data for query execution, making evaluation of conditions and query support difficult (if at all possible). The problem of providing support for fine-grained access (i.e., retrieval of data satisfying given conditions) over encrypted data has been under the attention of the research and development community in recent years and several investigations have been carried out. Among the analyzed techniques there are: cryptographic techniques supporting keyword-based searches (e.g., [4]), homomorphic encryption (e.g., [22]), the use of different layers of encryption each supporting specific operations [33], and metadata (indexes) attached to the data and used for fine-grained information retrieval and execution of specific kinds of queries (e.g., [5,25,39]). The major difficulty in such investigations is the tradeoff existing between providing support for query processing and ensuring that such support does not leak sensitive information otherwise protected by encryption (or fragmentation).

Selective data access. Data stored in the cloud may be subject to different access control policies, meaning that different users might need to enjoy different views on the outsourced data. Enforcing authorizations providing such selective access in cloud environments results particularly challenging since, if on one side clearly the data owner cannot provide such enforcement itself (as it would mean intercepting every access to data), on the other side, the cloud provider may not be fully trusted for such an enforcement. Also, the policy itself might be sensitive or leak information on the data content. There are two main lines of work investigating solutions for enforcing access control policies in the cloud. The first line of work, under the generic umbrella of *attribute-based encryption* (ABE) [24,41] is based on public key encryption and enforces authorizations by ensuring that encryption depends on the values of certain attributes (which characterize authorized users). This way, a user will be able to access data if her set of attributes matches conditions on the attributes associated with the encrypted data. ABE allows enforcement of authorizations that depend on different conditions, thus providing expressive authorization support. The main limitation of such approaches relate to the evaluation cost (given the use of public key encryption) and to the difficulty of enforcing revocation. The second line of work, called *selective encryption* [12,13], is based instead on the use of symmetric encryption and enforces authorizations by translating the authorization policy into an equivalent encryption policy so that data can be encrypted with different keys and keys are distributed to users in such a way that they can decrypt all (completeness) and only (correctness) the data they are authorized to access. A hierarchical organization of the encryption keys employed enables enforcing such authorizations (providing different views over data) while ensuring both a single copy of the data and the use of only one key per user. In fact, proper organization of keys in a hierarchy, with tokens enabling key derivation allows users to derive, from their own key all and only the keys enabling access to data they are authorized to access. Selective encryption provides efficient access control, as only symmetric encryption is used. Also, the use of public tokens enabling key derivation allows their storage in the cloud itself. Changes to the access policies (i.e., grant or revocation of authorizations) can be conveniently enforced by over-encryption, by which the data owner can enforce changes to authorizations with the cooperation of the cloud provider, that - when demanded - can wrap the data with a further level of encryption at the provider side (encrypting resources already encrypted by the owner). Over-encryption enforces authorization changes without the need for the data owner of retrieving, re-encrypting, and re-uploading data already stored in the cloud.

Query privacy. In addition to data themselves, several scenarios may also require confidentiality guarantees on accesses made on data. A classical example of these scenarios is a medical encyclopedia: while the encyclopedia itself is not sensitive and neither might be (with respect to the storing provider) the identity of users accessing it, the specific entries that a user looks for might be considered confidential as they may disclose her (or of a person close to her) health condition. Similar query confidentiality guarantees might also be requested when stored data are encrypted (as knowledge of the access might even compromise the confidentiality of the stored data). The problem then arises of guaranteeing *access confidentiality,* that is, the fact that a given access aims at given data, as well as *pattern confidentiality*, that is, the fact that two accesses aim at the same data. We call these new confidentiality guarantees *query privacy* as the aim is to have them while also supporting efficient access to data for query support (e.g., index-search and evaluation of range conditions). Classical solutions providing access privacy, such as private information retrieval proposals, offer strong guarantees but limited access functionality and bear performance overhead that make them not applicable in real-life scenarios. Among more recent approaches, Path ORAM and the shuffle index, provide better performance and therefore applicability. Both these solutions are based on specific index structures and provide protection by either relying on a local stash, with dynamic re-mapping and delayed writing (Path ORAM) [37] or by relying on caching, cover searches, and shuffling with dynamic re-allocation of data at every access (shuffle index) [16,17]. Open issues are related to the need of decreasing the performance overhead, providing more support for queries, and ensuring strong confidentiality guarantees.

Integrity of query results. In addition to confidentiality, integrity of data can also be put at risk when the involved provider(s) may not be fully trustworthy. While for storage integrity classical techniques (e.g., chaining and signature) can be used, ensuring integrity of data dynamically retrieved, or resulting from computation, is challenging. Assessing integrity for query results or computations entails providing users with the ability to verify that the result returned by the cloud provider is complete (i.e., computed on the whole data collection), correct (i.e., computed on genuine data and correctly performing the computation), and fresh (i.e., computed on the most recent version of the data). Approaches for guaranteeing integrity of query results can be classified as *deterministic* (e.g., [30–32]) and *probabilistic* (e.g., [15,23,36,38,40,42]). Deterministic solutions use authenticated data structures (e.g., signature chains, Merkle hash trees, skip lists) or encryption-based solutions and can detect an integrity violation only for queries formulated on the attribute(s) on which they have been defined. Probabilistic solutions are based on the insertion of fictitious information or checks in a dataset whose absence in a query result signals an integrity violation. Probabilistic approaches can detect an integrity violation for any query but only with probabilistic guarantees, meaning that they are subject to false negative results. The problem of assessing integrity of query results becomes even more complex in emerging scenarios for distributed computation, where different providers or workers may be involved (e.g., [11]).

Collaborative computation with selective sharing. In several scenario computation or query execution in the cloud might involve data under the control of different authorities (data owners) and stored at different providers, which may impose different access and sharing restrictions on their data. Some approaches have addressed the problem of performing collaborative computations in contexts where no sharing is possible between the involved parties and only the query result can be known to them (e.g., secure multi-party computation or *sovereign joins* [2]). These solutions are based on the use of encryption together with the possible involvement of a trusted computing base. Other approaches have considered scenarios where data can be selectively shared with other parties and different data owners and/or cloud providers need to collaborate, and selectively share information with others, for query execution. The problem addressed is then the distributed query execution (which necessarily entails exchange of data among the involved parties) in such a way that data are made visible only to authorized parties and no information is improperly shared or leaked [14,43]. In this context, ongoing work is investigating novel techniques for expressing and enforcing sharing policies, regulating information flows in query execution, and efficiently computing a query execution plan ensuring that no information is improperly released or leaked. Other approaches address the orthogonal problem of protecting the objectives of queries from the providers involved in their evaluation (e.g., [20]).

Acknowledgments. This work was supported in part by: the EC within the 7FP under grant agreement 312797 (ABC4EU) and within the H2020 under grant agreement 644579 (ESCUDO-CLOUD); the Italian Ministry of Research within PRIN project "GenData 2020" (2010RTFWBH).

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, USA (2005)
2. Agrawal, R., Asonov, D., Kantarcioglu, M., Li, Y.: Sovereign joins. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), Atlanta, GA, USA (2006)
3. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., Song, D.: Remote data checking using provable data possession. ACM Trans. Inf. Syst. Secur. **14**(1), 12 (2011)
4. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multikeyword ranked search over encrypted cloud data. In: Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM 2011), Shanghai, China (2011)
5. Ceselli, A., Damiani, E., Capitani, D., di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Trans. Inf. Syst. Secur. **8**(1), 119–152 (2005)
6. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 171–186. Springer, Heidelberg (2007)
7. Ciriani, Valentina, De Capitani di Vimercati, Sabrina, Foresti, Sara, Jajodia, Sushil, Paraboschi, Stefano, Samarati, Pierangela: Enforcing confidentiality constraints on sensitive databases with lightweight trusted clients. In: Gudes, Ehud, Vaidya, Jaideep (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 225–239. Springer, Heidelberg (2009)

8. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: outsourcing data while maintaining confidentiality. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 440–455. Springer, Heidelberg (2009)
9. Ciriani, V., Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. **13**(3), 22 (2010)
10. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE Trans. Dependable Secure Comput. **11**(6), 510–523 (2014)
11. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Integrity for distributed queries. In: Proceedings of the 2nd IEEE Conference on Communications and Network Security (CNS 2014), San Francisco, CA, USA (2014)
12. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: management of access control evolution on outsourced data. In: Proceeding of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (2007)
13. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Trans. Database Syst. **35**(2), 12 (2010)
14. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Authorization enforcement in distributed query evaluation. J. Comput. Secur. **19**(4), 751–794 (2011)
15. Capitani, D., di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Integrity for join queries in the cloud. IEEE Trans. Cloud Comput. **1**(2), 187–200 (2013)
16. Capitani, D., di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Efficient and private access to outsourced data. In: Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapolis, MN, USA (2011)
17. Capitani, D., di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Shuffle index: efficient and private access to outsourced data. ACM Trans. Storage **11**(4), 19 (2015)
18. Capitani, D., di Vimercati, S., Foresti, S., Samarati, P.: Managing and accessing data in the cloud: privacy risks and approaches. In: Proceedings of the 7th International Conference on Risks and Security of Internet and Systems (CRiSIS 2012), Cork, Ireland (2012)
19. di Vimercati, S.D.C., Foresti, S., Samarati, P.: Data security issues in cloud scenarios. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2015. LNCS, vol. 9478, pp. 3–10. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26961-0_1
20. Farnan, N., Lee, A., Chrysanthis, P., Yu, T.: PAQO: preference-aware query optimization for decentralized database systems. In: Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE 2014), Chicago, IL, USA (2014)
21. Foresti, S.: Preserving Privacy in Data Outsourcing. Advances in Information Security, vol. 51. Springer, US (2011)
22. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st ACM Symposium on Theory of Computing (STOC 2009), Bethesda, MD, USA (2009)

23. Ghazizadeh, P., Mukkamala, R., Olariu, S.: Data integrity evaluation in cloud database-as-a-service. In: Proceedings of the 9th IEEE World Congress on Services (SERVICES 2013), Santa Clara, CA, USA (2013)
24. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, USA (2006)
25. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the 21th ACM SIGMOD International Conference on Management of Data (SIGMOD 2002), Madison, WI, USA (2002)
26. Jhawar, R., Piuri, V.: Fault tolerance management in IaaS clouds. In: Proceedings of the IEEE Conference in Europe about Space and Satellite Telecommunications (ESTEL 2012), Rome, Italy (2012)
27. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: Proceedings of the 15th IEEE International Conference on Computational Science and Engineering (CSE 2012), Paphos, Cyprus (2012)
28. Jhawar, R., Piuri, V., Santambrogio, M.: Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst. J. **7**(2), 288–297 (2013)
29. Juels, A., Kaliski, B.: PORs: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007). Alexandria, VA, USA (2007)
30. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Authenticated index structures for aggregation queries. ACM Trans. Inf. Syst. Secur. **13**(4), 32 (2010)
31. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. ACM Trans. Storage (TOS) **2**(2), 107–138 (2006)
32. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying completeness of relational query results in data publishing. In: Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), Baltimore, MD, USA (2005)
33. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processin. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011), Cascais, Portugal (2011)
34. Samarati, P.: Data security and privacy in the cloud. In: Huang, X., Zhou, J. (eds.) ISPEC 2014. LNCS, vol. 8434, pp. 28–41. Springer, Heidelberg (2014)
35. Samarati, P., Capitani, D., di Vimercati, S.: Cloud security: issues and concerns. In: Murugesan, S., Bojanova, I. (eds.) Encyclopedia on Cloud Computing. Wiley, New York (2016)
36. Sheng, G., Wen, T., Guo, Q., Yin, Y.: Verifying correctness of inner product of vectors in cloud computing. In: Proceedings of the 2013 International Workshop on Security in Cloud Computing. Hangzhou, China (2013)
37. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple oblivious RAM protocol. In: Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS 2013), Berlin, Germany (2013)
38. Umadevi, G., Saxena, A.: Correctness verification in outsourced databases: more reliable fake tuples approach. In: Proceedings of the 7th International Conference on Information Systems Security (ICISS 2011), Kolkata, India (2013)

39. Wang, H., Lakshmanan, L.: Efficient secure query evaluation over encrypted XML databases. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea (2006)
40. Wang, H., Yin, J., Perng, C., Yu, P.: Dual encryption for query integrity assurance. In: Proceedings of the 17th Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, CA, USA (2008)
41. Waters, Brent: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, Dario, Fazio, Nelly, Gennaro, Rosario, Nicolosi, Antonio (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
42. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (2007)
43. Zeng, Q., Zhao, M., Liu, P., Yadav, P., Calo, S., Lobo, J.: Enforcement of autonomous authorizations in collaborative distributed query evaluation. IEEE Transactions on Knowledge and Data Engineering (TKDE) **27**(4), 979–992 (2015)

Quantitative Aspects of Security Assurance

Composable Bounds on Information Flow from Distribution Differences

Megumi Ando(✉) and Joshua D. Guttman

The MITRE Corporation, Bedford, USA
{mando,guttman}@mitre.org

Abstract. We define information leakage in terms of a "difference" between the *a priori* distribution over some remote behavior and the *a posteriori* distribution of the remote behavior conditioned on a local observation from a protocol run. Either a maximum or an average may be used. We identify a set of notions of "difference;" we show that they reduce our general leakage notion to various definitions in the literature. We also prove general composability theorems analogous to the data-processing inequality for mutual information, or cascading channels for channel capacities.

Keywords: Information flow · Non-disclosure · Limited disclosure · Information theory · Mutual information · Min-entropy leakage · Composability · Channel capacity

1 Introduction

One of us recently [11] introduced the *Frame Model* for studying information disclosure in distributed systems. Frames represent systems by directed graphs; their arcs represent the channels of communication permitted by the system. Disclosure occurs when the local behavior of one portion of the system (the "source") affects what local behaviors may be observed at another portion of the system. That paper shows that limitations on disclosure respect a *cut principle*: Each limit on the disclosure from a source to a cut set of channels in the graph is also enforced on disclosure from the source to any more distant set of channels. This result furnishes a kid of compositionality for limited disclosure. However, the notion of limited disclosure in [11] was "possibilistic," i.e. non-quantitative.

The purpose of this paper is to take key steps toward adapting its results to a quantitative treatment of disclosure. Given a probability distribution over the local behaviors of the system, we can generalize the cut principle to include a probabilistic analysis for quantifying leakage. As in the motivating example below, a quantitative analysis may capture insecurities that a possibilistic approach may overlook.

To focus our work, however, we have decided to omit one aspect of this problem. Namely, the frame model allows non-determinism. Generally, to obtain

© Springer International Publishing Switzerland 2016
J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 13–29, 2016.
DOI: 10.1007/978-3-319-29883-2_2

probability distributions on runs of a non-deterministic system, one must introduce a "scheduler" that chooses what events happen when different alternatives exist. Specifying these schedulers is subtle, essentially because a scheduler sensitive to the system's secrets can signal them to the observer through its choices. In this paper, we ignore the resolution of non-determinism. Our analysis here applies in any case in which a probability distribution on executions is well-defined. In future work we will define methods for resolving non-determinism without giving the scheduler unfair ways to signal secrets.

A Motivating Example. David Chaum first introduced the Dining Cryptographers' Protocol (DCP) as a means to study secure multi-party boolean-OR computation [5]. Chaum describes a scenario where a group of three cryptographers are at dinner, and either the Spymaster (their boss) or one of the cryptographers at the table pays for the meal. The protocol guarantees that each party can determine whether the Spymaster or one of the cryptographers at the table paid; and in the latter case, the identity of the payer remains hidden from the non-payers.

Let A, B, and C denote the three cryptographers. Without loss of generality, let us assume that A is a non-paying cryptographer, and consider her viewpoint. A flips a coin with B to get r_{AB} and flips another coin with C obtaining r_{AC}. She computes $m_A = r_{AB} \oplus r_{AC}$, and announces m_A to the table. (As the payer, she would have announced $m_A \oplus 1$.) From B and C's announcements, she surmises the overall parity $m = m_A \oplus m_B \oplus m_C$, from which she can determine whether the boss paid.

If m is odd, A learns that one of the other cryptographers paid but cannot know for sure which. Possibilistically, we say that the identity of the payer remains undisclosed to A. Further, *no* information is disclosed since the set of possibilities remains the same before and after a protocol run.

Despite provable non-disclosure, information (quantified in a certain way) can still leak if the coins are biased: Suppose that the payer is chosen from a fixed distribution. Conditioned on either B or C paying, the payer identity is a Bernoulli random variable $X_A \sim \mathsf{Bern}(p)$ with probability p that player B is the payer, and probability $(1-p)$ that player C is the payer. Suppose further that the coin flips are independent and identically distributed Bernoulli random variables: $R_{AB}, R_{BC}, R_{CA} \sim \mathsf{Bern}(q)$.

The set of A's sent and received messages is another (multi-dimensional) random variable: $O_A = R_{AB}, R_{CA}, M_A, M_B, M_C$, where M_A, M_B, and M_C denote the cryptographers' respective m-messages. Rather than merely confirming that the set of possible payers remains the same, we can compare the distributions of X_A pre- and post- protocol run. One way to do this is by computing the difference between the entropies of the a priori and a posteriori distributions:

$$\mathcal{I}(X_A; O_A) = \mathcal{H}(X_A) - \mathcal{H}(X_A|O_A), \tag{1}$$

where $\mathcal{I}(\cdot;\cdot)$, $\mathcal{H}(\cdot)$, and $\mathcal{H}(\cdot|\cdot)$ denote mutual information, entropy, and conditional entropy, resp. (All are formally defined in Sect. 2.1).

In the special case when there is an equal chance of B or C paying ($p = 0.5$), and the coin flips are fair ($q = 0.5$); the unconditional and conditional distributions O_A and $O_A|X_A$ are uniform, and there is no leakage. However, this is not generally true as shown by Chatzikokolakis *et al.* [4]. This example illustrates that information may leak even in scenarios where there is provably no disclosure. In this paper, we show how to generalize the results of [11] to include quantitative analyses such as the type presented above.

Other Related Work. In contrast to the possibilistic approach in [11], there are a number of well-cited papers that use information theoretic definitions for quantifying anonymity, information flow, or non-interference in distributed systems: Díaz *et al.* [9] and Serjantov and Danezis [9] use Shannon entropy; Clarkson *et al.* [7] and Deng *et al.* [8], relative entropy; Köpf and Basin [12], guessing entropy; Chatzikokolakis, Malacaria, Zhu, and others [4,6,13,15], (conditional) mutual information or channel capacity; and Palamidessi, Smith, and others [1–3,10,14], min-entropy leakage. This list is not exhaustive.

Of these information theoretic concepts, we show that our leakage notion is reducible to mutual information and channel capacity [4,6,13,15]. As seen in our motivating example, mutual information is a measure of reductions in uncertainty, where uncertainty is defined as the entropy of a distribution. For a specified a priori distribution, there is no leakage provided that the mutual information between X and O is zero. This idea is generalized by allowing for some intentionally revealed information (represented as a reveal random variable), such that security is achieved with zero *conditional* mutual information or capacity. This is the approach taken in by Chatzikokolakis *et al.* [4] and Clark *et al.* [6] and summarized in Sect. 2.1.

Reduction in uncertainty can also be measured by min-entropy leakage, which is defined as the difference between the min-entropy of the a priori distribution and the conditional min-entropy of the a posteriori distribution. Currently, there is no consensus on how conditional min-entropy should be defined. Indeed, Cachin [3] defines the conditional min-entropy $\mathcal{H}_\infty(X|Y)$ of $X|Y$ as

$$\mathcal{H}_\infty(X|Y) = -\sum_{y\in\mathcal{Y}} \mathsf{P}(Y = y) \cdot \log \max_{x\in\mathcal{X}} \mathsf{P}(X = x|Y = y). \tag{2}$$

whereas Palamidessi, Smith, and others [1,2,10,14] define it with the logarithm and summation reversed. In Sect. 2.1, we provide a summary of min-entropy leakage as defined in [3], and we show how min-entropy leakage derived from this former conditional min-entropy also relates to our notion of leakage in Sect. 4.

Our Contributions. As in [11], we describe whether and (how much) information can leak from one portion of a distributed system to another. We also identify scenarios where the leakage provides an upperbound on information flow to more remote portions of the network. In these cases, compositions of local leakages bounds are meaningful globally.

In addition to providing a generalization of the cut-blur principle in [11], the contributions of this paper are the following:

- We define information flow in a distributed system very generally: Leakage is defined as the max (for worst-case) or average (for average-case) “difference” between the a priori distribution over some remote behavior and the a posteriori distribution of the remote behavior conditioned on a possible local observation from a protocol run.
- We identify a set of distribution differences that relate this unified notion of leakage to accepted definitions in the literature: namely mutual information, min-entropy leakage, and limited or non-disclosure.
- We also prove equivalence and implication relations between different leakage definitions. For zero leakage, we prove that zero mutual information provides the strongest security and implies zero leakage under all distribution differences satisfying the coincidence axiom.
- We identify a sufficient property (convexity) of distribution differences for the composability of leakage bounds analogous to one of the bounds in the data-processing inequality for mutual information, or cascading channels for channel capacities: Given a Markov chain $X \rightarrow Y \rightarrow Z$, the leakage from X to Z is bounded by the leakage from X to Y. If the leakage under the distribution distance is additionally symmetric, then we get the other bound: The leakage from X to Z is also bounded by the leakage from Y to Z. The composability property can also be seen as a generalization of the cut-blur principle for limited disclosure.

Road Map of Paper. Leakage definitions using mutual information, min-entropy leakage, and limited disclosure are described in Sect. 2. In Sect. 2.2, we provide informal descriptions of limited disclosure and the cut-blur principle (the main composability result of [11]). In Sect. 2.3, we formally define distribution differences, which are used in our leakage definitions in Sect. 4. Sections 3–5 contain our problem statement, leakage definitions, and results. We conclude with extensions to our results in Sect. 6.

2 Preliminaries

2.1 Mutual Information, Capacity, and Min-Leakage

Chatzikokolakis *et al.* [4] use conditional channel capacity for quantifing information leakage in anonymity protocols given intentionally revealed information. Below is their leakage definition, preceded by some information theoretic definitions.

Definition 1. *Let $X : \mathcal{X} \rightarrow [0,1]$, $Y : \mathcal{Y} \rightarrow [0,1]$, and $Z : \mathcal{Z} \rightarrow [0,1]$ be discrete random variables.*

1. *The entropy of X, denoted $\mathcal{H}(X)$, is given by*

$$\mathcal{H}(X) = -\sum_{x \in \mathcal{X}} \mathsf{P}(X = x) \cdot \log \mathsf{P}(X = x),$$

where $\mathsf{P}(\cdot)$ denotes probability.

2. *The conditional entropy of $X|Y$, denoted $\mathcal{H}(X|Y)$, is given by*

$$\mathcal{H}(X|Y) = \sum_{y \in \mathcal{Y}} \mathsf{P}(Y = y) \cdot \mathcal{H}(X|Y = y)$$

3. *The mutual information between X and Y, denoted $\mathcal{I}(X;Y)$, is given by*

$$\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y)$$

4. *The mutual information $\mathcal{I}(X;Y|Z)$ between X and Y, conditioned on Z, is given by*

$$\mathcal{I}(X;Y|Z) = \mathcal{H}(X|Z) - \mathcal{H}(X|Y,Z) = \sum_{z \in \mathcal{Z}} \mathsf{P}(Z = z) \cdot \mathcal{I}(X;Y|Z = z)$$

In [4], an anonymity protocol is modeled by a conditional distribution $p_{O|X}(\cdot|\cdot)$ over the space $\mathcal{O} \times \mathcal{X}$, where $\mathcal{X}$ and $\mathcal{O}$ are the domains of a secret random variable $X : \mathcal{X} \rightarrow [0,1]$ and an observable random variable $O : \mathcal{O} \rightarrow [0,1]$, resp. Every run of the protocol produces an independent observable sampled from this conditional distribution. Anonymity is achieved with zero capacity.

If the protocol intentionally reveals some information R, represented as a random variable, then the protocol is secure if it achieves *relative anonymity*, defined below.

Definition 2 *(Informal). Given an anonymity protocol $p_{O|X}(\cdot|\cdot)$, we say it achieves relative anonymity if*

$$\max_{p_X(\cdot)} \mathcal{I}(X;O|R) = 0, \tag{3}$$

where the maximization is over all input distributions $p_X(\cdot)$ on $\mathcal{X}$. In other words, $I(X;O|R) = 0$ for all possible $p_X(\cdot)$.

See [4] for a formal treatement. Min-entropy leakage in [3] is defined analogously using min-entropy and conditional min-entropy:

Definition 3. *Let $X : \mathcal{X} \rightarrow [0,1]$, $Y : \mathcal{Y} \rightarrow [0,1]$, and $Z : \mathcal{Z} \rightarrow [0,1]$ be discrete random variables.*

1. *The min-entropy of X, denoted $\mathcal{H}_\infty(X)$, is given by*

$$\mathcal{H}_\infty(X) = -\log \max_{x \in \mathcal{X}} \mathsf{P}(X = x) \tag{4}$$

2. *The conditional min-entropy of $X|Y$, denoted $\mathcal{H}_\infty(X|Y)$, is given by*

$$\mathcal{H}_\infty(X|Y) = \sum_{y \in \mathcal{Y}} \mathsf{P}(Y = y) \cdot \mathcal{H}_\infty(X|Y = y) \tag{5}$$

3. *The min-entropy leakage from X to Y, denoted $\mathcal{M}(X;Y)$, is given by*
$$\mathcal{M}(X;Y) = \mathcal{H}_\infty(X) - \mathcal{H}_\infty(X|Y), \tag{6}$$
where $\mathcal{H}_\infty(X|Y)$ is as defined in Definition 3.[1]
4. *The min-entropy leakage $\mathcal{M}(X;Y|Z)$ between X and Y, conditioned on Z, is given by*
$$\mathcal{M}(X;Y|Z) = \mathcal{H}_\infty(X|Z) - \mathcal{H}_\infty(X|Y,Z) \tag{7}$$

2.2 Limited Disclosure in the Frame Model

The Frame Model was introduced in [11] as a means for studying composable information disclosure. Communication is modeled as point-to-point, and messages are delivered synchronously. Partial ordering on the message deliveries models true concurrency within a protocol run.

A frame $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$ consists of a set of locations $\mathcal{LO}$, a set of channels $\mathcal{CH}$, a set of data $\mathcal{D}$, and methods $\mathsf{ends}(\cdot)$ and $\mathsf{traces}(\cdot)$ defined on $\ell \in \mathcal{LO}$. Graphically, a frame $\mathcal{F}$ can be represented as a directed graph, where the nodes are the locations, and the (directed) edges are the channels. Each edge is labeled. The label represents the data that can be transmitted along that edge from the exit node to the entry node.

A channel endpoint is either an entry or an exit point of a channel; so $\mathsf{ends}(\ell)$ returns the set of all endpoints that either enter into or exit from ℓ. A trace is an ordered sequence of local *events* that represents a location's interactions with the other locations; where calling the $\mathsf{chan}(\cdot)$ method on an event object returns a channel, and calling the $\mathsf{data}(\cdot)$ method returns a data. So $\mathsf{traces}(\ell)$ returns all possible local sequences representing all the ways in which ℓ might participate in a (potentially incomplete) run of the protocol.

The authors of [11] provide a mathematical notion of an *execution* (a run of a protocol) within the Frame Model and define the portion of an execution relevant to a set C of channels as a C-run.

Definition 4 *(Informal). A function* $\mathsf{blur} : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ *is a* blur operator *if it satisfies the properties:*

1. Inclusion: $\mathcal{T} \subseteq \mathsf{blur}(\mathcal{T})$
2. Idempotence: $\mathsf{blur}(\mathsf{blur}(\mathcal{T})) = \mathsf{blur}(\mathcal{T})$
3. Union property: $\forall \Sigma \subseteq \mathcal{P}(\mathcal{S})\,.\,\mathsf{blur}\left(\bigcup_{\mathcal{T}\in\Sigma} \mathcal{T}\right) = \bigcup_{\mathcal{T}\in\Sigma} \mathsf{blur}(\mathcal{T})$, *where $\mathcal{P}(\cdot)$ denotes the powerset.*

Given a frame $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$, a set $\mathsf{src} \subseteq \mathcal{CH}$ of source channels, and a set $\mathsf{obs} \subseteq \mathcal{CH}$ of observable channels; let $\mathcal{S}$ be the set of source-runs (i.e., $\mathcal{S} = \mathsf{src\text{-}runs}$), and let $\mathcal{O}$ be the set of observable-runs (i.e., $\mathcal{O} = \mathsf{obs\text{-}runs}$).

Information disclosure is restricted by a blur operator $\mathsf{blur}(\cdot)$ if, for every observable $o \in \mathcal{O}$, the set $\mathcal{T} \subseteq \mathcal{S}$ of completed source-runs compatible with the observable o is blur-blurred, i.e., $\mathcal{T} = \mathsf{blur}(\mathcal{T})$, where $\mathsf{blur}(\cdot)$ is a blur operator.

[1] There is an alterative definition for conditional min-entropy [1,2,10,14]. We will not be dealing with this alternative definition here.

The main result of the paper is the so-called cut-blur principle below. See [11] for a formal treatment.

Theorem 1 (Cut-blur Principle, Informal). *Given a frame*

$$\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces}), \tag{8}$$

a set $\mathsf{src} \subseteq \mathcal{CH}$ *of source channels, a set* $\mathsf{cut} \subseteq \mathcal{CH}$ *of cut channels, and a set* $\mathsf{obs} \subseteq \mathcal{CH}$ *of observable channels, such that* cut *is a cut-set partitioning* src *from* obs*, the source information disclosed at* cut *limits the source information disclosed at* obs*.*

2.3 Distribution Differences

In our motivating example, we compared the a priori and a posteriori distributions in order to quantify how much information leaked. Intuitively, if the distributions are the same under some specified way of measuring, there is no measurable leakage. Stated as such, leakage is expressed in terms of "distribution differences," which we define formally below.

Definitions 6–8 are distribution differences, which we use later on to relate our unified leakage notion in Sect. 4 to accepted leakage definitions in the literature, namely: conditional mutual information, min-entropy leakage, and limited information disclosure from [3,4,11]. These leakage definitions are also summarized in Sects. 2.1–2.2.

Definition 5. *Let* $\mathbb{X}_{\mathcal{X}}$ *denote a family of random variables defined over the same alphabet* $\mathcal{X}$*. A distribution difference* $\triangle : \mathbb{X}_{\mathcal{X}} \times \mathbb{X}_{\mathcal{X}} \to \mathbb{R}$ *is a function that takes two random variables defined over the same alphabet and returns a real number. In discussing this definition, we are often interested in the following properties:*

1. Coincidence axiom: $\forall \mathcal{X}, \forall X \in \mathbb{X}_{\mathcal{X}} \,.\, \triangle(X, X) = 0$
2. Nonnegativity: $\forall \mathcal{X}, \forall X_1, X_2 \in \mathbb{X}_{\mathcal{X}} \,.\, \triangle(X_1, X_2) \geq 0$
3. Convexity: $\forall \mathcal{X}, \forall X, X_1, X_2 \in \mathbb{X}_{\mathcal{X}}, \forall \alpha \in [0,1] \,.$

$$\triangle(X, (\alpha X_1 + (1-\alpha)X_2)) \leq \alpha \cdot \triangle(X, X_1) + (1-\alpha) \cdot \triangle(X, X_2) \tag{9}$$

Definition 6. *For any random variables* X_1 *and* X_2 *over the same alphabet* $\mathcal{X}$*, we say that the Shannon-difference between* X_1 *and* X_2*, denoted* $\triangle_S(X_1, X_2)$*, is given by*

$$\triangle_S(X_1, X_2) = \mathcal{H}(X_1) - \mathcal{H}(X_2). \tag{10}$$

Definition 7. *For any random variables* X_1 *and* X_2 *over the same alphabet* $\mathcal{X}$*, we say that the minH-difference between* X_1 *and* X_2*, denoted* $\triangle_{\min}(X_1, X_2)$*, is given by*

$$\triangle_{\min}(X_1, X_2) = \mathcal{H}_\infty(X_1) - \mathcal{H}_\infty(X_2). \tag{11}$$

Definition 8. *For any random variables X_1 and X_2 over the same alphabet $\mathcal{X}$, we say that the maxH-difference between X_1 and X_2, denoted $\triangle_{\max}(X_1, X_2)$, is given by*

$$\triangle_{\max}(X_1, X_2) = \begin{cases} 0 & \mathsf{supp}(X_1) = \mathsf{supp}(X_2) \\ \infty & \textit{otherwise,} \end{cases} \tag{12}$$

where $\mathsf{supp}(X)$ denotes the support of a random variable X.

Note that Shannon-, minH-, and maxH-differences all satisfy the coincidence axiom and convexity. MaxH-difference additionally satisfies nonnegativity.

3 Problem Statement

While [11] presents purely set theoretic ideas, we generalize the cut-blur results to include information theoretic analyses. To do this, we shift from a possibilistic view of local behaviors to a probabilistic perspective. To begin with, we consider local behaviors from only completed executions, where a completed execution is the partially ordered entire set of messages from a completed protocol run. Our results in Sects. 4 and 5 cover leakages from complete observations. This allows us to present our work using cleaner notation. Extension to leakages from partial observations is covered in Sect. 6.

Below, we borrow the formalism from the Frame Model [11] to make our problem statement explicit. Our problem statement is defined with respect to a frame $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$, a fixed set $\mathsf{src} \subseteq \mathcal{CH}$ of source channels, and a fixed set $\mathsf{obs} \subseteq \mathcal{CH}$ of observable channels.

Definition 9. *Given a frame $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$ and a location $\ell \in \mathcal{LO}$, let $\mathsf{chans}(\ell)$ be the set of channels adjacent to ℓ:*

$$\mathsf{chans}(\ell) = \{c \in \mathcal{CH} : \mathsf{entry(c)} \in \mathsf{ends}(\ell) \vee \mathsf{exit(c)} \in \mathsf{ends}(\ell)\}, \tag{13}$$

where $\mathsf{entry}(\cdot)$ and $\mathsf{exit}(\cdot)$ return the entry and exit points of a channel, resp.

Definition 10. *Given a frame $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$ and a location $\ell \in \mathcal{LO}$, let $T(\ell) \subseteq \mathsf{traces}(\ell)$ be defined by*

$$T(\ell) = \{tr \in \mathsf{traces}(\ell) : \exists tr' \in \mathsf{traces}(\ell),\ tr\ is\ a\ proper\ prefix\ of\ tr'\} \tag{14}$$

($T(\ell)$ is the set of traces of ℓ that are proper prefixes of other traces of ℓ.) Let $\mathsf{traces}^*(\ell) = \mathsf{traces}(\ell) \setminus T(\ell)$, and call it the completed traces of ℓ.

Definition 11. *An event set $\mathcal{E} = (E, \preceq)$ is a well-founded, partially ordered set E of events and is generally denoted by the name of the set and in curly-font.*

Definition 12. *Given an event set $\mathcal{E} = (E, \preceq)$ and a set C of channels, the restriction $\mathcal{E} \restriction C$ of $\mathcal{E}$ to C is the event set $(E_c, \preceq_c)$, where:*

1. $E_c = \{e \in E : \mathsf{chan}(e) \in C\}$*, and*
2. $\preceq_c = \preceq \cap E_c \times E_c$*.*

Definition 13. *An event set* $\mathcal{E} = (E, \preceq)$ *is a completed execution in a frame* $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$*, if for all* $\ell \in \mathcal{LO}$*:*

1. $(\mathcal{E} \restriction \mathsf{chans}(\ell))$ *is totally (linearly) ordered, and*
2. $(\mathcal{E} \restriction \mathsf{chans}(\ell)) \in \mathsf{traces}^*(\ell)$*.*

We call the set of all completed executions in a frame $\mathcal{F}$ the completed execution set, denoted $\mathsf{Exe}^*(\mathcal{F})$.

Definition 14. *Given a frame* $\mathcal{F} = (\mathcal{LO}, \mathcal{CH}, \mathcal{D}, \mathsf{ends}, \mathsf{traces})$ *and a set of channels* $C \subseteq \mathcal{CH}$*, let* C-runs^* *be the set of restrictions of completed executions to* C*:*

$$C\text{-}\mathsf{runs}^* = \{\mathcal{E} \restriction C : \mathcal{E} \in \mathsf{Exe}^*(\mathcal{F})\} \tag{15}$$

Let $\mathcal{S}$ be the finite set of completed source-runs (i.e., $\mathcal{S} = \mathsf{src\text{-}runs}^*$), and let $\mathcal{O}$ be the finite set of completed observable-runs (i.e., $\mathcal{O} = \mathsf{obs\text{-}runs}^*$).

We are also provided a partitioning function $f(\cdot,\cdot)$ which is a deterministic function over the joint space $\mathcal{S} \times \mathcal{O}$ and a joint probability mass function (pmf) $p_{S,O}(\cdot,\cdot)$ over $\mathcal{S} \times \mathcal{O}$, such that the supports for the corresponding marginal probabilities are $\mathcal{S}$ and $\mathcal{O}$, resp. In other words, $p_{S,O}(\cdot,\cdot)$ written as a matrix has no all zero rows or columns. The partitioning function additionally has the property that for any $s_1, s_2 \in \mathcal{S}$ and $o \in \mathcal{O}$ such that $p_{S,O}(s_1, o), p_{S,O}(s_2, o) > 0$; $f(s_1, o) = f(s_2, 0)$.

Definition 15. $S : \mathcal{S} \longrightarrow [0,1]$ *is the random variable on the completed source-runs, which maps to the marginal probabilities of the source-runs.*

$$S(s) = \sum_{o \in \mathcal{O}} p_{S,O}(s, o). \tag{16}$$

S *represents the a priori remote behavior.*

Definition 16. $(S|O = o) : \mathcal{S} \longrightarrow [0,1]$ *is the conditional random variable on the completed source-runs, which maps to the probabilities of the source-runs conditioned on the observable-run* $O = o$*.*

$$(S|O = o)(s) = \frac{p_{S,O}(s, o)}{\sum_{\omega \in \mathcal{S}} p_{S,O}(\omega, o)}. \tag{17}$$

$S|O$ *represents the a posteriori remote behavior.*

Definition 17. *Let* $f : \mathcal{S} \times \mathcal{O} \longrightarrow \mathcal{R}$ *be any deterministic function on the joint space.* $R_f : \mathcal{R} \longrightarrow [0,1]$ *is the random variable over the range of* $f(\cdot,\cdot)$ *whose probabilities are given by*

$$R_f(r) = \sum_{(s,o) \in \mathcal{R}_r} p_{S,O}(s, o), \tag{18}$$

where $\mathcal{R}_r = \{(s, o) \in \mathcal{S} \times \mathcal{O} : f(s, o) = r\}$*. Any intentionally revealed information may be represented by some* R_f*.*

Given this set-up, we are interested in defining information flow in $\mathcal{F}$ from source channels to observable channels. In this sense, this work is meant to generalize the main results of [11].

4 Leakage from Distribution Differences

Given a partitioning function $f(\cdot,\cdot)$ and a pmf $p_{S,O}(\cdot,\cdot)$ over the joint space $\mathcal{S}\times\mathcal{O}$; let S, O, and R_f representing remote behavior, local behavior, and intentionally revealed information be as defined in Sect. 3 above. Further, let $\mathcal{V}$ denote the support of R_f, and for any $r \in \mathcal{V}$, let

$$\mathcal{O}_r = \{o \in \mathsf{supp}(O) : \exists s \in \mathcal{S}, f(s,o) = r\}. \tag{19}$$

The definitions, theorems, and corollaries in Sect. 4 are with respect to this set-up. In Definitions 18 and 19, leakage is defined very generally as the max or average difference between the a priori and a posteriori distributions. These are definitions of leakage conditioned on some reveal random variable R_f. Note that unconditional leakage is captured by any all-to-one function $f(\cdot,\cdot)$.

Definition 18 (Worst-case Leakage). *The worst-case leakage $\mathcal{L}_{S;O|R_f}$ conditioned on R_f, is given by the maximum difference between the a priori distribution on $(S|R_f = r)$ and the a posteriori distribution $(S|R_f = r, O = o)$ over $r \in \mathcal{V}$ and $o \in \mathcal{O}_r$:*

$$\mathcal{L}_{S;O|R_f} = \max_{r\in\mathcal{V}} \max_{o\in\mathcal{O}_r} \triangle\left((S|R_f = r), (S|R_f = r, O = o)\right), \tag{20}$$

for some notion of distribution difference $\triangle$. This is the worst-case leakage over all partitions. We say that there is zero conditional worst-case leakage when $\mathcal{L}_{S;O|R_f} = 0$.

Definition 19 (Average-case Leakage). *The average-case leakage $L_{S;O|R_f}$ conditioned on R_f is given by average difference between the a priori distribution on $(S|R_f = r)$ and the a posteriori distribution $(S|R_f = r, O = o)$ over $r \in \mathcal{V}$ and $o \in \mathcal{O}_r$:*

$$\begin{aligned} L_{S;O|R_f} = \sum_{r\in\mathcal{V}} &\mathsf{P}(R_f = r)\cdot \\ &\sum_{o\in\mathcal{O}_r} \mathsf{P}(O = o|R_f = r)\cdot\triangle\left((S|R_f = r), (S|R_f = r, O = o)\right), \end{aligned} \tag{21}$$

for some notion of distribution difference $\triangle$. This is the average-case leakage over all partitions. We say that there is zero conditional average-case leakage when $L_{S;O|R_f} = 0$.

We chose to study nonstandard distribution differences instead of standard distribution distances, such as the Kullback-Leibler divergence (relative entropy) or statistical-closeness, because our general leakage definitions above reduce to accepted leakage notions in the literature under these distribution difference. Lemmas 1 and 2 and in Theorem 2 illustrate these equivalences.

Below, we show that average leakage under Shannon-difference is equivalent to mutual information between remote and locally observable behaviors. Likewise, average leakage under minH-difference is equivalent to min-entropy leakage.

Lemma 1. *Average-case conditional leakage under Shannon-difference is equivalent to conditional mutual information between S and O.*

Proof. Average leakage under Shannon-difference can be converted to conditional mutual information by pulling $\mathcal{H}(S|R_f = r)$ out from the summation and from the definitions of conditional entropy, mutual information, and conditional mutual information. $\square$

Lemma 2. *Average-case conditional leakage under minH-difference is equivalent to conditional min-entropy leakage from S to O.*

Proof. Same proof as above. $\square$

Zero leakage occurs when the a priori and a posteriori situations are equivalent under some specified distribution difference. Whereas unconditional zero leakage corresponds to no leakage in an absolute sense, conditional zero leakage corresponds to a somewhat weaker notion: Other than some intentionally revealed information, there is no leakage.

In Theorem 2 below, we prove that zero conditional worst-case leakage under maxH-difference is equivalent to limited disclosure, where the blur operator is related to the partitioning function. (Note that the equivalence is up to completed runs. See Sect. 6 for the extended results over partial observations.)

Theorem 2 (Non-disclosure over Blur-sets). *Zero conditional leakage under maxH-difference is equivalent to information disclosure restricted by a blur-operator* $\mathsf{blur}_{f,o} : \mathcal{P}(\mathcal{S}) \longrightarrow \mathcal{P}(\mathcal{S})$, *given by*

$$\mathsf{blur}_{f,o}(\mathcal{T}) = \bigcup_{t \in \mathcal{T}} \{s : f(s,o) = f(t,o) \ \wedge \ (S|R_f = f(t,o))(s) > 0\}. \tag{22}$$

Proof. Clearly, $\mathsf{blur}_{f,o}(\cdot)$ is inclusive, idempotent, and satisfies the union property. So $\mathsf{blur}_{f,o}$ is a blur-operator.

Let $\mathcal{T}_o = \mathsf{supp}(S|O = o)$. For any fixed $o \in \mathcal{O}$, $\forall s \in \mathcal{S}$ where $p_{S,O}(s,o) > 0$, $f(s,o)$ maps to the same value, which we denote by r_o; thus, $\mathcal{T}_o = \mathsf{supp}(S|R_f = r_o, O = o)$.

($\Longrightarrow$) For any $\mathcal{T} \subseteq \mathcal{T}_o$, $\mathsf{blur}_{f,o}(\mathcal{T}) = \mathsf{supp}(S|R_f = r_o)$ by construction of the blur-operator; and $\mathsf{supp}(S|R_f = r_o) = \mathsf{supp}(S|R_f = r_o, O = o)$, by equality in the maxH-difference. So, $\mathsf{blur}_{f,o}(\mathcal{T}) = \mathcal{T}_o$ as desired.

($\Longleftarrow$) Given any fixed $r \in \mathcal{V}$ where $R_f(r) > 0$ and any $o \in \mathcal{O}_r$, $\mathcal{T}_o = \mathsf{supp}(S|R_f = r)$ by definition of blur-limited disclosure. So, $\mathsf{supp}(S|R_f = r) = \mathsf{supp}(S|R_f = r_o, O = o)$. In the case where $R_f(r) = 0$, this is vacuously true. □

It can be shown that average zero leakage under Shannon-difference is equivalent to worst-case zero leakage under Shannon-difference.

Theorem 3. *Zero conditional worst-case leakage under Shannon-difference is equivalent to zero conditional mutual information between S and O:*

$$\max_{r\in\mathcal{V},\, o\in\mathcal{O}_r} [\mathcal{H}(S|R_f = r) - \mathcal{H}(S|R_f = r, O = o)] = 0 \iff \mathcal{I}(S;O|R_f) = 0 \quad (23)$$

So from Lemma 1, both definitions are equivalent to zero mutual information between the remote and local behaviors.

Below, we prove that zero leakage under Shannon-difference is the strongest form of zero leakage, which trumps leakages under all other distribution differences that satisfy the coincidence axiom. Thus, while min-entropy captures a stronger notion of randomness compared with Shannon entropy and is often toted as the "correct" entropic notion for security analyses, zero mutual information capture a stronger notion of security than zero min-entropy leakage.

Corollary 1. *Zero conditional leakage under Shannon-difference implies: (i) zero conditional worst-case leakage $\mathcal{L}_{S;O|R_f}$ under $\triangle$, and (ii) zero conditional average-case leakage $L_{S;O|R_f}$ under $\triangle$, for any distribution difference $\triangle$ satisfying the coincidence axiom.*

Proof. Zero conditional leakage under Shannon-difference is equivalent to zero conditional mutual information between S and O (Lemma 1 and Theorem 3). For all $r \in \mathcal{V}$ and for all $o \in \mathcal{O}_r$, the distributions $(S|R_f = r)$ and $(S|R_f = r, O = o)$ are the same; and

$$\triangle((S|R_f = r), (S|R_f = r, O = o)) = 0, \quad (24)$$

by the coincidence axiom. □

In Theorem 2, we proved that zero worst-case leakage under maxH-difference is equivalent to limited disclosure restricted by a blur operator. Theorem 4 below states that zero worst-case leakage is equivalent to zero average-case leakage for non-negative distribution differences. Since maxH-difference is non-negative, Theorem 4 establishes the equivalence of worst-case and average-case zero leakages under maxH-difference.

Theorem 4. *Zero conditional average-case leakage is equivalent to zero conditional worst-case leakage under any reasonable, non-negative distribution difference $\triangle$.*

It can also be shown that worst-case leakage under minH-difference implies average-case leakage under minH-difference which, from Lemma 2, is equivalent to min-entropy leakage.

Theorem 5. *Zero conditional worst-case leakage under minH-difference implies zero conditional min-entropy leakage from S to O. (Note that the reverse implication does not hold, however.)*

Proof (of Theorem 5). From Lemma 2, it suffices to prove that zero worst-case conditional leakage implies zero conditional average-case leakage.

For a fixed $r \in \mathcal{V}$, let p' denote the largest probability mass in the a priori distribution, so that $\mathcal{H}_\infty(S|R_f = r) = -\log p'$. By the hypothesis, the difference between the a priori $\mathcal{H}_\infty(S|R_f = r)$ and a posteriori $\mathcal{H}_\infty(S|R_f = r, O = o)$, for any $o \in \mathcal{O}_r$, is bounded by zero

$$\mathcal{H}_\infty(S|R_f = r) - \mathcal{H}_\infty(S|R_f = r, O = o) \leq 0 \tag{25}$$

Thus, the largest probability mass in each of the a posteriori distributions is at most p'. Suppose that there exists an a posteriori distribution for which the largest probability mass is strictly less than p'. Then, in order for the largest probability mass in the marginals to be p', there must exist another a posteriori distribution for which the largest probability mass is strictly greater than p' to compensate. This contradicts our earlier claim, and so the largest probability mass of every a posteriori distribution must be exactly p'. This is true over all r's and o's. □

5 Composing Leakage Bounds

In order to generalize the cut-blur principle and more generally for composing leakage bounds, we desire a result similar to the data-processing inequality for mutual information, or cascading channels for channel capacities: Given a Markov chain of random variables $X \to Y \to Z$, we wish to bound the leakage from X to Z by the leakage from X to Y, as well as the leakage from Y to Z.

We prove that a sufficient property for achieving the first bound is the convexity of the distribution difference. If the leakage is also symmetric, we obtain the second bound.

Theorem 6. *Let $X : \mathcal{X} \to [0,1]$, $Y : \mathcal{Y} \to [0,1]$, and $Z : \mathcal{Z} \to [0,1]$ be discrete, finite random variables; such that $X \to Y \to Z$ form a Markov chain in that order. Leakage from X to Z is upper bounded by leakage from X to Y under any convex distribution difference $\triangle$.*

Proof (Worst-case leakage). By definition of worst-case leakage, there exists $y' \in \mathcal{Y}$, such that

$$\triangle(X, (X|Y = y')) = \mathcal{L}_{X;Y} \tag{26}$$

Let $\mathsf{Dist}(\cdot)$ denote distribution. For any $z \in Z$,

$$\begin{aligned}
\mathsf{Dist}(X|Z=z) &= \sum_{y\in\mathcal{Y}} \mathsf{P}(Y=y|Z=z)\cdot \mathsf{Dist}(X|Y=y,Z=z) \\
&= \sum_{y\in\mathcal{Y}} \mathsf{P}(Y=y|Z=z)\cdot \mathsf{Dist}(X|Y=y) \qquad (27)\\
\triangle(X,(X|Z=z)) &\leq \sum_{y\in\mathcal{Y}} \mathsf{P}(Y=y|Z=z)\cdot \triangle(X,(X|Y=y)) \qquad (28)\\
&\leq \triangle(X,(X|Y=y')) = \mathcal{L}_{X;Y}, \qquad (29)
\end{aligned}$$

(27) holds from the conditional independence of X and Z by the definition of a Markov chain; so for all $(x,y,z) \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$

$$\mathsf{P}(X=x|Y=y) = \mathsf{P}(X=x|Y=y,Z=z). \qquad (30)$$

(28) holds from the convexity of $\triangle$. (29) follows from y' defined above. From (29), $\mathcal{L}_{X;Z} \leq \mathcal{L}_{X;Y}$. □

Proof (Average-case leakage).

$$\begin{aligned}
L_{X;Z} &\leq \sum_{z\in\mathcal{Z}} \mathsf{P}(Z=z) \sum_{y\in\mathcal{Y}} \mathsf{P}(Y=y|Z=z)\cdot\triangle(X,(X|Y=y)) \qquad (31)\\
&= \sum_{y\in\mathcal{Y}} \mathsf{P}(Y=y)\cdot\triangle(X,(X|Y=y)) = L_{X;Y} \qquad (32)
\end{aligned}$$

(31) holds from (28). □

Theorem 7. *Let X, Y, and Z be discrete, finite random variables; such that $X \to Y \to Z$ form a Markov chain in that order. Leakage from X to Z is upper bounded by leakage from Y to Z for any* symmetric *leakage defined under any convex distribution difference $\triangle$. (Symmetry is achieved when leakage computed in one direction is always equal to leakage in the opposite direction.)*

Proof. The proof follows the symmetry of the leakage and Theorem 6 above. □

5.1 Generalized Cut-Blur Principle

We extend our original problem statement to the following scenario:

Given a frame $\mathcal{F} = (\mathcal{LO},\mathcal{CH},\mathcal{D},\mathsf{ends},\mathsf{traces})$, let $\mathsf{src},\mathsf{cut},\mathsf{obs} \subseteq \mathcal{CH}$ be any three subsets of $\mathcal{CH}$, such that cut is a cut-set partitioning src from obs. Let $\mathcal{S}$, $\mathcal{C}$, and $\mathcal{O}$ be the sets $\mathsf{src\text{-}runs}^*$, $\mathsf{cut\text{-}runs}^*$, and $\mathsf{obs\text{-}runs}^*$, resp. We are given a partitioning function $f(\cdot,\cdot)$ and a pmf $p_{S,C}(\cdot,\cdot)$ over the joint space $\mathcal{S}\times\mathcal{C}$, and so the a posteriori distributions are now conditioned on $C=c$ for $c\in\mathcal{C}$. We assume that there are no all zero rows or columns in $p_{SC}(\cdot,\cdot)$; and we define the random variables S, $S|C$, and R_f, representing the a priori remote behavior, the a posteriori remote behavior at the cut set, and information intentionally revealed at the cut channels as before.

We can apply Theorem 6 and 7 to obtain a generalization of the cut-blur principle in cases where a cut-set imposes a Markov chain on the source, cut, and observable random variables. The leakage bounds below hold for any pmf $p_{S,C,O}(s, c, o)$ over the joint space $\mathcal{S} \times \mathcal{C} \times \mathcal{O}$. In other words, the leakage bounds hold for any conditional distribution $p_{O|C}(o|c)$.

Corollary 2. *Given a cut-set that imposes a Markov chain on the source, cut, and observable random variables; the leakage of the source behavior at the observable channels is bounded by the leakage of the source behavior at the cut when leakage is defined under a convex distribution difference* $\triangle$*. If the leakage is additionally symmetric, it is also bounded by the leakage of the cut behavior at the observable channels.*

Corollary 3. *Suppose that the secret information* X *is not the remote behavior, but determined by the source behavior; so* $X = g(S)$*.*

In this case, the leakage of X *at the cut is bounded by the leakage of* X *at the source channels when leakage is defined under a convex distribution difference* $\triangle$*. If the leakage is additionally symmetric, it is also bounded by the leakage of the cut behavior at the observable channels.*

Corollary 4. *Given a cut-set that imposes a Markov chain on the source, cut, and observable random variables and a leakage which is symmetric and defined under a convex distribution difference; zero leakage of the source behavior at the cut implies zero leakage of the source behavior at the observable channels, zero leakage of* $X = g(S)$ *at the cut, and zero leakage of* X *at the observable channels* $\triangle$*.*

Since leakage under maxH-difference is symmetric and defined under a convex distribution difference, Theorems 6 and 7 and Corollaries 2–4 apply; in particular, the cut-blur principle is obtained from Cor. 4 under maxH-difference.

Corollary 3 bounds the leakage from X to C. Suppose we wish to compute the leakage from X to C instead, rather than merely obtaining a bound for it. Since we are given the function $g(\cdot)$ which relates S to X, we can compute the a priori and a posteriori distributions of X and $X|C = c$ from $g(\cdot)$ and the pmf $p_{SC}(\cdot, \cdot)$, and leakage is computable as the max or average difference between the a priori and a posteriori distributions.

6 Extensions to Our Results

Only completed runs were considered in Sects. 3–5: Zero leakage under maxH-difference and limited disclosure were proven equivalent only up to completed runs. Likewise, the generalization of the cut-blur principle applies only to completed runs.

Suppose that we are provided a function $h_t : \mathcal{O} \to \mathcal{O}_t$ mapping the completed observations to partial observations at some relativistic time t, so that every completed run maps to the unique partial run at time t from which it can progress to completion. Then, we can define a random variable $O_t = h_t(O)$, and $S \rightarrow C \rightarrow O \rightarrow O_t$ form a Markov chain. Thus the leakage to the partial observations O_t is bounded above by the leakage to the completed observations, assuming that the leakage is symmetric and defined under a convex distribution difference. Moreover, we can compute a tighter bound on the leakage to O_t given $h_t(\cdot)$, in much the same way that we computed the leakage from X to C given $g(\cdot)$ in Sect. 5.1.

Suppose that we were provided the conditional probability $p_{C|S}(\cdot|\cdot)$. Then, leakage can be defined as the maximum (over all possible a priori distributions $p_S(\cdot)$) of the maximum or average (over the r's and o's) difference between the a priori and a posteriori distributions; and the results from the Sects. 4 and 5.1 trivially carry through. Under Shannon-difference, these correspond to channel capacity and cascading channel bounds.

Acknowledgments. We are grateful to Chris Eliopoulos Alicea, Joseph J. Ferraro, Vineet Mehta, Paul D. Rowe, John D. Ramsdell, Joe J. Rushanan, and the reviewers of this paper for helpful comments.

References

1. Alvim, M.S., Andrés, M.E., Chatzikokolakis, K., Degano, P., Palamidessi, C.: Differential privacy: on the trade-off between utility and information leakage. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 39–54. Springer, Heidelberg (2012)
2. Alvim, M.S., Chatzikokolakis, K., Palamidessi, C., Smith, G.: Measuring information leakage using generalized gain functions. In: Proceedings of the 25th Computer Security Foundations Symposium (CSF 2012) (2012)
3. Cachin, C.: Entropy Measures and Unconditional Security in Cryptography. Ph.D. thesis, Swiss Federal Institute of Technology Zürich (1997)
4. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. Inf. Comput. **206**(2–4), 378–401 (2008)
5. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. J. Cryptology **1**, 65–75 (1988)
6. Clark, D., Hunt, S., Malacaria, P.: Quantitative information flow, relations and polymorphic types. J. Logic Comput. **15**(2), 181–199 (2005)
7. Clarkson, M.R., Myers, A.C., Schneider, F.B.: Belief in information flow. In: Proceedings of the 18th Computer Security Foundations, (CSFW-18 2005) (2005)
8. Deng, Y., Pang, J., Wu, P.: Measuring anonymity with relative entropy. In: Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S. (eds.) FAST 2006. LNCS, vol. 4691, pp. 65–79. Springer, Heidelberg (2007)
9. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
10. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
11. Guttman, J.D., Rowe, P.D.: A cut principle for information flow. In: Proceedings of the 28th Computer Security Foundations Symposium (CSF 2015). IEEE, July 2015

12. Köpf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: Proceedings of the 14th Computer and Communications Security (CCS 2007). ACM (2007)
13. Malacaria, P.: Assessing security threats of looping constructs. In: ACM SIGPLAN Notices, vol. 42. ACM (2007)
14. Smith, G.: Quantifying information flow using min-entropy. In: Proceedings of the 8th Quantitative Evaluation of Systems (QEST 2011), pp. 159–167, September 2011
15. Zhu, Y., Bettati, R.: Anonymity vs. information leakage in anonymity systems. In: Proceedings of the 25th Distributed Computing Systems (ICDCS 2005). IEEE (2005)

Quantitative Analysis of Network Security with Abstract Argumentation

Francesco Santini[1] and Artsiom Yautsiukhin[2](✉)

[1] Dipartimento di Matematica e Informatica, Università di Perugia, Perugia, Italy
francesco.santini@dmi.unipg.it

[2] Instituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
artsiom.yautsiukhin@iit.cnr.it

Abstract. An Abstract Argumentation Framework (AAF) represents a useful technique for the analysis of arguments supporting or discouraging decisions (i.e., information can be in conflict). In particular, we apply Abstract Argumentation to support the administration of security in computer networks. Our approach captures the high-level topology of a system and helps to specify which and where security countermeasures are more appropriate. We design a quantitative analysis on AAFs (modelling our domain knowledge) with the purpose to compare different decisions and select the most suitable one to protect the critical assets.

1 Introduction

Arguments distinguish themselves from proofs by the fact that they are defeasible, that is, the validity of their conclusions can be disputed by other arguments. Therefore, whether an argument is accepted depends on the existence of possible counterarguments, which can themselves be attacked by other counterarguments (and so on). Nowadays, a part of research on the topic of Argumentation Theory is based on Abstract Argumentation [13]. The central concept in this work is an Abstract Argumentation Framework (AAF), which is essentially a directed graph $\langle A, R \rangle$ in which arguments A are represented as nodes, and attack relations R are represented by directed edges. Given an AAF, one can find all accepted arguments depending on the selected argumentation semantics, which is defined by a certain degree of scepticism or credulousness.

In this work we focus on the application of Abstract Argumentation for decision-making during the risk management of an IT system. We assume that the administrator of the considered system has already identified the main threats, and would like to check whether the installed security controls are sufficient to maintain the risk level at minimum. Such analysis may be used

F. Santini supported by GNCS-INDAM "Efficient Methods for Argumentation-based Decision Support".

A. Yautsiukhin supported by H2020-MSCA-ITN "European Network on Computer Security" (675320) and the PRIN Security Horizons project.

separately, or as a part of the overall risk assessment process (see, for example, [1,11,14,16,24]). The main goal of any risk assessment is to check whether the security goals are achieved (e.g., to protect critical assets). Moreover, security should not violate the core business goals for operating the system. For example, security controls should not significantly impact its productivity. In addition, some controls may conflict with each other, reducing the overall security level.

Usually, the three steps performed to reason with (Abstract) Argumentation are, *(i)* first to identify the knowledge base to generate a set of arguments and determine in which ways these arguments attack each other, then *(ii)* to determine the sets of arguments that can be accepted, using a pre-defined criterion corresponding to an argumentation semantics, and finally, *(iii)* to identify the set(s) of accepted conclusions. This procedure can help us to determine whether the system has enough controls to deal with every possible threat [19]. On the other hand, this qualitative analysis does not allow checking the effectiveness of the installed measures and, also, comparing different alternatives.

In this paper, we extend the capability of analysis of IT networks with AFFs [19], by introducing a quantitative analysis on top of it. We add weights to AFFs and we consider it as a weighted graph. The aim is to perform a pre-computation on the graph to measure the effects of security controls on possible penetrations. Each node (argument) in the considered AAF is associated with two values denoting a positive and a negative effect respectively. We assign values to controls, whose meaning is the amount of protection provided by each of them, and values to threats, which represent a strength score of attacks, measuring their impact. In our model we also consider conflicting controls, which may reduce the efficiency of each other, e.g., a networks-based IDS and a VPN over the same network.

The paper extends the work in [19] by proposing a quantitative analysis of resilience of the network against separate penetrations (i.e., attacks with a single origin). The paper is structured as follows: in Sec. 2 we provide the necessary background notions related to AAFs (Sec. 2.1) and semiring algebraic structures (Sec. 2.2). In Sec. 3 we describe how to build an AAF by using the network-topology description of a system. The AAF quantitative analysis using c-semirings is introduced in Sec. 4. The related work is analysed in Sec. 5. Finally, Sec. 6 wraps up the paper and proposes the intended future work.

2 Background

2.1 Abstract Argumentation Frameworks

In this section we briefly summarise the background information related to classical Abstract Argumentation Frameworks (AAFs) [13].

Definition 1 (AAF). *An Abstract Argumentation Framework (AAF) is a pair* $\langle A, R \rangle$ *where* A *is a set of arguments, and a binary relation* $R \subseteq A \times A$*.* $\forall a, b \in A$*,* aRb *(or,* $a \rightarrowtail b$*) means that* a *attacks* b*. An AAF may be represented by a*

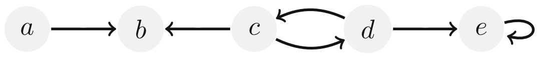


Fig. 1. An example of AAF.

directed graph whose nodes are arguments, and edges represent the attack relation. A set of arguments $S \subseteq A$ attacks an argument a, i.e., $S \rightarrowtail a$, if a is attacked by an argument of S, i.e., $\exists b \in S.b \rightarrowtail a$.

Definition 2 (Defence). *Given $F = \langle A, R \rangle$, an argument $a \in A$ is defended (in F) by a set $S \subseteq A$ if for each $b \in A$, such that $b \rightarrowtail a$, also $S \rightarrowtail b$ holds.*

The "acceptability" of an argument can be defined under different semantics σ, depending on the occurrence of its membership to some sets, called *extensions*: such semantics characterise a collective "acceptability" for arguments. In Definition 3 we only report the original semantics given by Dung [13] (successive proposals can be found in the literature [21, Chap. 2.5]: $\sigma = \{adm, com, prf, stb, gde\}$, which stand for admissible, complete, preferred, stable, and grounded semantics.

Definition 3 (Semantics [13]). *Let $F = \langle A, R \rangle$ be an AAF. A set $S \subseteq A$ is conflict-free (in F), denoted $S \in cf(F)$, iff there are no $a, b \in S$, such that $a \rightarrowtail b$ or $b \rightarrowtail a \in R$. For $S \in cf(F)$, it holds that* (i) *$S \in adm(F)$, if each $a \in S$ is defended by S;* (ii) *$S \in com(F)$, if $S \in adm(F)$ and for each $a \in A$ defended by S, $a \in S$ holds;* (iii) *$S \in prf(F)$, if $S \in adm(F)$ and there is no $T \in adm(F)$ with $S \subset T$;* (iv) *$S \in stb(F)$, if for each $a \in A \backslash S$, $S \rightarrowtail a$;* (v) *$S = gde(F)$ if $S \in com(F)$ and there is no $T \in com(F)$ with $T \subset S$.*

We also recall that the different requirements in Definition 3 define an inclusion hierarchy on the extensions: from the most to the least stringent we have $stb(F) \subseteq prf(F) \subseteq com(F) \subseteq adm(F)$. For such reason, this hierarchy also defines a degree of credulousness (conversely, "strength") for a considered subset of arguments, e.g., the stable semantics is the least credulous (strongest) among all, and this is why it will be extensively used in the rest of the paper. The grounded extension is the minimal fixed point (on complete extensions) of a framework: it minimises the arguments that are taken in.

Moreover, we can also define a strength level for each argument. A sceptically accepted argument proves to be stronger than a credulously accepted one.

Definition 4 (Arguments acceptance-state). *Given one of the semantics σ in Definition 3 and a framework F, an argument a is* (i) *sceptically accepted in iff $\forall S \in \sigma(F), a \in S$,* (ii) *$a$ is credulously accepted if $\exists S \in \sigma(F), a \in S$ and a is not sceptically accepted.*

Example 1. Consider $F = \langle A, R \rangle$ in Fig. 1, with $A = \{a, b, c, d, e\}$ and $R = \{a \rightarrowtail b, c \rightarrowtail b, c \rightarrowtail d, d \rightarrowtail c, d \rightarrowtail e, e \rightarrowtail e\}$. In F we have

$adm(F) = \{\emptyset, \{a\}, \{c\}, \{d\}, \{a,c\}, \{a,\ d\}\}$, $com(F) = \{\{a\}, \{a,c\}, \{a,d\}\}$, $prf(F) = \{\{a,d\},\ \{a,c\}\}$, $stb(F) = \{\{a,d\}\}$, and $gde(F) = \{a\}$. Hence, argument a is sceptically accepted in $com(F)$, $prf(F)$ and $stb(F)$, while it is only credulously accepted in $adm(F)$.

2.2 C-Semirings

In this work we use *c-semiring* [6] with the purpose to generalise the discussion on quantitative aspects of our approach: c-semirings abstract the considered metrics and operators over them. This algebraic structure allows us to specify the main qualities of our approach, up to the constraints of c-semirings.

A c-semiring can be defined as a domain of values D and two different operators defined over D: multiplication ($\otimes$) and addition ($\oplus$). Moreover, we have two special elements in this domain: the worst (bottom) value $\mathbf{0}$, and the best (top) one $\mathbf{1}$ (see [6] for more details). Formally,

Definition 5. *A* c-semiring *S is a tuple $\langle D, \oplus, \otimes, \mathbf{0}, \mathbf{1}\rangle$ where*

- *D is a (possibly infinite) set of elements and $\mathbf{0}, \mathbf{1} \in D$;*
- *$\oplus$ is a binary, commutative, idempotent, and associative operator, with $\mathbf{0}$ as is its* unit *element and $\mathbf{1}$ as is its absorbing element;*
- *$\otimes$ is a binary, commutative, associative, and distributive over $\oplus$, with $\mathbf{1}$ as is its* unit *element and $\mathbf{0}$ as is its* absorbing *element;*

Definition 6. *$\leq_S$ is a partial order relation over D: $s_1 \leq_S s_2$ iff $s_1 \oplus s_2 = s_2$.*

Anytime we are able to prove that some security metric satisfies the requirements above, then this metric can be automatically used in our framework as a particular instance of a c-semiring.

Example 2. Some c-semirings that can help representing security-related problems (applying various security metrics [17]) are the following ones:

Cost of penetration: $S^1 = \langle \mathbb{N} \cup +\infty, \min, +, +\infty, 0\rangle$, also known as tropical semiring. It is commonly used with the aim to minimise costs.
Skill level: $S^2 = \langle \{1,2,3,4,5\}, \min, \max, 5, 1\rangle$. It is derived from the Fuzzy semiring.
Probability of success: $S^3 = \langle [0,1], \max, \times, 0, 1\rangle$, also known as the Viterbi semiring. It is commonly used to maximise the probability of an event.

`Cost of penetration` can be measured with natural numbers and the best cost for the attacker is $\mathbf{1} = 0$, while the worst cost is $\mathbf{0} = +\infty$. Every step requires additional cost (arithmetic sum $+$), when among several alternatives the cheapest one is selected (min operation). Thus, `cost of penetration` can be modelled with the tropical semiring, i.e., S^1. Similarly, the `probability of successful penetration` metric is an example of Viterbi semiring S^3.

3 Modelling Security Systems with Argumentation Frameworks

In this section we show how to draw an AAF that models our problem, with in mind the successive security analysis of complex networks (later in Sec. 4). Our approach is based on the topology of a network in order to: *(i)* precisely indicate the source and target of a threat; *(ii)* explicitly check that there is a possibility for a threat to propagate from its source to its target; *(iii)* indicate the place in the network where security countermeasures are the most effective.

3.1 Running Example

As a running example we consider a hypothetical small research-and-development company. The company collaborates with other enterprises to develop high-tech goods. Thus, such company would like to protect its knowledge from external attackers (i.e., a *PKE* goal), and internal dishonest employees e.g., deliberately spreading sensitive information (i.e., a *PKI* goal).

A company administrator has identified the following threats and security controls to achieve the security goals and protect against the identified threats:

- hacker penetration (HP):
  ∇ host IDS (HI),
  ∇ network IDS (NI);
- employee abuse (EA):
  ∇ VPN (virtual private network) (VPN),
  ∇ encrypted line (EL);

As usual, the administrator would like to know: *Is the current protection enough? What can improve the security system?* In the following of this paper we help the administrator to answer to these questions.

3.2 From Topology to AAF

We use a simplistic model for network topology, which can be seen as a (topology) graph $TG = \langle E, L \rangle$ with elements E of the network (devices, channels, and networks) as nodes, and edges L representing direct communication-links between these elements. Channels can be seen as special sub-networks working over the more general network, and they require a special attention (e.g., protection).

Example 3. The considered company has an IT system (see Fig. 2) containing only two devices: a server SV (where the secret information is stored and processed), and a workstation WS (a terminal of an operator) connected by a local access network LAN. WS is also connected to the Internet for general usage. In addition, we also consider a special communication channel CH between WS and SV, working over LAN. More devices may have access

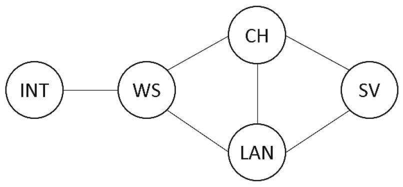


Fig. 2. A model for the topology of our running example.

to LAN, so channel CH provides additional protection to the most important communication sessions. In this work we consider only SV and WS as members of LAN, for the sake of simplicity.

The sets of all the elements in the considered network is $E = \{WS, SV, LAN, CH, INT\}$, and connection links are $L = \{\langle WS, LAN\rangle, \langle SV, LAN\rangle, \langle WS, CH\rangle, \langle SV, CH\rangle, \langle CH, LAN\rangle, \langle WS, INT\rangle\}$. Besides to links between devices and networks, we also add a link between the LAN and CH in order to show that CH is working over LAN. Figure 2 shows the corresponding topology-graph $TG = \langle E, L\rangle$.

We consider only one type of threat when a penetrator is able to manipulate an element, or connect to a network. Every element of network and every direct communication-link can be either in compromised or in protected state and represented as a negative or a positive argument correspondingly.

When an element is directly connected to another one, this means that the couple of arguments for the former element attacks the couple of arguments for the connecting link, which in turn attacks the couple of arguments for the latter element.[1] By an attack between two couples of arguments we mean that the negative argument of the first couple attacks the positive argument of the second couple, and vice versa. For example, Fig. 3 shows how a network and a device are connected together in this way: we show the relative arguments and attacks, as previously explained. An example of network-through connection of one device to another one (and vice versa) is depicted in Fig. 4 instead.

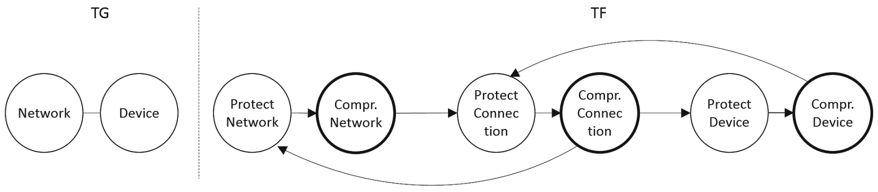


Fig. 3. Attacks between arguments of a generic network-device connection.

[1] We do not distinguish between incoming and outgoing control, but this separation can be easily taken into account by using two couples of connecting arguments (one for incoming and another for outgoing traffic), instead of only one couple.

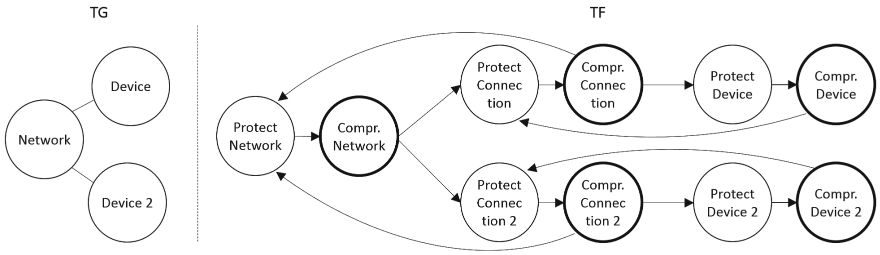


Fig. 4. Attacks between arguments of a generic device-network-device connection.

Thus, now we can transform our topology graph into a graph composed by arguments, i.e., an AAF. We can see each argument as either positive, which means that the considered element (or connection link) is not controlled by a penetrator, or negative, which denotes the fact that an element (or a connection link) is under the control of a penetrator. Thus, initially our AAF only contains the nodes for seizing threat propagation.

Let Q be a set of all threats we would like to consider, plus a seizing control threat $q_{int} \in Q$. We denote a protection argument of threat type q for an element e as $P^q(e)$, while $C^q(e)$ denotes the compromising argument for the same element and the same threat type. The same arguments for the communication links between e_1 and e_2 are $P^q(e_1 - e_2)$ and $C^q(e_1 - e_2)$.

Let e_1 and e_2 be two directly connected elements of a network. By $e_1 \xrightarrow{q} e_2$ we denote a graph $subTF = \langle subA, subR \rangle$, similar to Fig. 3, where $subA = \{P^{q_{int}}(e_1), C^{q_{int}}(e_1), P^q(e_1 - e_2), C^q(e_1 - e_2), P^q(e_2), P^q(e_2)\}$, and where $subR = \{\langle P^{q_{int}}(e_1), C^{q_{int}}(e_1)\rangle, \langle C^{q_{int}}(e_1), P^q(e_1 - e_2)\rangle, \langle P^q(e_1 - e_2), C^q(e_1 - e_2)\rangle,$ $\langle C^q(e_1 - e_2), P^q(e_2)\rangle, \langle P^q(e_2), C^q(e_2)\rangle\}$. We also use a similar notation to show that an element may attack some positive argument $a : e_1 \xrightarrow{q} a$ to express a graph $subTF = \langle subA, subR \rangle$ formed by three arguments, such that: $subA = \{P^q(e_1), C^q(e_1), a\}$, $subR = \{\langle P^q(e_1), C^q(e_1)\rangle, \langle C^q(e_1), a\rangle\}$.

Let $TF = \langle A, R \rangle$ be an preliminary AAF assembled from $TG = \langle E, L \rangle$ as follows: for every communication link $l = \langle e_1, e_2 \rangle \in L$, add $e_1 \xrightarrow{q_{int}} e_2$ and $e_2 \xrightarrow{q_{int}} e_1$ to TF.

3.3 Threats

Since there are threats that do not require seizing control over elements, but have other negative effects (e.g., they lead to information leakage), we use different arguments for such threats, marked by different threat-type indexes (q). Nevertheless, such attacks require access to the target element: a penetrator has to control some element that is connected to the target, and only then the final step can be executed.

Every threat has a single[2] *origin*, i.e., the element through which the threat enters into the network, and a *target*, i.e., the element that, once compromised, leads the threat to success.[3] Thus, we denote a threat as $T(E_1, E_2, Q)$: T identifies a specific threat, E_1 is the origin (i.e., an element from set E), a target element E_2 from the same set, and a threat type $q \in Q$. For example, $HP(INT, SV, q_{int})$ means that a hacker penetration (HP) starts from the Internet and targets seizing control over server SV.

3.4 Security Controls

Let C be the set of all the controls. Every security control can be seen as a new element of a network, linked to some existing element. A control cannot be compromised, and it always attacks a compromising argument for the protected element. We add two additional elements in between, in order to represent a link, as it is done in Fig. 5. Conflicting controls do not attack each other directly, but they affect the protective argument for this link.

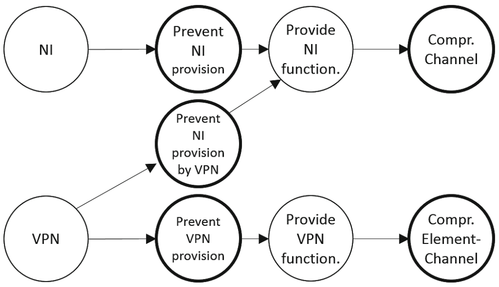


Fig. 5. An example of modelling a conflict between two security controls, *NI* and *VPN*.

Let CE be a set of potentially compromised entities, which can be defined as a tuple: $\{E \cup L \cup \langle C, \{E \cup L\}\rangle\} \times Q$ denotes an element, a link or a link between a control and an element, affected by a threat of a specific type. Then a security control can be formalised as $C(2^{CE}, 2^{CE}, 2^{G})$, where G is the set of goals. Thus, in order to define a control, we specify its name from the set C, a set of the topology elements or communication links that is protected by the control against a specific threat type; a set of conflicts between the considered control and other controls protecting some elements or communication links against a specific threat type; and the business goals affected by the control. For example, a VPN can be seen as $VPN(\{\langle WS, CH\rangle \times q_{int}\}, \{NI \times CH \times q_{int}, MF \times WS -$

[2] In the current modelling we do not consider distributed attacks.

[3] One may argue that a threat also should specify a goal, next to the compromised element, e.g., a hacker may want to attack the PKE goal. In this paper we assume that every threat is binary linked to its goal, and an explicit specification of this goal is then redundant.

$CH \times q_{abuse}\}, \emptyset)$. This means that a VPN protects the connection between WS and channel CH from being used by unauthorised users. Moreover, it prevents a network-based IDS to protect channel CH in full power; however, no business goal is affected.

3.5 Methodology

Let $TG = \langle E, L\rangle$ be a topology graph, T, C and G be the sets of threats, controls and goals respectively. In the following, we use T, C and G as arguments, without explicitly stating this. Goals and countermeasures could be seen as protective arguments, when threats are compromising arguments. Then, we extend the preliminary argument framework TF and build the final $AAF = \langle A, R\rangle$ in the following way:

1. Start with TF built from TG: $AAF = TF$.
2. Add goals to AAF : $A = A \cup G$.
3. Add threats to AAF. For every threat $t(e_1, e_2, q)$ attacking goal $g \in G$:
   (a) Connect the threat itself to the origin of the threat. $A = A \cup \{t\}$, $R = R \cup \{\langle t, P^{q_{int}}(e_1)\rangle\}$;
   (b) If $q = q_{int}$ then
      i. add an attack from the target element to the goal: $R = R \cup \{\langle C^{q_{int}}(e_2), g\rangle\}$;
   (c) else
      i. add a special couple of arguments for the target element e_2 marked with the type of threat e_2^q: add $e_2^q \xrightarrow{q} g$ to AAF;
      ii. for all elements e' connected to e_2: add $e' \xrightarrow{q} e^q$ to AAF.
4. Add security controls to an AAF. For every control $c(CE_p, CE_c, G_c)$:
   (a) Add to AAF an argument for a control: $A = A \cup \{c\}$;
   (b) Connect a control with the protected entities by adding two intermediate arguments for that link. $\forall ce_p = \langle e, q\rangle \in CE_p$ (or $ce_p = \langle l, q\rangle \in CE_p$):
      i. $A = A \cup \{C^q(c-e)\} \cup \{P^q(c-e)\}$ (or $A = A \cup \{C^q(c-l)\} \cup \{P^q(c-l)\}$);
      ii. $R = R \cup \langle c, C^q(c-e)\rangle \cup \langle C^q(c-e), P^q(c-e)\rangle \cup \langle P^q(c-e), C^q(e)\rangle$ (or $R = R \cup \langle c, C^q(c-l)\rangle \cup \langle C^q(c-l), P^q(c-l)\rangle \cup \langle P^q(c-l), C^q(l)\rangle$);
   (c) Add the conflicts between controls as attacks. For all $\forall ce_c = \langle c_2 - e, q\rangle \in CE_c$ (or $ce_p = \langle c_2 - l, q\rangle \in CE_c$):
      i. $A = A \cup \{C^q(c - c_2 - e)\}$ (or $A = A \cup \{C^q(c - c_2 - l)\}$);
      ii. $R = R \cup \langle c, C^q(c - c_2 - e)\rangle \cup \langle C^q(c - c_2 - e), P^q(c_2 - e)\rangle$ (or $R = R \cup \langle c, C^q(c - c_2 - l)\rangle \cup \langle C^q(c - c_2 - l), P^q(c_2 - l)\rangle$);
   (d) Add the conflicts between a control and the goals, as attacks. For all $\forall g_c \in G_c$
      i. $R = R \cup \langle c, g_c\rangle$.

Example 4. First we identify all the goals, threats, and countermeasures. The set of goals has been already defined in Sec. 3.1: $A^g = \{PKE, PKI\}$.

The formal definitions of threats are:

– hacker penetration (to SV): $HP(INT, SV, q_{int})$;
– employee abuse of WS (through the Internet): $EA(WS, INT, q_{abuse})$.

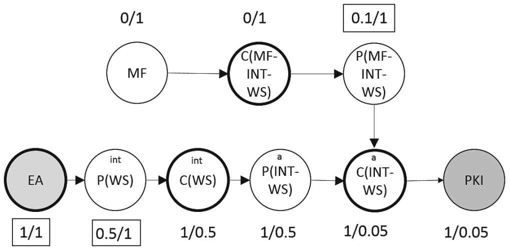


Fig. 6. The AAF for the analysis of the EA threat.

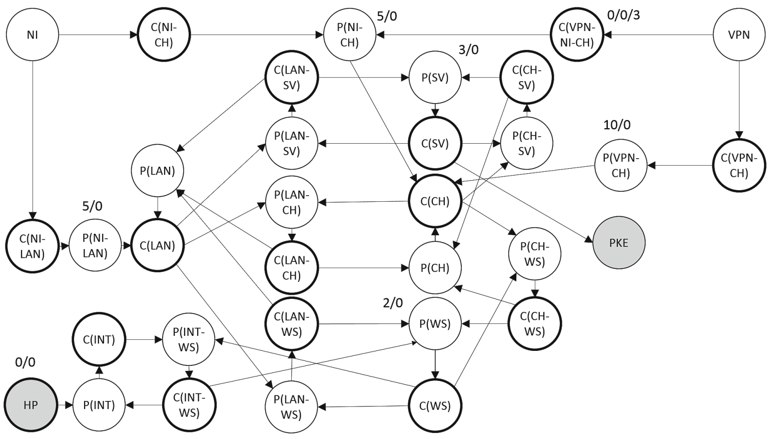


Fig. 7. The AAF for the analysis of HP threat.

Hacker penetration needs the ability for a penetrator to use SV (the server). Thus, an additional node for this threat is not required (see steps 3c-i and 3c-ii): only a link from the target elements to the compromised goal is added (3b-i).

On the other hand, we have to add an additional argument for the abuse threat. First, we add an argument for the threat, and then we connect it to the origin: $A = A \cup \{EA\}$ and $R = R \cup \{\langle EA, P^{q_{int}}(WS)\rangle\}$. Then, we add an attack for compromising the security goal (step 3c-i) for arguments marked with threat "abuse" (q_a) rather than "integrity" (q_{int}: $INT^a \xrightarrow{q_a} PKI$). Finally, we add a link between WS and the newly added arguments-couple: $WS \xrightarrow{q_a} INT^a$. Note that if a dishonest employee knows about the monitoring of its traffic going to the Internet, and there is another node in the network connected to the Global Network, then she (theoretically) may compromise another device in the LAN, and redirect the secrete data through it.

Now we are able to add the security controls installed in the system. These controls may be defined as follows:

– Network-based IDS: $NI(\{LAN \times q_{int}, CH \times q_{int}\}, \emptyset, \emptyset)$.
– Host-based IDS: $HI(\{SV \times q_{int}\}, \emptyset, \emptyset)$.

- Monitoring functionality: $MF(\{MF, \langle WS, INT\rangle \times q_a\}, \emptyset, \emptyset)$.
- VPN: $VPN(\{CH \times q_{int}\}, \{NI \times CH \times q_{int}, MF \times \langle WS, CH\rangle \times q_a\}, \emptyset)$.
- Encrypted line: $EL(\{CH \times q_{int}\}, \{NI \times CH \times q_{int}, MF \times \langle WS, CH\rangle \times q_a\}, \emptyset)$.

Two observations are important here. First, every definition of security control should be defined by experts in terms of rules to be applied. For example, Network-based IDS protects not only the primary network, but also sub-networks (if it has access to them). Second, we see that VPN and EL have negative effects on the operation of a network-based IDS and monitoring tools. Note that the second negative effect is not important in our example, since the monitoring of channel CH is not required.

Figures 6 and 7 show the AAFs for the analysis of potential abuse and hacker penetration. We purged Fig. 6 of the arguments not relevant for our analysis. Moreover, in Fig. 7 we removed the indexes for the attack type, since it is q_{int} everywhere.

4 Quantitative Analysis

By lifting the AAF obtained to represent the security of our system we can consequently perform a quantitative security-analysis. By cost we mean a numerical representation describing any resource the attacker has to consume to propagate further (e.g., cost, time, skill level, probability of success). Every such score is modelled as a c-semiring, and it has to satisfy all the properties in Sec. 2.

For every argument $a \in A$ we consider two values denoting a positive (protective) effect d_a^p and a negative (threatening) effect d_a^n. For the conflicting effect of one control onto another one we use an additional value d_a^c. Initially, we assign $d_a^p = \mathbf{1}$ and $d_a^n = \mathbf{1}$ for each argument. In other words, a penetrator pays no cost trespassing through the network and all the installed countermeasures are not able to stop her.

First, we prepare the graph for the analysis by pre-computing the effects of security controls on the possible penetration. We assign values to controls, whose meaning is the amount of protection provided by countermeasure. Control values are assigned to the positive arguments connecting the countermeasure argument c, and the arguments for the affected element e or link l, i.e., to argument $P(c{-}e)$ (or $P(c-l)$) for the considered threat type. The meaning of these values is the additional obstacle the penetrator has to bypass in order to advance further. The conflicting effect of one control c is assigned to the negative arguments connected to the positive argument of another control c_2 affecting an element e or link l, i.e., to argument $C(c - c_2 - e)$ (or $P(c - c_2 - l)$) (see Fig. 7). The meaning of this conflicting effect is the reduction of the protective power of the affected countermeasure. Finally, we are able to modify the values related to the protective arguments of network elements. The positive effects increase the penetration cost. In the simplest case, no value is provided and, thus, only the countermeasures prevent a penetrator to reach its goal without any cost.

We define a special associative operation $\circledcirc : S \times S \mapsto S$, which reduces the effect of a countermeasure due to the installation of another (conflicting) countermeasure. The concrete form for this operation should be specified for every specific c-semiring separately. We do not impose any additional constraint on the definition of this associative operation, and we require $\circledcirc$ only to return a value from the c-semiring domain. Thus, if one control has a positive value d^p_a and another one conflicts with the first one with a value $d^c_{a'}$, the resulting positive effect of the first control is: $d^p_a = d^p_a \circledcirc d^c_{a'}$.

Example 5. Consider the `cost of penetration` defined in Example 2; $\circledcirc$ can be defined as: $s_1 \circledcirc s_2 = d_1 - d_2$ if $d_1 - d_2 \geq 0$ or 0 otherwise. An example of value assignment to two conflicting countermeasures is shown in Fig. 7. In this example, the final value for $P(NI - CH)$ (denoted as $d^p_{P(NI-CH)}$) is impeded by $C(VPN - NI - CH)$ with value 3 (denoted as $d^c_{C(VPN-NI-CH)}$). Thus, the resulting positive effect value for $P(NI - CH)$ is $d^p_{P(NI-CH)} = d^p_{P(NI-CH)} - d^c_{C(VPN-NI-CH)} = 5 - 3 = 2$.

Similarly, for the `skill level` c-semiring (see Example 2) $\circledcirc$ can be defined as: $s_1 \circledcirc s_2 = d_1 - d_2$ if $d_1 - d_2 \geq 1$ or 1 otherwise.

For `probability of success` c-semiring $\circledcirc$ can be defined as: $s_1 \circledcirc s_2 = d_1 + d_2 - d_1 d_2$, where the meaning of the value of s_2 is the conditional probability to overpass the control due to the additional obstacle for that control.

Now we can pre-compute the control effects. Let C contain all the countermeasures, and for all the arguments we have their values as defined above. In order to get a resulting effect for every control, we need to re-compute the positive effect of each argument $a \in A$ such that $a = P(c - e)$, with $c \in C$. For this purpose we repetitively apply $\circledcirc$ to the initial positive value of an argument and all the values of the arguments attacking it, i.e., $\forall a'_i \in A$ $(\langle a'_i, a\rangle \in R : \quad d^p_a = ((...(d^p_a \circledcirc d^c_{a'_1}) \circledcirc ...) \circledcirc d^c_{a'_n})$. At the end of this step, all the countermeasure effects are ready to be taken into account.

Analysis phase. Consider our $AAF = \langle A, R\rangle$ as updated with the effect of countermeasures and the values assigned to nodes. Let also a_i be the argument for the considered threat.

We slightly modify the Dijkstra algorithm [12] for a shortest-path search [7], simplifying the Floyd-Warshall algorithm based on semirings [8].

```
1. Put $a_i$ in set $a_i \rightarrow A_w$ and mark it.
2. While $A_w$ is not empty:
   (a) extract $a$ from $A_w$ such that $\forall \hat{a} \in A_w (d^n_a \oplus d^n_{\hat{a}} = d^n_a)$.
   (b) for all $a' \in A$ such that $\exists \langle a, a'\rangle \in R$ and $a'$ is not marked:
       i   Aggregate the values $\bar{d^n_{a'}} = d^n_a \bigotimes_{\forall \langle a'', a'\rangle} d^p_{a''}$.
       ii  $d^n_a = \bar{d^n_{a'}}$.
       iii Mark $a'$.
       iv  If $a'$ is the target node - exit.
       v   Add $a'$ to $A_w$.
```

Note that after such pre-analysis phase, the values denoting positive effects do not change. Moreover, because of the properties of $\otimes$ we have that $\bar{d^n_{a'}} \oplus d^n_a = \bar{d^n_{a'}}$ (item i in the algorithm). Thus, we always know the final values of computation at step 2b-i as soon as we reach it. We mark out the already considered arguments, in order to avoid re-processing them once again.

The result corresponds to the final negative values for the target nodes. In Examples 6 and 7 we show the result of our quantitative analysis on the two previously presented examples, i.e., employee abuse (EA) and hacker penetration (HP) respectively.

Example 6. First consider the AAF for the analysis of employee abuse shown in Fig. 6. Here we analyse the `probability of success`, and we use the c-semiring $S^3 = \langle [0,1], \max, \times, 0, 1 \rangle$ from Example 2 and the $\circledcirc$ operation defined in Example 5.

First, we assign the initial values to nodes as specified. In our example, for every argument a we compute the initial value as $d^p_{a'}/d^n_{a'}$; we compute it as $d^p_{a'}/d^n_{a'}/d^c_{a'}$ in presence of two conflicting controls. Since in S^3 we have that $\mathbf{0} = 0$ and $\mathbf{1} = 1$, then we assign $1/1$ to all the arguments. Afterwards we specify that monitoring of the connection of WS to the Internet stops 90 % of attempts. We also assume that the probability that an employee starts behaving dishonestly is 50 %. Initially assigned values and the origin of the threat (EA) are indicated with a frame around the values. No conflicting effects have place.

Thus, the pre-analysis phase does not change any value (and $P^{int}(MF - WS - INT)$ in particular). Then, we can start from the origin of the threat and find the best path to the target argument (the goal PKI).

The computation is simple and its result step by step is (as it is also indicated in Fig. 6):

1. $d^n_{P^{\mathtt{int}}(WS)} = d^n_{EA} \otimes d^p_{EA} = 1 \times 1 = 1$
2. $d^n_{C^{\mathtt{int}}(WS)} = d^n_{P^{\mathtt{int}}(WS)} \otimes d^p_{P^{\mathtt{int}}(WS)} = 1 \times 0.5 = 0.5$
3. $d^n_{P^{\mathtt{a}}(INT-WS)} = d^n_{C^{\mathtt{int}}(WS)} \otimes d^p_{C^{\mathtt{int}}(WS)} = 0.5 \times 1 = 0.5$
4. $d^n_{C^{\mathtt{a}}(INT-WS)} = d^n_{P^{\mathtt{a}}(INT-WS)} \otimes d^p_{P^{\mathtt{a}}(INT-WS)} \otimes d^p_{P(MF-INT-WS)} = 0.5 \times 1 \times 0.1 = 0.05$
5. $d^n_{PKI} = d^n_{C^{\mathtt{a}}(INT-WS)} \otimes d^p_{C^{\mathtt{a}}(INT-WS)} = 0.05 \times 1 = 0.05$

Example 7. Consider the analysis for the hacker penetration threat (HP) (see Fig. 7) with `cost of penetration` c-semirings $S^1 = \langle \mathbb{N} \cup +\infty, min, +, +\infty, 0 \rangle$ from Example 2 and the $\circledcirc$ operation defined in Example 5.

First, we assign $\mathbf{1} = 0$ to positive and negative values for all the nodes. Then, we assign a positive effect of NI: 5 for LAN and 5 for CH; and positive effect of VPN: 10 for CH. Then, we see that VPN has a negative impact to the protection of the channel CH by NI (with value 3). Therefore, we need the pre-analysis computation of the real effect of NI on CH, in presence of VPN: $d^p_{P(NI-CH)} \circledcirc d^c_{C(VPN-NI-CH)} = 5 - 3 = 2$. We also assume that the hacker needs some effort to compromise WS (with value 2) and SV (with value 3).

Now, by executing the algorithm provided in this section, we find that $d^n_{PKE} = d^p_{P(WS)} \otimes d^p_{P(NI-LAN)} \otimes d^p_{P(SV)} = 2 + 5 + 3 = 10$, since the alternative path (through CH) results in a score of $d^n_{PKE} = 17$. Hence, we prefer the path with the lowest (best) cost, i.e., 10.

5 Related Work

Since the application of Argumentation to Cybersecurity-related issues is relatively a new field (or, at least, not deeply investigated), there is a few related work to be mentioned. A bunch of works applying Argumentation-based conflict-resolution to the specific case of firewall rules are [3–5]. In our approach, however, we provide a general reasoning-tool, not focused on firewall rules only, but applicable to network security in general.

In [9] the authors formalise the reasoning about access control using a planning theory formalised in Dung's abstract argumentation framework [13]; such planning is based on an adaptation of Dung's notion of defence. Their formal argumentation framework allows arguments about the backward derivation of plans from objectives and policy rules (abduction), as well as arguments about the forward derivation of goals from general objectives. Parties negotiate to find an agreement about which policy to apply, even though there may be more than one way to achieve a security objective.

A first general and introductory work on Argumentation and Cybersecurity is proposed in [22]. There the authors suggest the use of Argumentation to provide automated support for Cybersecurity decisions. Three different tasks where Argumentation can contribute are surveyed in the paper: first, the establishment of a security policy, drawing from a range of information on best practice and taking into account likely attacks and the vulnerability of the system to those attacks. Secondly, the process diagnosis to determine if an attack is underway after some apparent anomaly in system operation is detected; the final goal is to decide what action, if any, should be taken to ensure system integrity. At last, Argumentation can be used to reconfigure a security policy in the aftermath of a successful attack: this reconfiguration needs to ensure protection against future similar-attacks, without creating new vulnerabilities.

The work in [10] introduces an approach for the enforcement of security requirements based on argumentative logic; the aim is to reason about activation or deactivation of different security mechanisms under certain functional and non-functional requirements. The framework is applied to an automotive on-board system. Differently from this work, in [10] the authors take advantage of Argument-based Logic Programming (see [21, Chap. 8]), and not Abstract Argumentation (see Sec. 2.1).

In [18], some of the authors of this paper propose how arguments can support the decision making process: the aim is to help the system security administrator to react (or not) to possible ongoing attacks. For instance, a decision can be taken either to disable traffic through port 80 or not to disable it. The work in [18] represents a first step along the line presented here; however, it does not consider the topology of a network.

Finally, the overall idea of considering the behaviour of a penetrator as a sequence of steps aggregated in a graph is similar to attack graphs [15,20,23,25]. One of the differences w.r.t. our work is that an attack graph considers the possible exploitation of existing vulnerabilities as steps for a penetrator, while we assume that the existence of a link between elements is already an opportunity for a penetrator. Thus, our approach analyses networks without scanning, implicitly taking into account the existence possibility of zero-day, non-technical, and non-standard vulnerabilities: in general, vulnerabilities that cannot be detected during network scanning. Furthermore, our approach can model the potential conflicts between controls and their effects on business goals.

6 Conclusion

In this paper we have proposed a high-level quantitative analysis of network systems by modelling knowledge with Abstract Argumentation Frameworks [13]. The main advantage of our approach is that the model construction is pretty simple, since it only requires the topology description and the information about threats, countermeasures, and main goals. Indeed, the AAF graph itself is complex to read, but one can easily provide a supporting tool to automatise the transformation and hide details. This is indeed one feasible direction for the future work.

Specification of threats, countermeasures, and goals is context-specific and it should be performed by a person who knows the system very well. On the other hand, a knowledge-base can be defined by experts, with the purpose to help the security analyst. For example, a network-based IDS protects a network (not a device) from specific types of attacks. Conflicts between countermeasures can also be pre-defined. Furthermore, even values for the analysis, which are always difficult to specify, can be preassigned by an expert. Thus, it is possible to limit the work of an analyst to selecting the most suitable choice, by using a knowledge-base and additional predicates over the elements of a network.

In this work, we equipped Abstract Argumentation with a quantitative analysis on the directed graph represented by the AAF itself. This does not invalidate the original semantics-based analysis (see Sec. 2.1), but we flank other preference-based approaches, as e.g., [2]. Hence, now it is not only possible to find the different extensions satisfying one of the semantics in Sec. 2.1, but we can compare these alternatives with our graph-based quantitative approach. In the paper we have not considered such combined analysis, but we would like to close this gap in the future, by designing a more comprehensive framework.

Finally, we plan to apply our approach in a real scenario and test its scalability and usefulness.

References

1. Alberts, C., Dorofee, A., Stevens, J., Woody, C.: Introduction to the octave approach. Technical report, Software Engineering Institute, Carnegie Mellon University (2003)
2. Amgoud, L., Cayrol, C.: On the acceptability of arguments in preference-based argumentation. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI 1998, pp. 1–7. Morgan Kaufmann Publishers Inc. (1998)
3. Applebaum, A., Levitt, K.N., Rowe, J., Parsons, S.: Arguing about firewall policy. In: Verheij, B., Szeider, S., Woltran, S. (eds.) COMMA, Frontiers in Artificial Intelligence and Applications, vol. 245, pp. 91–102. IOS Press (2012)
4. Bandara, A.K., Kakas, A.C., Lupu, E.C., Russo, A.: Using argumentation logic for firewall policy specification and analysis. In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T. (eds.) DSOM 2006. LNCS, vol. 4269, pp. 185–196. Springer, Heidelberg (2006)
5. Bandara, A.K., Kakas, A.C., Lupu, E.C., Russo, A.: Using argumentation logic for firewall configuration management. In: Integrated Network Management, pp. 180–187. IEEE (2009)
6. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44**, 201–236 (1997)
7. Bistarelli, S., Montanari, U., Rossi, F., Santini, F.: Unicast and multicast QoS routing with soft-constraint logic programming. ACM Trans. Comput. Logic **12**(1), 5 (2010)
8. Bistarelli, S., Santini, F.: Two trust networks in one: using bipolar structures to fuse trust and distrust. In: Twelfth Annual International Conference on Privacy, Security and Trust, pp. 383–390. IEEE (2014)
9. Boella, G., Hulstijn, J., van der Torre, L.W.N.: Argumentation for access control. In: AI*IA, pp. 86–97 (2005)
10. Bouyahia, T., Idrees, M.S., Cuppens-Boulahia, N., Cuppens, F., Autrel, F.: Metric for security activities assisted by argumentative logic. In: Garcia-Alfaro, J., Herrera-Joancomartí, J., Lupu, E., Posegga, J., Aldini, A., Martinelli, F., Suri, N. (eds.) DPM/SETOP/QASA 2014. LNCS, vol. 8872, pp. 183–197. Springer, Heidelberg (2015)
11. Butler, S.A.: Security attribute evaluation method: a cost-benefit approach. In: Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), pp. 232–240. ACM Press (2002)
12. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematlk **1**, 269–271 (1959)
13. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–357 (1995)
14. ITC. MAGERIT Version 1.0 Risk analysis and management methodology for information systems Procedures Handbook. Information Technology Council (2000)
15. Jha, S., Sheyner, O., Wing, J.: Two formal analysis of attack graphs. In: Proceedings of the IEEE Computer Society Security Foundations Workshop, p. 49. IEEE Computer Society, Washington, DC (2002)
16. Karabacak, B., Sogukpinar, I.: Isram: information security risk analysis method. Comput. Secur. **24**(2), 147–159 (2005)

17. Krautsevich, L., Martinelli, F., Yautsiukhin, A.: Formal approach to security metrics. what does "more secure" mean for you? In: Proceedings of the 1st International Workshop on Measurability of Security in Software Architectures. ACM Press (2010)
18. Martinelli, F., Santini, F.: Debating cybersecurity or securing a debate? In: Cuppens, F., Garcia-Alfaro, J., Zincir Heywood, N., Fong, P.W.L. (eds.) FPS 2014. LNCS, vol. 8930, pp. 239–246. Springer, Heidelberg (2015)
19. Martinelli, F., Santini, F., Yautsiukhin, A.: Visualising network security through arguments. In: Thirteenth Annual International Conference on Privacy, Security and Trust (PST). IEEE (2015)
20. Ortalo, R., Deswarte, Y., Kaaniche, M.: Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Trans. Softw. Eng. **25**(5), 633–650 (1999)
21. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence, 1st edn. Springer, US (2009)
22. Rowe, J., Levitt, K., Parsons, S., Sklar, E., Applebaum, A., Jalal, S.: Argumentation logic to assist in security administration. In: Proceedings of the Workshop on New Security Paradigms, pp. 43–52. ACM (2012)
23. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 273. IEEE Computer Society (2002)
24. Stoneburner, G., Goguen, A., Feringa, A.: Risk management guide for information technology systems. Technical Report 800–30, National Institute of Standards and Technology (2001)
25. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Comput. Commun. **29**(15), 2917–2933 (2006)

Security-Based Adaptation of Multi-cloud Applications

Kyriakos Kritikos[1] and Philippe Massonet[2](✉)

[1] ICS-FORTH, Heraklion, Greece
kritikos@ics.forth.gr
[2] CETIC, Gosselies, Belgium
philippe.massonet@cetic.be

Abstract. Multi-cloud application management can optimize the provisioning of cloud-based applications by exploiting whole variety of services offered by cloud providers and avoiding vendor lock-in. To enable such management, model-driven approaches promise to partially automate the provisioning process. However, such approaches tend to neglect security aspects and focus only on low-level infrastructure details or quality of service aspects. As such, our previous work proposed a security meta-model, bridging the gap between high- and low-level security requirements and capabilities, able to express security models exploited by a planning algorithm to derive an optimal application deployment plan by considering both types of security requirements. This work goes one step further by focusing on runtime adaptation of multi-cloud applications based on security aspects. It advocates using adaptation rules, expressed in the event-condition-action form, which drive application adaptation behaviour and enable assuring a more-or-less stable security level. Firing such rules relies on deploying security metrics and adaptation code in the cloud to continuously monitor rule event conditions and fire adaptation actions for applications when the need arises.

1 Introduction

Security is considered as an important factor influencing the decision of business organisations to move to the cloud. Cloud computing constitutes a major trend towards provisioning applications and business processes in a flexible and suitable manner. Cloud computing promises access to infinite resources on demand. However, especially in public clouds, there are new risks which must be appropriately addressed to reduce the probability of security breaches.

Security risks are increased when an application is deployed in a hybrid or multi-cloud. These deployment modes are of interest to the research and business community, as they offer advantages over single-cloud deployments: (a) no vendor lock-in; (b) exploitation of great variety of cloud services such that the best possible service level can be achieved. As such, we can now see some initial prototypes and public platform extensions towards supporting the hybrid or multi-cloud provisioning of applications.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 47–64, 2016.
DOI: 10.1007/978-3-319-29883-2_4

One main direction for achieving multi-cloud application provisioning comes via a model-driven approach promising to partially automate the cloud application lifecycle activities. Many research projects are working on the cloud deployment problem. PaaSage (http://www.paasage.eu/) proposes a new domain specific language (DSL), CAMEL [1], to describe the whole application lifecycle, thus facilitating automation of the management of cloud deployments. Its architecture comprises three main modules that cooperatively enable complete support of design- and run-time adaptation loops for cloud deployments.

Unfortunately security is often overlooked in cloud deployment languages which tend to neglect this critical aspect that must be addressed to increase trust in cloud computing usage. To remedy this, our previous research focused on specifying security aspects and considering them in optimising multi-cloud application deployment [2]. In particular, CAMEL was extended with a security meta-model [3] to capture high- and low-level security requirements and capabilities as well as appropriately linking them. The latter meta-model leads to extending a deployment planner towards filtering the cloud provider space based on user-defined security requirements.

To cover both runtime and design-time adaptation, this paper extends our work by proposing using: (a) adaptation rules that are triggered by application security level thresholds and (b) security metrics in the optimisation function of deployment planning problems such that security level breaches are avoided in the long run. In essence, the second bridges the gap when adaptation rules cannot deal with security level issues at run-time and considers security in the design time loop as an equal factor with respect to performance or cost to find application deployments with better security levels. A critical success factor for the second direction is the appropriate design of optimisation functions which suitably consider the trade-offs between security, performance and cost. As such, this paper provides guidelines to application providers on how to construct appropriate optimisation functions capturing such trade-offs.

The rest of the paper is structured as follows. Section 2 provides background information on the PaaSage multi-cloud application lifecycle and our previous work on optimising secure multi-cloud deployments. Section 3 constitutes the paper core by presenting our security solution's extended PaaSage architecture and the way adaptation rules and optimisation functions are modelled. Section 4 applies our solution to a use case thus highlighting its main benefits. Section 5 analyses related work. Finally, the last section summarizes the paper and draws directions for further research.

2 Background

2.1 PaaSage's Adaptive Model-Based Application Provisioning Workflow

The PaaSage model-driven deployment approach relies on an extensive DSL capturing many aspects in multi-cloud application life-cycle, including the specification of requirement, deployment, scalability, organisation and provider models. This approach relies on an architecture, shown in Fig. 1, comprising three modules: Upperware, MetaDataDataBase (MDDB) and Executionware. Upperware is able to map user requirements and application profiles to concrete deployment plans to be executed by the Executionware. The latter is responsible for monitoring the application and performing scalability rule actions when respective events are triggered. The MDDB is the medium for model-based communication between the previous two modules. It offers a persistent storage for models as well as capabilities for sophisticated querying, event listening and added-value fact derivation via a Knowledge Base.

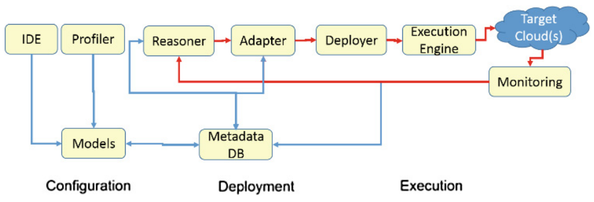


Fig. 1. Deployment workflow

Figure 1 shows a subset of each module's components reflecting the flow of information taking place during a cloud application's deployment and provisioning. Initially, user requirements and application specifications defined in CAMEL models are provided via an IDE and stored in the MDDB. The profiler then exploits these models to construct the application's profile and a constraint optimisation problem reflecting user Service Level Objectives (SLOs) and optimisation requirements. The Reasoner then solves this problem to discover the best possible deployment plan for the application. The Adapter inspects the deployment plan derived to find the differences between the initial application configuration and the proposed one such that a minimal deployment action set can be derived prescribing the way the proposed configuration can be reached from the initial one. Executionware finally closes the runtime loop by executing this minimal action set and deploying a measurement and assessment system to monitor and detect SLO violations as well as react on them via executing scaling actions described in local scalability rules that have been triggered.

However, Executionware cannot handle two cases: (a) the scalability limits prescribed by the application configuration are reached preventing scalability rules from being executed; (b) no scalability rule can address the current situation leading to user SLO violations. In both cases, Executionware informs the Upperware to perform global adaptation to close the design time loop by reaching a new application configuration which is SLO-violation free. This is achieved by (a) the Reasoner which computes a new application configuration by considering initial user requirements and current application situation (e.g., current load led to violating SLOs) and (b) the Adapter which derives a minimum deployment action set to reach the new configuration.

2.2 Security Aspects Consideration

Apart from the need to specify security requirements and capabilities, there are two steps in the previous deployment flow where security must be considered to avoid security level deterioration as well as increase trust in cloud computing by application providers: (a) during reasoning to drive the derivation of a suitable deployment plan by filtering cloud providers/services not satisfying the user-specified requirements; (b) during runtime adaptation by using security-based scalability rules indicating the security SLO violations that must lead to executing particular adaptation actions.

Our previous work [2] partially addressed the first step by advocating using two types of requirements to filter cloud providers: security controls and security SLOs. This work proposes a solution fully covering both steps as follows: (a) we advocate considering security metrics in optimisation functions to drive the selection of deployment plans which will tend not to violate the security requirements; (b) we extend the sub-DSL of scalability rules called SRL [4] to transform it into an adaptation rule DSL by covering both performance and security conditions for adaptation events and considering different adaptation actions apart from scaling ones; (c) we extend the PaaSage measurement and assessment architecture.

2.3 Security Modelling

We have proposed [3] a minimal but quite suitable security meta-model which was integrated into CAMEL. Figure 2 depicts this meta-model comprising a small set of concepts to be used for describing security requirements and capabilities. The main starting point is the *SecurityModel* concept which represents a security model and is the container for respective security elements that must be specified. The main security concepts which can be instantiated in a security model are described below.

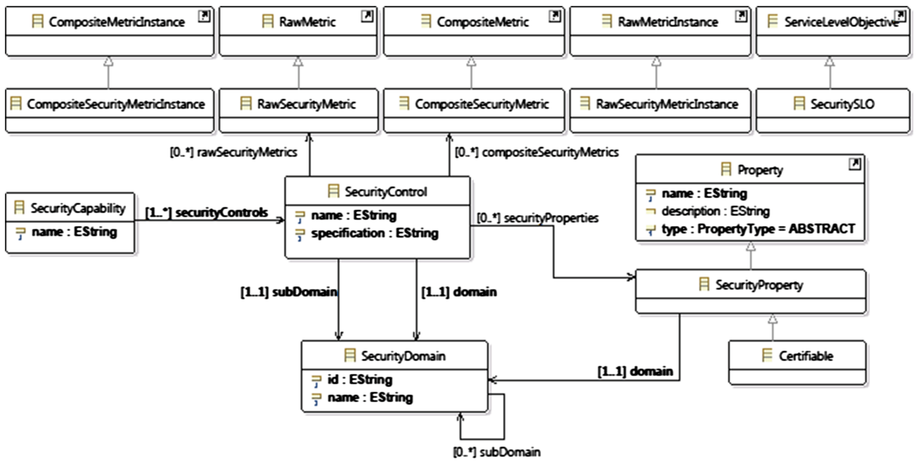


Fig. 2. Security meta-model

A *SecurityProperty* represents a security property at any abstraction level from abstract to certifiable properties (represented by *Certifiable* concept). Such a property can be measured by a raw security metric or a composite one. The former type of metric is immediately computed from security sensors, while the latter is computed from lower-level metrics by executing an aggregation formula.

A *SecurityControl* represents a security control that may have to or has been realized by a cloud provider. It constitutes the main construct for building high-level security requirements as security controls represent coarse grained reflections of the overall security level exhibited by a cloud provider. It also relates to security metrics and properties. As such, we can go from high-level security requirements to low-level ones and thus evaluate whether a promised high-level security level is delivered.

A *SecurityRequirement* can be refined into security control requirements and security SLOs. A Security control requirement comprises a set of security controls required from all cloud providers whose services are to be used for user application deployment. A *Security SLO* is a type of SLO involving the evaluation of security metric or property conditions. To symmetrically represent requirements and capabilities, facilitating their proper matching, a *SecurityCapability* is refined into security control capabilities and SLOs promised by cloud providers.

The security meta-model suitability and extensiveness is guaranteed through its integration with CAMEL's scalability rule (SRL) [4] and the requirement meta-models. Both security metrics and properties are sub-classed to SRL metrics and properties thus allowing their full specification by using the complementary information enabled by these SRL concepts, including metric scheduling and aggregation formulas. Via this integration, conditions on security metrics and properties can also be expressed by using the *MetricCondition* and *PropertyCondition* concepts, which are sub-classes of *Condition* used to specify SLOs in CAMEL requirement meta-model. By also sub-classing security SLOs to SLOs, we can then fully support specifying security SLOs.

Another integration point with the requirement meta-model is *OptimisationRequirement* enabling the specification of (security) optimisation requirements as optimisation functions over a (security) metric or property for an application or component. Such requirements are soft as they are directives to the Reasoner on how to optimize the constraint problem considered. They also come with a capability to include a priority dictating the relative significance of one requirement over the others for the user.

2.4 Reasoning

The basic reasoning process filters the provider space based on VM and quality requirements and attempts to find the best possible deployment solution based on a simple optimisation requirement at the global level considering the aspect of cost or quality (e.g., minimise cost or maximise application availability). This basic process was extended by our previous work [2, 3] to become security-oriented. In particular, our security meta-model allows the attachment of both types of security requirements at the global level, holding for the whole application, and the component/local level, holding for an application component. These security requirements are then used to further filter the provider space based on the next matching procedures.

The matching of security controls involves checking whether the security controls required are covered by those realized by the current cloud provider. More formally, by denoting as SC_R the set of security controls required and as SC_P the security controls provided, the security control matching can be expressed as follows:

$$match(SC_R, SC_P) = (SC_R \subseteq SC_P) = (SC_R - SC_P == \emptyset)$$

SLOs matching checks whether the SLO promised is stricter than that required. More formally, if the required SLO_R is expressed by condition $X\ op_1\ v_1$ and provided SLO_P is expressed as $X\ op_2\ v_2$, where X is a security metric, op_i represent comparison operators and v_i values from the metric's value type, SLO matching can be expressed as:

$$match(SLO_R, SLO_P) = \begin{cases} v_1 = v_2 \wedge (op_1 = op_2) \\ v_1 > v_2 \wedge op_1 = (\leq | <) \wedge op_2 = (\leq | <) \\ v_1 < v_2 \wedge op_1 = (\geq | >) \wedge op_2 = (\geq | >) \end{cases}$$

The matching is performed by transforming the user requirement and deployment model to a constraint problem, where all globally required security controls are moved to the local level to be satisfied for each application component. Then, security matching for each application component and each cloud provider offering a compatible VM for this component maps to matching all security requirements posed and security capabilities exhibited by the cloud provider. If there is no matching for a security requirement, the respective VM offering is discarded from further reasoning.

Our previous work did not consider global security SLOs as this requires combining and aggregating security metrics across the selected cloud providers for each application component. Such global security SLOs are not widely used and only local security SLOs are usually exploited. However, in this work, we advocate using such security SLOs to also express global metrics involved in suitable optimisation functions and thus be able to derive optimized deployment plans guaranteeing the user-required security levels.

The deployment plan reasoner proposed considers simple optimisation functions involving simple metrics like cost. Such functions can be enhanced not only based on the aforementioned way but by also considering performance metrics aggregated across some or all application components, as Sect. 4 will show. For instance, an application's execution time could be derived by adding its service components' execution time, if this application has a set of service components sequentially executed.

3 Deployment Security Monitoring and Adaptation Approach

This section presents our security oriented extension of the PaaSage deployment workflow and architecture. First it presents the modelling extensions to CAMEL [1] to support the design-time and runtime adaptation loop, it then analyses how the constraint optimization problem is enhanced by using particular utility functions to close the design-time adaptation loop and explains how the PaaSage architecture was extended according to which components. It finally explicates how our solution can be applied as well as some guidelines to follow.

3.1 Modelling Extensions

To support both design-time and runtime adaptation loops, a slight and a more significant extension to the security and SRL meta-models were performed, respectively. The security meta-model was extended by adding a new concept called *SecurityOptimisationRequirement*, sub-concept of *OptimisationRequirement* which indicates whether the values of a particular metric must be minimized or maximized at the level of application or its components, to declaratively support the specification of security optimisation requirements associated only to security metrics. An example of a security optimisation requirement would be to maximize the time between incidents for an application.

OCL constraints are used for checking associations and overall domain semantics. For instance, in case of a *SecurityOptimisationRequirement*, an OCL constraint indicates that the metric association inherited from the parent *OptimisationRequirement* should map to a metric which is an instance of *SecurityMetric*. By using OCL and UML-based modelling constraints, users are enforced to generate only structurally and semantically valid security models. This is a great added-value feature of CAMEL with respect to other cloud-oriented languages.

The SRL extensions focused on specifying adaptation rules aiming at transforming SRL into an adaptation rule DSL which covers the security aspect as well as enables the execution of adaptation workflows. Figure 3 depicts these extensions, highlighted with a grey colour with respect to the original concepts, which include:

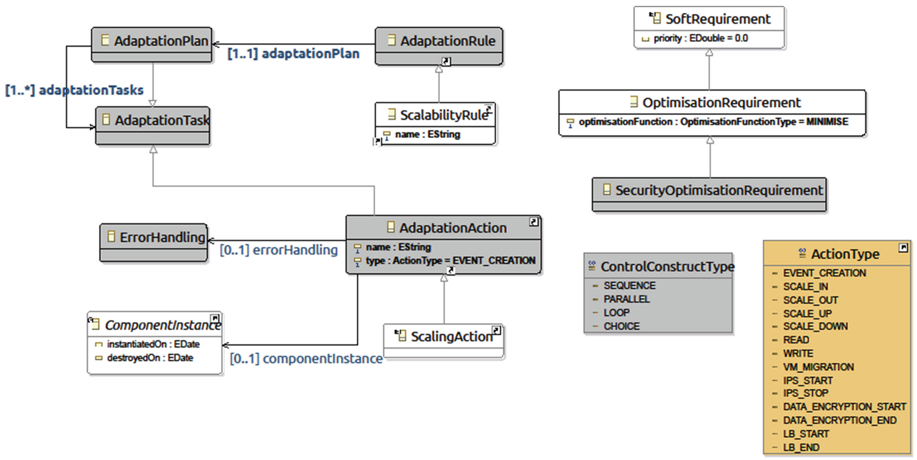


Fig. 3. SRL meta-model extensions

1. Introducing *AdaptationRule* as a super-concept of *ScalabilityRule*. A scalability rule is a mapping from a single (mapping to a particular condition) or composite event (a combination of events via time or logical operators) to one or more scaling actions which attempt to increase or decrease the cloud resources available for the application at hand. Both horizontal (scale-in & scale-out) and vertical (scale-up & scale-down) scaling actions can be expressed. On the other hand, an adaptation rule subsumes a scalability rule as it can express adaptation handling cases at different levels apart from the resource one, such as the level of services or business processes, and for different aspects (including cross-cutting ones like security). As the simultaneous handling of different levels can include a complex adaptation logic, an adaptation rule is now able to map a single or composite event to an adaptation plan and not to a set of adaptation actions to be sequentially executed. Concerning security, this means that we will be able to adapt the application at hand according to the current context by invoking one or more security adaptation actions (e.g., start encryption software). An example of an adaptation rule for security is the following: $E_1 \rightarrow \text{DATA\_ENCRYPTION\_START}$, where $E_1 = mtbi < 4\text{min}$ and mtbi is the mean time between incidents metric.
2. Introducing *AdaptationPlan* concept representing an adaptation plan to be executed when the associated adaptation rule is triggered. This plan allows executing adaptation actions in a workflow manner to support other workflow control constructs apart from the sequence. An adaptation plan maps to a workflow construct with the *ControlConstructEnumeration* as its type comprising all basic workflow constructs. It is also associated to an adaptation task set to be executed based on the semantics of the referenced control flow construct. An *AdaptationTask* is a super-concept of *AdaptationPlan* and *AdaptationAction*. As such, we enable creating a tree of adaptation plans with only concrete adaptation actions as its leaves and thus express quite complex adaptation workflows. For instance, we can express that two or more security-based adaptation actions can be sequentially executed when there are dependencies between them such that one must start before the other or in parallel when we have complementary security actions that can cooperatively remedy a certain security vulnerability or problematic situation.
3. *Action* is renamed to *AdaptationAction* to clearly indicate that adaptation is performed. *ScalingAction* is now a sub-concept of *AdaptationAction*. An adaptation action, apart from indicating the action type to be performed, is related to a particular component (instance of *Component* in CAMEL's deployment meta-model) to be adapted. As a component specification in the deployment meta-model is related to a certain configuration, the PaaSage platform will have all appropriate information to configure, enact and manage the life-cycle of components responsible for executing adaptation actions. An action also relates to an *ErrorHandling* used to indicate what should happen when the execution of this action fails.
4. The *ErrorHandling* concept includes the semantics of the way errors can be addressed based on a specific enumerated type. Depending on the semantics type, additional information might be specified in form of an adaptation plan/action.
5. The *ActionType* enumeration was extended to include other types of adaptation actions. These types include VM migration, starting/stopping IPS software, starting/stopping data encryption software and starting/stopping of load balancers.

3.2 Optimisation Function Specification

We envisage that deployment planning is a multi-objective optimisation problem which has to consider different aspects, including performance, cost and security, as there are trade-offs between them that have to be taken into account so as to derive the best possible deployment solution. To this end, we need to transit from a single cost objective to a full-fledged optimisation function able to express these trade-offs.

This transit can be realized by considering the Analytical Hierarchy Process (AHP) [11] and Simple Additive Weighting (SAW) technique [10] whose combination has been widely considered in the context of service composition. AHP can be used to indicate the relative significance of each optimisation criterion to capture the respective trade-offs. Thus, by following this widely used procedure, we can provide meaningful priorities to optimisation requirements. SAW can then be used to transform the current problem from a multi-objective to a single-one by specifying that the overall optimisation objective is to minimise the weighted sum of the partial utility functions for each initial objective. So, the problem now becomes how to formulate these partial utility functions to complete the specification of the overall optimisation function.

Such utility functions should first normalize the values of the respective metrics involved to establish the respective trade-off in a fair and uniform manner. This means that each utility function should uniformly map each value of a metric to a utility taking values from a common value type usually mapping to the set of [0.0, 1.0].

Depending on metric monotonicity, two types of piece-wise linear utility functions are mainly used, the first mapping to positively monotonic metrics (e.g., mean time between incidents - MTBI) and the second to negatively monotonic (e.g., cost). Such utility functions cannot solve the over-constrained user requirements problem but can be updated based on the approach in [9] to allow a slight requirement violation to still propose a solution to the user. Such an exception must be reported to assist users in deciding whether to proceed with the deployment indicated by the plan generated, but provides an added-value to them in contrast to other approaches that require modifying the requirements without indicating the exact violation reason to support this modification. The final utility functions are specified below in a consolidated manner:

$$f_n(x) = \begin{cases} a_n + \frac{q_n^{max}-x}{q_n^{max}-q_n^{min}} \cdot (1-a_n), & q_n^{min} \le x \le q_n^{max}, x \downarrow \\ \max\left(a_n - \frac{q_n^{min}-x}{q_n^{max}-q_n^{min}} \cdot (1-a_n), 0\right), & x < q_n^{min} \\ \max\left(a_n - \frac{x-q_n^{max}}{q_n^{max}-q_n^{min}} \cdot (1-a_n), 0\right), & x > q_n^{max} \end{cases},$$

$$f_n(x) = \begin{cases} a_n + \frac{x-q_n^{min}}{q_n^{max}-q_n^{min}} \cdot (1-a_n), & q_n^{min} \le x \le q_n^{max}, x \uparrow \\ \max\left(a_n - \frac{q_n^{min}-x}{q_n^{max}-q_n^{min}} \cdot (1-a_n), 0\right), & x < q_n^{min} \\ \max\left(a_n - \frac{x-q_n^{max}}{q_n^{max}-q_n^{min}} \cdot (1-a_n), 0\right), & x > q_n^{max} \end{cases}$$

where a_n represents the percentage of values allowed outside the required user range for the metric, q_n^{min} and q_n^{max} are the lowest and highest values this metric can take. To explain, these utility functions work as follows: (a) in case the value is within the user-provided range, the utility linearly increases from a_n to 1.0 depending on the distance of the value from the worst value in the range (e.g., in case of cost and a range of 4-8 euros, the best utility is 1.0 obtained when the cost value maps to the best bound of 4 euros); (b) otherwise, the utility linearly decreases from a_n to 0.0 depending on the distance of the value from the closest bound (e.g., based on previous example and a_n equal to 0.4, then both the values of 9 and 3 euros will map to a 0.2 utility).

In this respect, the final optimisation function to be maximized is specified below, where w_n reports the weight given to each metric, uf_n represents the metric's utility function and v_n represents the overall metric value for the current deployment solution under consideration by the Reasoner: $\sum\limits_{n} w_n \cdot uf_n(v_n)$.

As utility functions are specific to the metric type involved, the user input, enabling the automatic generation of complete optimisation functions, comes in form of priorities to the optimisation requirements posed (mapping to the metrics to be optimized) and the configuration of the respective metric utility function based on the a_n parameter. To enable the latter, *OptimisationRequirement* was slightly extended to involve the suitable attribute definition. In result, the extended PaaSage system can take the limited user input and automatically produce an optimisation function which respects user optimisation requirements leading to constructing a suitable constraint optimisation problem which enforces all requirements posed based on all aspects considered.

3.3 Extended PaaSage Architecture

The current PaaSage architecture was extended via the following additions: (a) vulnerability scanning probes based on the OpenVas[1] software scanning the reports produced and producing values for particular raw security metrics, including MTBI; (b) the realization of an AdaptationEngine component extending Executionware's adaptation functionality by using the Activiti[2] Business Process Management engine, the functionality mapping to executing IPS and data encryption software as adaptation actions in form of a security library and the transformation functionality from *AdaptationPlan*s to BPMN models; (c) our implementation of the Reasoner to be selected from those available by the *MetaSolver* component by relying on the type of constraints involved in the constraint problem (i.e., security requirements in our case).

The security library was extended with the next open-source software: (a) fail2ban[3] which scans logs files and bans IPs for a certain amount of time that exhibited malicious behaviour; (b) Snort[4], a network based intrusion detection and prevention system (IDPS) using signature, anomaly and protocol methods to detect cyber attacks; (c) OSSEC[5], a host-based IDPS exploiting signature and profile methods to detect cyber attacks; (d) Suricata[6], a network based intrusion detection system exploiting a

[1] www.openvas.org.

[2] activiti.org.

[3] www.fail2ban.org.

[4] www.snort.org.

[5] www.ossec.net.

[6] suricata-ids.org.

signature based method; (e) AESCrypt[7], a multi-platform file decryption software (FDS) based on AES; (f) DiskCryptor[8], a Windows-based FDS relying on AES 256, Twofish, and Serpent; (g) VeraCrypt[9], a multiplatform decryption software relying on the same ciphers as with DiskCryptor. IPS, host and network based software can be combined to reach a more sophisticated protection solution.

Figure 4 depicts the extended PaaSage architecture, showing that our vulnerability scanning probe along with other user-provided probes, needed to compute application-specific metrics, and the security library, to enable executing security adaptation actions, are deployed on user VMs. In each user VM, there is also a management component handling the lifecycle management of the VM and the application and security components that it contains. As such, we can support deletions of component and VM instances when the need arises or execute particular security adaptation actions on respective VMs to remedy the security problems encountered.

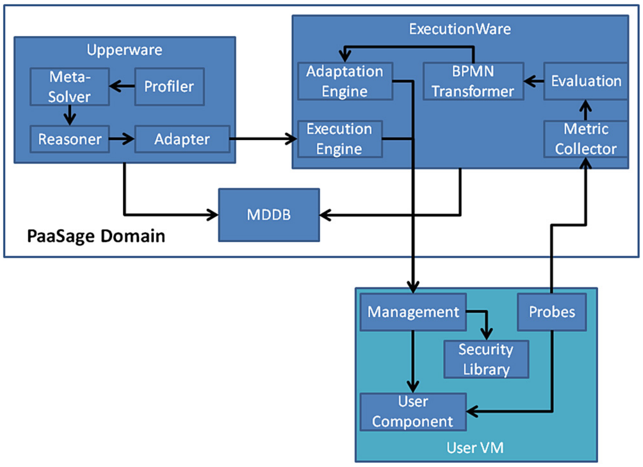


Fig. 4. Extended architecture

The *AdaptationEngine* is deployed in the PaaSage domain and inside the *ExecutionwareEngine*. This engine is informed when an adaptation plan must be performed based on the respective adaptation rule triggering. The *Evaluation* component hands over the adaptation plan to the *BPMNTransformer* component which maps it to a BPMN model and sends it for execution to the *AdaptationEngine*. This transformation needs to differentiate between the different adaptation actions involved so as to specify the respective task in the BPMN business process model. For scaling and migration actions, the task specification maps to executing the *ScalingAdapter* service.

[7] https://www.aescrypt.com/.
[8] https://diskcryptor.net/wiki/Main_Page.
[9] https://veracrypt.codeplex.com/.

For security actions, the task specification maps to executing the *Management* component in the appropriate VM instance. Each component call requires input information derived from the deployment plan specification and from MDDB via respective associations. For instance, a security adaptation action maps to the VM on which the adaptation must be performed and the *BPMNTransformer* has to obtain additional information for this VM, including its public IP address, to formulate the suitable service call.

Our Reasoner component is deployed in the PaaSage domain as part of the Upperware module. Its implementation relied on the Choco solver[10], an open-source Constraint Programming solving engine that: (a) supports both integer, set, and real variables; (b) realized various state-of-the-art constraint propagation algorithms; (c) produces explanations to report those user requirements not satisfied by current cloud offerings.

3.4 Solution Application Guidelines

In this subsection, we explain how our solution can be applied in overall for applications and which are the concrete guidelines to follow. The main idea is that there is usually a trade-off between security, performance and cost which should be investigated. In particular, users might desire deployment solutions which could be less costly but not certain that will guarantee the security level envisaged or prefer costlier solutions from the very beginning to ensure that the desired security level is always maintained. To this end, we advocate exploiting two alternative solutions: (a) either the user does not provide any security optimisation requirement or (b) he/she does provide them. The first solution does not imply that the second solution will not be finally applied during application runtime when undesirable security situations are reached. Let us not explain in detail what is the main outcome of these solutions and which ones to select for which cases.

The first solution can lead to a deployment plan which might not reflect the security level envisaged. However, the user can rely on reacting to security related events via adaptation rules such that free security-related adaptation actions are performed that may impact application performance but could resolve the issues encountered. If performance is not much deteriorated, the solution would be suitable, especially as adaptation rules can cater for scenarios where opposite security adaptation actions are fired to enhance application performance when the security situations have been adequately addressed.

In case the performance reaches undesirable levels or the security level is still low, this is an indication that more drastic measures have to be taken which can take the form of including and executing adaptation rules which perform a migration or a new global deployment or resorting to the second solution through specifying security optimisation requirements as well as imposing the exploitation of security services. Migration might solve the security problems encountered but it induces an overhead

[10] www.choco-solver.org.

which could deteriorate application performance. Moreover, it is not certain that security problems will not re-appear, especially if the same cloud is selected for migration. A re-deployment can be an even worse solution as it leads to performing significant deployment adaptations while it is not certain that it can be security-issue free. To this end, it might be better to resort to the second solution or at least assess the costs incurred for performing more costlier and time-consuming adaptation actions before proceeding with such a move.

In the case of the second solution, the user is certain that either security issues will be resolved or there will be a remedy for SLO violations by the respective cloud provider. Moreover, while the exploited security services will not come for free they might be more lightweight with respect to open-source ones and optimized for the cloud at hand, thus not deteriorating so much application performance when run on user VMs. However, apart from performance deterioration, deployment cost is also increased. If this is acceptable, the new deployment solution might be more suitable from the initial one as it will tend to keep the user-required security level. Overall resorting to the second solution looks as a more informed decision from the user side if it can be tolerated based on the user budget and requirements.

4 Application to Use Case

Our approach is now applied on a particular use case to highlight its main benefits. This use case concerns a traffic monitoring application executed in certain city areas to regulate traffic and sustain certain noise and pollution levels. Such an application comprises 3 main components: (a) a monitoring component sensing the current noise, pollution levels, traffic size and patterns in the designated area; (b) a traffic analysis (*Anal*) component which analyses traffic and proposes particular traffic reconfiguration plans; (c) a traffic configuration (*TC*) component executing the produced plan. As a municipality responsible for the area deploys such an application, there is a need of a particular security level to prevent adversary users from monitoring any traffic configuration decisions or even modifying them to create a traffic chaos. Thus, as we have a hybrid cloud scenario here where the first and last component are deployed in a municipal cloud, the analysis component has to be as secure as possible.

Table 1 shows the requirements posed for *Anal* in terms of deployment, security, cost and performance. Such requirements map to the global level as there are no major adaptation alternatives for the first and third component with respect to the second one. The security controls mentioned originate from the Cloud Control Matrix [5] of Cloud Security Alliance and have been already embraced by the research community and industry. Each security control's meaning is as follows: AAC-02 maps to conducting independent reviews and annual provider assessments, DSI-01 maps to the ability to classify data and services based on various criteria, DSI-05 maps to preventing data leakages, and TVM-02 maps to the timely vulnerability detection. Thus, the municipality is interested in a cloud provider which is constantly assessed, can prevent data leakages and timely detect security vulnerabilities. Such high-level security requirements are reasonable as the selection of provider should minimize the various vulnerability types that can be involved in the VM hosting *Anal*. The security SLO

Table 1. Requirements for deployment, security, cost and performance

Component	VM req.	Security controls	Security SLOs	Perform. req	Cost
Analysis	4 cores, 4 GBs of main memory, 40 GBs of hard disk	AAC-02, DSI-01, DSI-05, TVM-02	*mtbi* > = 1 h	exec. time <= 35 s, avail > = 99.99 %, min (exec.time):2.0, max (avail):2.0	<= 350$ per month, min (cost): 3.0

sensibly posed indicates that security incidents occur as rare as possible. The deployment requirements map to increased needs for *Anal* leading to "high" VM requirements. As *Anal* is a crucial component, extra requirements map to suitable performance and availability levels to address traffic situations that can unexpectedly occur.

Depending also on the respective metric monotonicity, different priorities have been put to different requirement types. In particular, cost minimisation is considered the topmost requirement followed by the minimisation of response time and the maximisation of availability which are equally important. Initialy no optimisation requirement had been specified on security metrics. This makes sense as the user is more interested in obtaining a solution mapping to the lowest possible application deployment cost.

Table 2 shows the VM & security offerings as well as security capabilities of three cloud providers (based on realistic information drawn from existing providers), where each offering comes with a particular cost per hour. By simply comparing Tables 1 and 2 we can easily see that the third cloud provider, while offering a suitable VM offering,

Table 2. VM, security offerings, security capabilities of three cloud providers

Provider	VM offering	Security controls	Security capabilities	Security offering
A	(1) 4 cores, 7.5 GB, 80 GB → 0.210 $ per hour (2) 4 cores, 15 GB, 80 GB → 0.280$ per hour	AAC-02, AAC-03, DSI-01, DSI-05, EKM-03, TVM-02, SEF-05	*mtbi* > = 1 h	IPS_A → 0.300 $ per hour
B	(1) 4 cores, 4 GB, 130 GB → 0.22 $ per hour	AAC-02, AAC-03, DSI-01, DSI-05, EKM-03, TVM-02, SEF-05	*mtbi* > = 2 h	IPS_B → 0.250 $ per hour
C	(1) 4 cores, 4 GB, 40 GB → 0.1$ per hour	AAC-02, AAC-03, DSI-05, EKM-03, TVM-02	mtbi > = 0.9 month	

has not realized a security control (DSI-01) and violates the security SLO posed. By considering the rest of suitable VM offerings, the Reasoner will derive the solution of choosing VM offering 1 of provider A to deploy *Anal* there.

The rationale of this choice depends both on the offering cost and all optimisation objectives provided as VM offerings affect also the performance level to be exhibited. In fact, performance metrics, like execution time, can be expressed as a linear resource metric combination by following an approach such as the one in [12], thus introducing a certain correlation to completely understand the benefits of obtaining a VM with better characteristics than those baseline required ones. As such, by considering an optimisation function derived from the optimisation requirements posed, VM offering 1 of A has the best utility as it is much cheaper than offering 2 in the same provider and offering 1 of B, although it's not memory optimized, and leads to a performance level similar to those to be exhibited when the rest of offerings are exploited. The fact that the security SLO promised by provider A is weaker than that promised by provider B will influence subsequent user decisions, as we will see later.

Suppose now that the following adaptation rules have been designed to regulate application adaptation behaviour.

$mtbi < 1 \ \& \ (\text{size}(sec\_sw) == 0) \rightarrow \text{execute}(Snort)$
$mtbi < 1 \ \& \ (\text{size}(sec\_sw) == 1) \rightarrow \text{execute}(OSSEC)$
$mtbi < 1 \ \& \ (\text{size}(sec\_sw) == 1) \rightarrow \text{migrate}(Anal)$

The first rule indicates that when the security SLO posed is violated and no security software runs at the *Anal* VM, the Snort network-based IPS software must be started. The second rule indicates to start the OSSEC IPS software, which has complementary functionality to Snort, when the SLO is still violated and a security software already runs. If either the security SLO is still violated with the running of the two IPS software or the *Anal* performance SLO is violated, the organisation has to decide whether to migrate or to go to a new deployment solution.

While the constraint on *mtbi* is about 1 h, its evaluation occurs more constantly to react as quickly as possible to an undesired security situation. Otherwise, it will not make sense to execute just one security adaptation action and then wait for 1 h to check whether no vulnerability has occurred.

By considering the existence of an administrator in the municipality, we can check whether the designed adaptation resolution behaviour is correct by using the PaaSage platform's monitoring dashboard. In case the same security problems occur even after migrating to new machines, the user requirements must be modified to include security optimisation ones and using security services. In our case, the new requirements involve exploiting an IPS cloud-specific service and requiring that MTBI is maximized with priority equal to 4 (the highest). Based on these new requirements, cloud provider B will be selected as it offers a higher security level and cheaper IPS service. Compared to the first solution, cost is higher. However, a specific security service is used guaranteeing, based also on administrator experience, MTBI's maximization.

5 Related Work

Security meta-models. The various security meta-models proposed [6–8, 13, 14] lack the expressiveness and richness of our meta-model. They usually focus on one security aspect or neglect appropriate measurement details that come with the specification of security metrics. They also lack a proper linking of concepts at different levels of abstraction with the sole exception of the CUMULUS security meta-model.

Security-based deployment reasoning. Our previous work [2, 3] is one of the initial attempts to tackle this area in multi-cloud computing. It can handle both high- and low-level security requirements. It initially relied on optimising just a cost objective but now is extended to automatically derive more complicated optimisation functions directly from user optimisation requirements. [15] presents an approach relying on the elicitation of high-level security goals to particular security constraints that are then exploited to instantiate a particular deployment template with IaaS and security services. This approach could complement our solution towards making a more informed selection of which actual security services to select as part of the application deployment. [16] can analyse the security properties of complex configurations of systems to discover possible vulnerabilities. Such an approach could be exploited a posteriori, after a deployment plan is produced, to select the respective security services resolving the detected vulnerabilities by considering the trade-offs between the various user requirements (e.g., cost and performance) and current cost of initial deployment plan.

Security-based cloud adaptation. While various approaches focus on exploiting a cloud's elastic capabilities to address increased demand by scaling out applications, there is no approach dealing with multi-cloud adaptation. Most cloud-based application adaptation work also focuses on the application's resource usage or also performance level. Such work usually employs scaling rules indicating how to detect the scaling needs and resize the application. Alternatively, other methods include predicting application performance decrease and executing scaling actions to remedy it. While rule-based approaches provide a nice and quick solution to the current situation, they require a good expertise level for rule specification and a suitable monitoring infrastructure supporting timely rule condition evaluation. PaaSage provides a better solution relying on two principles: (a) for cloud-specific adaptation, rules are used to quickly react to the current situation; (b) when rules fail or a situation occurs not covered by adaptation rules or concerning the global application context, there should be an application reconfiguration to better match this situation and guarantee the user requirement satisfaction. This is why incorporating our security solution will make the PaaSage prototype capable to completely handle all possible local and global situations according to not only performance and cost but also the security aspect.

6 Conclusions and Future Work

This paper has presented a security solution for multi-cloud applications enabling their adaptation at runtime to guarantee a certain security, cost and performance level. This solution will be integrated into the PaaSage platform to enable a complete handling of the security aspect in the management of the multi-cloud application lifecycle. The proposed solution comprises: (a) security-oriented and expressiveness extensions to the security and scalability meta-models of the CAMEL DSL; (b) extensions to the PaaSage platform architecture related to executing composite adaptation plans, monitoring security SLOs and executing security issue resolution software, such as IPS software, in the context of adaptation plans to address the security situations encountered and increase the security level exhibited.

Apart from the proposed solution, specific guidelines are proposed explaining the ways this solution can be applied to cater for different application security, performance and cost requirements by also explicating certain requirements imposed on the devop team of the application owner and certain ways to specify appropriate security-oriented adaptation rules. These guidelines along with the solution have been validated according to a particular use case showing the main benefits of our approach.

The next work directions are planned: (a) integrating our solution in the PaaSage platform and thoroughly evaluating it in different use cases/real circumstances; (b) extending adaptation functionality to include additional open-source software to drive executing more sophisticated security-based adaptation rules; (c) developing security adaptation rule patterns to be re-used/customized by application owners; (d) extending adaptation rule language to cover extra adaptation scenarios; (e) checking different optimisation function forms to investigate whether they can lead to better application deployments with a much less probability of required security level violations.

Acknowledgements. This work is partially funded by the EU FP7 PaaSage project.

References

1. Rossini, A., Nikolov, N., Romero, D., Domaschka, J., Kritikos, K., Kirkham, T., Solberg, A.: D2.1.2 – CloudML Implementation Documentation. Paasage project deliverable, April 2014
2. Massonet, P., Luna, J., Pannetrat, A., Trapero, R.: Idea: optimising multi-cloud deployments with security controls as constraints. In: Piessens, F., Caballero, J., Bielova, N. (eds.) ESSoS 2015. LNCS, vol. 8978, pp. 102–110. Springer, Heidelberg (2015)
3. Kritikos, K., Massonet, P.: An integrated security meta-model for run-time and design time cloud deployment adaptation. Submitted to IEEE Transactions on Cloud Computing (2015)
4. Kritikos, K., Domaschka, J., Rossini, A.: SRL: a scalability rule language for multi-cloud environments. In: CloudCom 2014, pp. 1–9 (2014)
5. Cloud Control Matrix (2011). http://www.cloudsecurityalliance.org/cm.html
6. Pannetrat, A.: D2.1: Security-aware SLA specification language and cloud security dependency model. Cumulus Project Deliverable (2013)
7. The Center for Internet Security: The CIS security metrics v.1.10, USA, Technical report 28 (2010)
8. Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., Robinson, W.: Performance measurement guide for information security. National Institute of Standards and Technology, USA, Technical report, July 2008

9. Mello Ferreira, A., Kritikos, K., Pernici, B.: Energy-aware design of service-based applications. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 99–114. Springer, Heidelberg (2009)
10. Hwang, C., Yoon, K.: Multiple Criteria Decision Making. Lect. Notes Econ. Math. (1981)
11. Saati: The Analytic Hierarchy Process. McGraw-Hill (1980)
12. Xiong, P., Pu, C., Zhu, X., Griffith, R.: vperfguard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments. In: ICPE, pp. 271–282. ACM, New York (2013)
13. Fenz, S., Ekelhart, A.: Formalizing information security knowledge. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS 2009, pp. 183–194. ACM, New York (2009)
14. Almorsy, M., Grundy, J., Ibrahim, A.: Adaptable, model-driven security engineering for saas cloud-based applications. Autom. Softw. Eng. **21**(2), 187–224 (2014)
15. Kang, E., Jackson, D.: A model-based framework for security configuration analysis (manuscript, 2012)
16. Kalloniatis, C., Mouratidis, H., Islam, S.: Evaluating cloud deployment scenarios based on security and privacy requirements. Requirements Eng. **18**(4), 299–319 (2013). ISSN: 0947-3602

AdIDoS – Adaptive and Intelligent Fully-Automatic Detection of Denial-of-Service Weaknesses in Web Services

Christian Altmeier[1](✉), Christian Mainka[2], Juraj Somorovsky[2], and Jörg Schwenk[2]

[1] Software AG, Saarbrucken, Germany
Christian.Altmeier@softwareag.com
[2] Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany
{christian.mainka,juraj.somorovsky,joerg.schwenk}@rub.de

Abstract. Denial-of-Service (DoS) attacks aim to affect availability of applications. They can be executed using several techniques. Most of them are based upon a huge computing power that is used to send a large amount of messages to attacked applications, e.g. web service. Web service apply parsing technologies to process incoming XML messages. This enlarges the amount of attack vectors since attackers get new possibilities to abuse specific parser features and complex parsing techniques. Therefore, web service applications apply various countermeasures, including message length or XML element restrictions. These countermeasures make validations of web service robustness against dos attacks complex and error prone.

In this paper, we present a novel adaptive and intelligent approach for testing web services. Our algorithm systematically increases the attack strength and evaluates its impact on a given web serice, using a blackbox approach based on server response times. This allows one to automatically detect message size limits or element count restrictions. We prove the practicability of our approach by implementing a new WS-attacker plugin and detecting new DoS vulnerabilities in widely used web service implementations.

1 Introduction

Motivation. The principle of the Service Oriented Architecture (SOA) technology is to build a system of devices and machines connected via web service. A typical web service technology to establish SOA is SOAP [9]. SOAP-based web service are built upon the platform independent markup language – XML. They are used in business to business (B2B) process integrations and supported by large vendors like IBM and Axway.

The availability of web service in SOA scenarios is of huge importance. Therefore, dos attacks on web service present a significant risk. Denial-of-Service (DoS) attacks attempt to exceed the consumption of computational resources, like CPU

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 65–80, 2016.
DOI: 10.1007/978-3-319-29883-2_5

time or memory, with the goal that the system is no longer available for regular use. There are numerous techniques to perform DoS attacks, but throughout this paper, we concentrate on XML-based DoS attacks [4]. XML-based DoS is a special variant of a DoS attack that targets the XML parser. This means that the DoS payload is a specially crafted XML document, for example, a message with numerous deeply nested XML elements.

Complexity of XML-based DoS attacks. Previous research has revealed many different types of XML-based DoS techniques [4]. Unfortunately, the knowledge of these attacks is only the tip of the iceberg. The real challenge is to validate whether the tested XML parser is vulnerable to them. This is a complicated task, because there are many varieties of each single attack. For example, placing the DoS payload at one position within the XML document may affect the parser and result in a successful DoS attack, but using another position, for example a sibling element, can lead to an unaffected parser. Since there are many elements to place the payload in addition to many other aspects to consider, the detection of successful attack varieties is not trivial:

1. XML parsers can be configured to restrict a specific number of elements to be parsed. Consequently, attacks using more payload elements than this specific threshold will result in unsuccessful attacks. To detect such a threshold, the DoS attacks must be executed first with a small payload and then adaptively be adjusted due to the measured results.
2. The XML document structure can be validated using XML Schema [12]. Therefore, the attack payload cannot be placed at arbitrary positions in these scenarios. We use an approach that automatically reads the used xmlschema and places the payload at so-called *extension points* in such a way that the XML document containing the DoS payload is valid against the schema.

Contribution. In our work, we concentrate on the automatic detection of XML-based DoS and the automatic bypassing of countermeasures (XML Schema validation, thresholds, ...). Our contributions are as follows:

– **AdIDoS** (Adaptive and Intelligent DoS), the first fully automatic XML-based DoS tool that detects DoS vulnerabilities with an intelligent and adaptive approach. Our tool extends the approach of [4], is open source, and part of WS-Attacker – a fully-automatic web service penetration testing framework.
– Our approach is **generic** and can be applied to XML scenarios beyond web services or even other DoS attacks beyond XML-based DoS[1].
– We evaluated **seven web service** implementations and give a detailed overview over their robustness against DoS attacks.

Outline. The following section will introduce the necessary foundations for this paper, including XML-based Web Services and XML-based DoS attacks.

[1] Our implementation in the WS-Attacker framework is split into two parts: (1) a generic library to apply DoS attacks on XML and (2) a plugin that is used to transmit SOAP messages.

Section 3 spots on the complexity on evaluating DoS attacks. In Sect. 4, we elucidate the high-level design of our Adaptive Intelligent Denial-of-Service (AdIDoS) tool, while Sect. 5 gives more details on its implementation. We evaluate AdIDoS in Sect. 6, by testing five different web service and two XML security firewalls. We discuss related work in Sect. 7 and conclude in Sect. 8.

2 Foundations

In this section, we give a brief introduction to the relevant standards and technologies for this paper.

2.1 XML and XML Schema

eXtensible Markup Language (XML) is a structured format [2] by the World Wide Web Consortium (W3C), which allows transmission, validation and interpretation of data. The interpreted data can be described independently of software and hardware, thus XML is ideal to exchange data between different applications and organizations. The structure of an XML document is defined by XML elements. An XML element typically consists of a start tag `<tag>` and an end tag `</tag>`. It can include further *child* elements, element *attributes*, or text contents.

XML Schema is a recommendation by the W3C for describing the structure of an XML document [12]. It is basically a set of rules that can describe the structure for each contained element. It covers its allowed attributes, the type of its value (e.g., a string or integer), a description of its allowed child elements and how often they may occur.

2.2 Web Services

A web service is a method for interprocess interactions over networks between different software applications. A web service can be implemented using different technologies, for example, REST [5] or SOAP [9].

In this paper, we consider the SOAP technology. SOAP (originally defined as Simple Object Access Protocol) is a W3C specification defining the structure of XML messages and a protocol to achieve a machine-to-machine communication. SOAP messages generally consist of *header* and *body*. The `<Header>` element includes message-specific data (e.g. timestamp, user information, or security tokens). The `<Body>` element contains function invocation data.

2.3 XML-based DoS Attacks

There are numerous XML-based DoS attacks. In the next section, we will give a more detailed description of the *Coercive Parsing Attack* and use this attack as a running-example in the following sections through this paper.

Coercive Parsing Attack. The Coercive Parsing attack creates a deeply nested XML document. If the document is parsed by a vulnerable service, memory exhaustion occurs. The following SOAP message gives an example with deeply nested elements.

```
<soap:Envelope xmlns:soap="...">
  <soap:Header></soap:Header>
  <soap:Body>
    <x>
      <x>
        <!-- deeply nested -->
      </x>
    </x>
  </soap:Body>
</soap:Envelope>
```

This is only one example for a Coercive Parsing attack. It is also possible to place the payload (the `<x>`-elements) in other elements, for example inside the `<Header>` element. Additionally, one can vary the number of nested elements. All these aspects affect the impact and the success-level of the attack.

Further XML-based DoS Attacks. The following XML-based DoS attacks are described in [4] and also implemented in AdIDoS:

– Coercive Parsing Attack
– XML Element Count Attack
– XML Attribute Count Attack
– XML Entity Expansion Attack
– XML External Entity Attack
– XML Overlong Names Attack
– HashCollision Attack

2.4 Attack Roundtrip Time Ratio (ARTR)

Our automatic tool AdIDoS evaluates the effectiveness of different DoS attacks against a server in a black-box manner. Thus, the only measurable metric is *time*.

We define the time of the last byte sent by a client's request up to the time of the first byte of the corresponding response as the roundtrip time:

$$RT = t_{\text{Received}} - t_{\text{Sent}}$$

If a request does not contain a DoS payload, we refer to it as an *untampered* request. Consequently, a request with DoS payload is referred as a *tampered request*.

We use the Attack Roundtrip Time Ratio (ARTR) [4] as a metric to measure the impact of each DoS attack variant and to be able to compare them. ARTR is defined as the quotient of the roundtrip time of tampered and untampered requests [4]. The higher the ARTR value is, the more effective is the attack.

2.5 WS-Attacker

WS-Attacker is a modular framework for web service penetration testing [8], available as an open source project on Github.[2] WS-Attacker uses a plugin architecture to execute XML-specific attacks on web service automatically. In its current version, WS-Attacker supports the following attacks:

1. SOAPAction Spoofing [8].
2. WS-Addressing Spoofing [8].
3. Basic XML Denial-of-Service Attacks [4].
4. XML Signature Wrapping [3].
5. Attacks on XML Encryption [7].

In this paper, we extend the functionality of WS-Attacker and implement AdIDoS as an attack plugin.

3 DoS Complexity

The complexity of DoS attacks is founded in two parts:

1. Since we assume to have *no physical access*, we can only perform black-box tests.
2. Our DoS attacks abuse weaknesses in XML parsers. As such, we need to make use of the XML document structure.

3.1 Black-Box Tests

A black-box penetration test refers to a methodology of testing a computer system without knowledge of its internals. Therefore, we do not have the possibility to measure the CPU load or memory consumption of the tested service. We only rely on the ARTR: We measure and evaluate the time that the service needs to process the request and compute the response, including the network transfer time.

3.2 XML Document Structure

XML-based DoS abuses weaknesses in the underlying XML parser. Each XML parser has its own behavior and can be adjusted to fit the service's requirements. This increases the complexity to apply a DoS attack dramatically. Some XML parserss:

1. Only process unexpected elements, if they are placed to a specific position in the XML document (they validate the XML Schema).
2. Only allow a specific number of elements or attributes in an XML document (thresholds).

[2] https://github.com/RUB-NDS/WS-Attacker.

Additionally, the service itself may restrict the amount of request by one client within a time period.

XML Schema. If an XML Schema validation is performed by the XML parser, placing the payload of an XML-based DoS attack at a specific position inside the document can be detected. The attack will not succeed. Taking the example of the Coercive Parsing attack, placing the `<x>` elements as a child of the `<soap:Envelope>` element would break the SOAP schema. However, placing the same `<x>` elements as a child of the `<soap:Header>` element is conform to the SOAP schema and the message will be accepted — the attack can potentially be applied.

In addition to the above behavior, some parsers skip specific document parts. For example, a web service that does not use *any* SOAP extensions could skip to parse the whole `<soap:Header>` element and continue with the `<soap:Body>`. This means that placing the XML-based DoS payload in the header does not result in a successful attack, while placing it in the body could.

Thresholds. Some XML parsers implement thresholds. They stop to process incoming messages if they parse more than a specific number of elements, attributes, or bytes.

Suppose an XML parser that only accepts messages up to 100 elements. Applying a Coercive Parsing attack with 5000 nested elements will result in an unsuccessful attack, but the implementation could be vulnerable if the attack is applied with, for example, 80 nested elements. To make the attack detection more accurate, it is important to start XML-based DoS attacks with a small payload, and increase it by time. This way, possible thresholds can be detected.

4 Design

In this section, we describe several principles and design decisions we followed in order to create the adaptive and intelligent XML-based DoS attack plugin AdIDoS.

4.1 Automatic DoS Detection Workflow

AdIDoS systematically tests the web service for DoS weaknesses. The detection workflow is fully automatic and AdIDoS uses the following algorithm to proceed (see Fig. 1):

1. AdIDoS chooses one attack from its pool of implemented DoS attacks.[3]
2. It specifies the position where to set the payload. Therefore, XML Schema is used to determine all matching positions.

[3] Its current implementation includes *Coercive Parsing*, *XML Attribute Count*, *XML Element Count*, *XML Entity Expansion*, *XML External Entity*, *XML Overlong Names*, and 4 variants of *HashCollision* attacks – 10 attack variants in total.

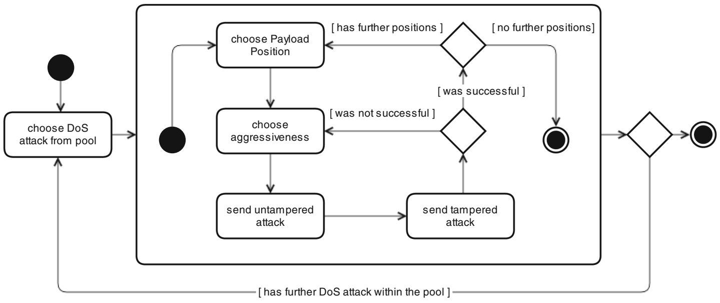


Fig. 1. AdIDoS simplified workflow of systematic DoS detection

3. The *aggressiveness* of the attack is specified. Aggressiveness means, how much XML payload is responsible for the attack. The more XML payload the attack uses, the more aggressive it is. For example, a coercive parsing attack using the payload `<x><x></x></x>` is more aggressive than an attack with `<x></x>`. Each attack variant will start with very low aggressiveness and adjust it depending on the ARTR.
4. The algorithm generates an untampered request and executes the attack against the web service. This information is used as a base line for the later decision, whether an attack is successful or not.
5. It generates a tampered request and executes the attack against the target web service.
6. It analyzes the roundtrip time of the untampered and tampered requests and decides whether the attack is successful or not by computing the ARTR (See Sect. 5.3 for details).
   – Successful: the attack is marked as successful for this payload position, the next position is specified, followed by Step 3.
   – Not successful: a more aggressive attack is set, followed by Step 4.

   This step is performed as long as further parameter sets are available for the DoS attack. Hereafter the next DoS attack is chosen and AdIDoS continues with Step 1.

4.2 Automatic Threshold Detection

The most effective countermeasure against XML-based attacks is to limit the number of elements/attributes which can occur in an XML document, or the size of the document. In our approach, we automatically detect and narrow down thresholds used by a web service. Thereby a variation of the binary search algorithm is used, which is shown in Fig. 2. The steps are as follows:

1. The threshold detection is initialized with the weakest and the strongest attack vector. The weakest vector (our minimum) is the least aggressive attack that was executed successfully. The strongest vector (our maximum) is the attack vector that was not successful.

2. The strength of each newly created attack vector is generated as an average of the weakest and the strongest aggressive attack vector.
3. The relevant information in this phase is the execution state of the attack:
   – Successful: the current attack vector strength is set as a new minimum
   – Not successful: the current attack vector strength is set as new maximum

Depending on the expected precision, these steps can be repeated several times. In our measurements, five iterations showed up to provide values giving enough information about the analyzed web service. The detected threshold is then stored in memory and can be considered for further web service investigation by the developer.

After the process of narrowing down the threshold, AdIDoS returns to its normal analysis but will now consider the detected threshold. In addition, the web service is tested for DoS weaknesses near the actual threshold.

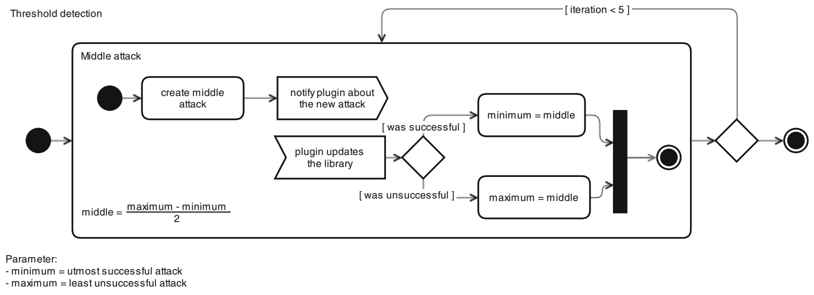

Threshold detection
[iteration < 5]
Middle attack
create middle attack
notify plugin about the new attack
[was successful]
minimum = middle
plugin updates the library
[was unsuccessful]
maximum = middle
middle = maximum - minimum / 2
Parameter:
- minimum = utmost successful attack
- maximum = least unsuccessful attack

Fig. 2. Threshold detection

5 Implementation

We implemented the concepts described in the previous section as a WS-Attacker plugin AdIDoS. In the following, we give a detailed view on some specific implementation issues.

5.1 AdIDoS for WS-Attacker

We implemented all XML-based DoS attacks listed in Sect. 2.3 as a WS Attacker plugin – called AdIDoS (Adaptive Intelligent Denial-of-Service). Each DoS attack executed by AdIDoS is a composition of multiple parameters. There are two types of attack parameters:

– **Independent** attack parameters are generic configuration parameters which can be used for all DoS attacks. Example for independent parameters are the number of used threads to send requests, or the delay between sending them.
– **Dependent** attack parameters are specific for the executed DoS attack. For example, in Coercive Parsing, AdIDoS chooses the number of nested elements.

In addition to that, there are multiple possibilities for placing the attack payload (e.g. in the `<soap:Header>`, in the `<soap:Body>`, . . .). All positions are marked in the XML message by analyzing its XML Schema. We used XML Schema parsing to automatically detect so called XML extension points.[4] These extension points can be used to place the payload without invalidating the schema. If the web service uses XML Schema validation, our generated attack messages do not harm the schema. Every DoS attack specifies where its payload can be placed:

– `ELEMENT`: the payload of an attack can be placed as a new element into the document Supported by Coercive Parsing, XML Element Count, XML Entity Expansion, XML External Entity and XML Overlong Names
– `ATTRIBUTE`: the payload can be placed within an existing element Supported by XML Attribute Count and HashCollision

5.2 Attack Configuration and Execution

By executing a concrete DoS attack, AdIDoS first uses the schema analyzer provided by WS-Attacker to identify all available extension points. Hereafter AdIDoS provides a pool of various XML-based DoS attacks, which can easily be configured through the configuration dialog as shown in Fig. 3 on the left. This is extremely useful if the tester just wants to execute a subset of the supported attacks to save time. Every attack has its own set of supported parameters. Figure 3 shows the attack parameters for the *Coercive Parsing Attack* on the right. *Coercive Parsing* uses two parameters:

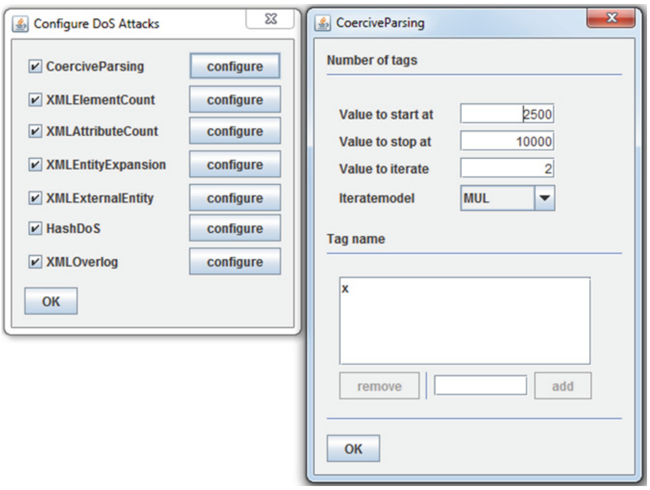
– *Number of tags*: For this parameter a range of values can be specified. In addition, the step size can be set.
– *Tag name*: This parameter can be specified as a list of values.

The range option allows one to perform attacks with various levels of aggressiveness.

5.3 Attack Success and Efficiency Decision

The success of an attack is calculated as follows: We use the median round trip time of untampered requests in comparison to the median round trip time of tampered requests. To compute the median, we use the last ten (untampered or tampered) requests sent to the web service. If the median round trip time of the tampered requests is three times higher[5] than the median round trip time of the untampered requests, the attack is marked as successful. It allows one to reliably recognize attacks as successful and minimizes the false positives.[6] The attack success is recognized as follows:

[4] Areas in the XML document, where additional elements or attributes can be placed according to the schema definition. Identified by `<xs:any>` and `<xs:anyAttribute>` in the XML Schema.
[5] This value was chosen empirically based on our tests in local networks.
[6] Here an attack is marked as successful even though is is not.



Configure DoS Attacks:

- ✓ CoerciveParsing
- ✓ XMLElementCount
- ✓ XMLAttributeCount
- ✓ XMLEntityExpansion
- ✓ XMLExternalEntity
- ✓ HashDoS
- ✓ XMLOverlog

CoerciveParsing:

Value to start at	2500
Value to stop at	10000
Value to iterate	2
Iteratemodel	MUL

Tag name: x

Fig. 3. Configuration of Denial-of-Service attacks

– *ratio time <3*: the attack was not successful
– *ratio time >= 3*: the attack was successful

Besides the information that an attack is successful, AdIDoS also provides an estimation of the attack efficiency. Again this estimation is based on the median round trip time of the two attack runs.

– *ratio time between >= 3 and <6*: the attack was efficient
– *ratio time >= 6*: the attack was highly efficient

To avoid false positives, the AdIDoS algorithm uses an approach with a single success confirmation. If the algorithm detects measurable differences between tampered and untampered round trip time, the server first gets some time to recover. This prevents that a DoS attack is marked as successful even though it is not, just because it is executed right after a successful attack. After the recovery time, a new attack vector is sent to the server and its response time is compared to the response times of untampered requests.

5.4 Extended ARTR Approach

Falkenberg et al. [4] presented an algorithm for attack success measurements that uses a blackbox approach with an ARTR metric (see Sect. 2.4). Their ARTR approach was based on measuring response times. The response time measurement always started with the first byte that was sent, and stopped with the last byte that was received. With this algorithm the comparison of two or more requests requires that the requests must have the same size. Otherwise the transmission of the data would affect the measurement.

AdIDoS also pursues a blackbox approach with the ARTR metric. However, it uses a slightly different measurement algorithm (see Fig. 4). The response time measurement starts with the last byte sent, and stops with the first byte received. The main benefit of this improved time measurement is that fluctuations, which can occur during transfer, do not affect the measurement as strongly as before. In addition, only the time is measured that the service needs to execute the request. Finally, it becomes less important to send requests of the same size.

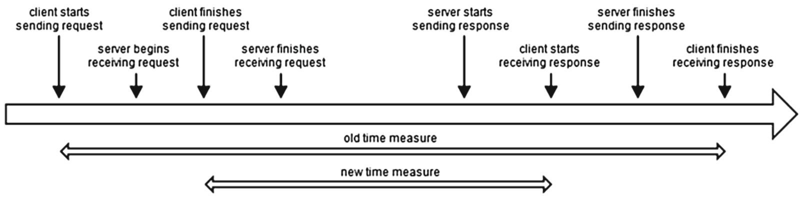

client starts sending request
server begins receiving request
client finishes sending request
server finishes receiving request
server starts sending response
client starts receiving response
server finishes sending response
client finishes receiving response
old time measure
new time measure

Fig. 4. Our new ARTR approach considers only time between the last byte that was sent, and the first byte that was received.

6 Practical Evaluation

Using AdIDoS, it becomes easy to test a given web service for DoS weaknesses. Multiple test scenarios were set up to investigate common web service frameworks: Apache Axis2 [13], Apache CXF [14], Metro [15], .NET [10] and PHP [16].

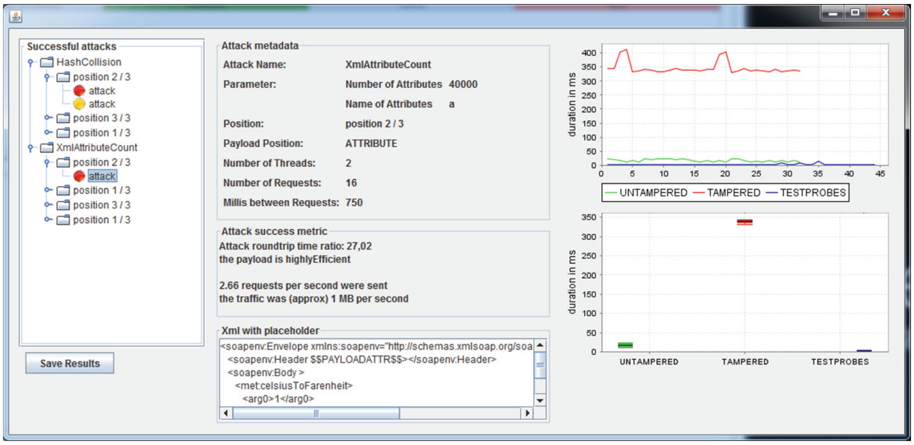
The services were hosted on a Windows 7 machine (@2,30 GHz, 4 GB Ram) with the following set up:

- Java based The services were hosted s: Tomcat 7.0.55 (Oracle Java7 1.7.0_71)
- .NET: IIS 7.5.7600.16385 (.NET framework v2.0.50727)
- PHP: Apache 2.4.12 (PHP 5.5.24.0)

The tests were performed from a second, independent Windows 7 machine within the same LAN with the default configuration and parameters. As a service a simple conversion service was implemented, which converts Fahrenheit to Celsius and vice versa. In addition, the XML Security Gateways WebSphere DataPower Integration Appliance XI50 [6] and Axway SOA Gateway 7.3.1 [1] were tested (Fig. 5).

Table 1 gives an overview of all tested web service. Apache CXF version was the most secure open source The services were hosted service framework. It was the only open source framework that provides a secure default configuration. The CXF implementation limits the possible appearance of elements in an XML document to achieve this goal.

The Apache Axis2 framework is vulnerable to Coercive Parsing, XML Attribute Count and HashCollision with the collision generators DJBX31A and DJBX33A. It is very unusual that one implementation is vulnerable to multiple collision generators, and we cannot explain this behavior. The vulnerability to



Successful attacks
- HashCollision
  - position 2 / 3
    - attack
    - attack
  - position 3 / 3
  - position 1 / 3
- XmlAttributeCount
  - position 2 / 3
    - attack
  - position 1 / 3
  - position 3 / 3
  - position 1 / 3

Save Results

Attack metadata

Attack Name:	XmlAttributeCount
Parameter:	Number of Attributes 40000
	Name of Attributes a
Position:	position 2 / 3
Payload Position:	ATTRIBUTE
Number of Threads:	2
Number of Requests:	16
Millis between Requests:	750

Attack success metric
Attack roundtrip time ratio: 27,02
the payload is highlyEfficient

2.66 requests per second were sent
the traffic was (approx) 1 MB per second

Xml with placeholder
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soa
  <soapenv:Header $$PAYLOADATTR$$></soapenv:Header>
  <soapenv:Body >
    <met:celsiusToFarenheit>
      <arg0>1</arg0>
```

Fig. 5. Automatically generated result view of successful attacks with concrete information.

Table 1. Results of our vulnerability scan. The Symbol ⚡ marks web service, where DoS weakness were found by AdIDoS.

	Apache Axis2	Apache CXF	Metro	.NET	PHP	XI50	Axway
Coercive Parsing	⚡	-	-	-	-	-	-
XML Element Count	-	-	-	-	-	-	-
XML Attribute Count	⚡	-	⚡	⚡	⚡	⚡	-
XML Entity Expansion	-	-	-	-	-	-	-
XML External Entity	-	-	-	-	-	-	-
HashCollision	⚡	-	⚡	-	⚡	-	-
XML Overlong Names	-	-	-	-	-	-	-

Coercive Parsing and XML Attribute Count (on ELEMENT) is limited to the `soap:Header`. This indicates that unexpected elements are only processed at this position. The highest impact comes from XML Attribute Count, only CXF was not vulnerable to this attack.

In contrast to the expected behavior of the two security gateways, the XI50 was also vulnerable to XML Attribute Count. By placing the attack payload within an existing element in the `soap:Body` there was a clear evidence for a higher processing time.

Besides the detection of DoS weaknesses, AdIDoS is able detect thresholds used by the implementations. These thresholds are considered for further investigation of a service. Table 2 shows the detected thresholds and their approximate value.

Table 2. Overview of thresholds used in the tested frameworks.

Threshold for	Apache CXF	XI50	Axway
Nested Elements	80 – 158	470 – 548	236 – 314
Number of Elements	-	-	783 – 1,173
Number of Attributes	626 – 704	-	704 – 782
Element name length	-	3,125 – 3,515	3,906 – 4,296
Attribute length	116,226 – 122,343	-	-
Number of Entities	-	-	16 – 32

Table 3. Average ARTR and attack parameters.

Attack name	Axis2	Metro	.Net	PHP
Coercive Parsing ARTR	6.52			
Number of Tags	2,500			
XML Attribute Count ARTR	4.02	7.00	3.30	10.65
Number of Attributes	10,000	10,000	10,000	10,000
HashCollision ARTR	12.75	6.21		155.88
Number of Collisions	3,750	3,750		1,250

AdIDoS performs multiple attacks against a web service. The impact of an attack is shown by ARTR and the used parameters. Table 3 illustrates the ARTR for the tested web service and Table 4 illustrates the ARTR for the XI50 security gateway. Beside the ARTR the used parameters for the single attacks are specified.

Table 4. Average ARTR and attack parameters for XI50

Attack name	XI50
XML Attribute Count ARTR	7.79
Number of Attributes	2,500

The goal of AdIDoS is to detect DoS weaknesses in XML-based web service and not to exploit them. For this reason, AdIDoS stops as soon as a DoS weakness for an attack class (e.g. Coercive Parsing) is detected. More aggressive attacks, which certainly result in a higher ARTR, are not performed.

7 Related Work

There are already DoS attacks that rely on handling of XML data. These attacks are partially supported by penetration testing tools like SoapUI,[7] or WSFuzzer.[8]

[7] http://www.soapui.org.
[8] http://sourceforge.net/projects/wsfuzzer.

SoapUI and WSFuzzer are tools developed specifically for testing web service platforms, but these tools have no support for automatic XML-based DoS analysis.

Oliveira et al. implemented a web service tool called WSFAggressor [17], which contains several DoS attacks. However, in order to evaluate the attack success, this tool requires access to the tested system. This prerequisite is not given by evaluating specific hardware devices such as IBM Datapower [6], or pentesting sensitive customers' servers. Moreover, this tool misses some important attack techniques such as HashDoS [18].

Falkenberg et al. studied XML-based DoS attacks [4] and implemented a WS-Attacker DoS plugin. The plugin does not need access to the tested web service in order to measure the attack success. It instead uses a blackbox approach using the server response times (ARTR) only. In contrast to AdIDoS, the authors do not analyze an adaptive approach of XML-based DoS testing: Values and size of tampered messages is chosen statically, and the penetration tester has to adapt these properties manually. This results in attack testing complexity and to possible false negatives. In our work, we extended the approach of Falkenberg et al. and implemented an adaptive and intelligent detection XML-based DoS attacks.

Very recently, Pellegrino et al. studied data compression attacks against several applications [11], including web service servers. In order to execute an attack against a web service server, the attacker inserts a huge number of spaces into a SOAP message and compresses the message using a deflate algorithm (used by zlib, gzip or zip libraries). This way, a compression ratio of about 1:1000 can be achieved. The authors reported that Apache Axis2 and Apache CXF were vulnerable to these attacks. These attacks are currently missing in WS-Attacker and can be implemented in a future work.

8 Conclusions and Future Work

In this paper, we developed a new approach for testing robustness of XML-based web service against DoS attacks. Our approach adapts an intelligent strategy that automatically increases the attack strength and searches for attack thresholds. We implemented the approach as a new plugin for the web service penetration testing framework WS-Attacker. Interestingly, the plugin allowed us to detect new attacks, previously overlooked in related works. This proves the feasibility of our new approach for testing DoS attacks.

While our paper investigates SOAP-based web service, the implemented library can be directly applied to further XML standards as well, e.g. SAML or REST-based web service. Moreover, the general idea of intelligent DoS testing can be adapted to other applications beyond XML as well.

Further research in this direction could be in extending the number of web service specific attacks. As described in [4,11], further attacks like Recursive Cryptography, XML Signature Key Retrieval DoS, or data compression attacks are applicable to web service as well.

The values for detection of attack success and efficiency were chosen empirically based on our observations in local networks. However, different network conditions could affect the results and introduce new false positives and false negatives. In order to detect DoS attacks over the Internet, the accuracy of our solution has to be improved.

Acknowledgements. We would like to thank our anonymous reviewers for their helpful comments. The research was supported by the *German Ministry of research and Education (BMBF)* as part of the `VERTRAG` research project.

References

1. Axway: Axway SOA gateway. https://www.axway.com/products-solutions/soa-governance/soa-gateway
2. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible markup language (xml) 1.0) (5th edn.), November 2008. http://www.w3.org/TR/REC-xml/
3. Mainka, C.: Automatic Penetration Test Tool for Detection of XML Signature Wrapping Attacks in Web Services, Master thesis supervised by Jörg Schwenk and Juraj Somorovsky, May 2012
4. Falkenberg, A., Mainka, C., Somorovsky, J., Schwenk, J.: A new approach towards DoS penetration testing on web services. In: IEEE 20th International Conference on Web Services (ICWS), 2013, pp. 491–498. IEEE (2013). http://dblp.uni-trier.de/db/conf/icws/icws2013.html#FalkenbergMSS13
5. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. **2**(2), 115–150 (2002). http://doi.acm.org/10.1145/514183.514185
6. IBM: websphere datapower integration appliance xi50. https://www-03.ibm.com/software/products/en/datapower-xi50
7. Kupser, D., Mainka, C., Somorovsky, J., Schwenk, J.: How to break XML encryption – automatically. In: 9th USENIX Workshop on Offensive Technologies (WOOT 15). USENIX Association, Washington, D.C., August 2015. https://www.usenix.org/conference/woot15/workshop-program/presentation/kupser
8. Mainka, C., Somorovsky, J., Schwenk, J.: Penetration testing tool for web services security. In: SERVICES Workshop on Security and Privacy Engineering, June 2012
9. McCabe, F., Booth, D., Ferris, C., Orchard, D., Champion, M., Newcomer, E., Haas, H.: Web services architecture. W3C note, W3C, February 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
10. Microsoft: .net framework. https://msdn.microsoft.com/en-us/library/a4t23ktk(v=vs.80).aspx
11. Pellegrino, G., Balzarotti, D., Winter, S., Suri, N.: In the compression hornet's nest: A security study of data compression in network services. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 801–816. USENIX Association, Washington, D.C., August 2015. http://blogs.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pellegrino
12. Sperberg-McQueen, C.M., Thompson, H.S., Maloney, M., Thompson, H.S., Beech, D., Mendelsohn, N., Gao, S.S.: W3C XML schema definition language (XSD) 1.1 part 1: Structures. Last call WD, W3C, December 2009. http://www.w3.org/TR/2009/WD-xmlschema11-1-20091203/

13. The Apache Software Foundation: Apache axis2. https://axis.apache.org/axis2/java/core/
14. The Apache Software Foundation: Apache CXF - index. https://cxf.apache.org/
15. The GlassFish community: Metro. https://cxf.apache.org/
16. The PHP Group: Php: Hypertext preprocessor. https://php.net
17. Vieira, M., Laranjeiro, N., Oliveira, R.A.: Experimental Evaluation of Web Service Frameworks in the Presence of Security Attacks, June 2012
18. Wälde, J., Klink, A.: Hash Collision DOS Attacks. 28C3, December 2011. http://www.nruns.com/_downloads/advisory28122011.pdf

Reputation, Monetization and Data Privacy Management

An Integrated Reward and Reputation Mechanism for MCS Preserving Users' Privacy

Cristian Tanas, Sergi Delgado-Segura, and Jordi Herrera-Joancomartí(✉)

Departament D'Enginyeria de la Informació I Les Comunicacions, Universitat Autònoma de Barcelona, Barcelona, Spain
{ctanas,sdelgado}@deic.uab.cat, jordi.herrera@uab.cat

Abstract. Mobile Crowd Sensing (MCS) presents numerous and unique research challenges most of them based on the fact that human participation is in the loop. In this paper we analyse three of the most important: user participation, data sensing quality and user anonymity. To solve them, we present PaySense, a general framework for user rewarding and reputation accountability that preserves users' privacy using cryptocurrencies. Furthermore, we detailed an implementable system using bitcoins.

1 Introduction

Mobile Crowd Sensing (MCS) arises as a new sensing paradigm based on the power of the crowd jointly with the sensing capabilities of smartphones. The increasing popularity of such devices paired with the inherent mobility of their owners enables the ability to acquire local knowledge from the individual's surrounding environment. This local knowledge ranges from location information to more specialized data such as pollution levels going through a longer list of personal and surrounding context, noise levels or traffic awareness among others.

A large number of crowd sensing applications have already been developed, although typically for experimental purposes and to show the usefulness of such a sensing paradigm. For instance, BikeNet [9] harnesses the sensing capabilities of smartphones paired with the individual's *smartness* to determine the most "bikeable" routes in a city. Similarly, the Common Sense project [8] allows individuals to measure their personal exposure to air pollution and share it with their social sphere. On the other hand, applications such Nericell [17] measure the interaction between individuals to infer the context within which they carry out their activities. In this particular case, traffic congestion.

However, MCS presents numerous and unique research challenges most of them based on the fact that human participation is in the loop and range from participatory and opportunistic data collection, proper incentive mechanisms, transient network communication and big data processing. Nonetheless, human participation raises singular issues regarding the privacy and security of data, as sensitive information such as human voice or location may be revealed. Furthermore, the quality and trustworthiness of the contributed data (e.g. counterfeit data contributed by malicious users) should also be addressed.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 83–99, 2016.
DOI: 10.1007/978-3-319-29883-2_6

In this paper we present PaySense, a practical and integrated system that, using the *Bitcoin* crypto currency, provides a general framework for user rewarding and reputation accountability while preserving the users' privacy.

The rest of the paper is organized as follows. In Sect. 2 we summarize the main challenges in MCS and review the most relevant existing proposals appeared so far in the literature. Section 3 introduces the main blockchain based cryptocurrency properties and how they can be applied for user rewarding and reputation accountability. In Sect. 4 we describe our proposal based on bitcoins. Finally, Sect. 5 concludes the paper and gives some guidelines for further research.

2 State of the Art

The three main challenges of MCS addressed in this paper have been analyzed and the main proposals appeared in the literature are summarized in this section.

2.1 User Participation

MCS systems typically involve a very large number of users or *crowd sensors* in the sensing task by collecting and sending local data obtained through their sensor-enabled mobile devices to a data collection center. The performance and usefulness of such sensor networks heavily depends on the crowd sensor's willingness to participate in the data collection process. Therefore, incentive mechanisms are of utmost importance in MCS scenarios to engage as many crowd sensors and provide to the data collection center with a considerable wealth of data.

Based on the nature of user participation, we can discern between two MCS paradigms, as introduced by Lane *et al.* [14]: *participatory* sensing and *opportunistic* sensing. The first sensing paradigm requires the user to have complete conciousness of what, where and when is being sensed. For instance, it may require the users to observe and describe their surrounding environment, typically assuming a higher degree of involvement for the crowd sensors. On the other hand, in opportunistic sensing scenarios, the data is acquired in the background, namely the data is being sensed opportunistically and automatically sent (i.e. without the user's active participation) using the device's network connection to the data collection center. Even though it may seem that users would be more willing to participate in this kind of scenarios, battery waste or large amounts of data being sent may cause the user to refuse to participate in sensing tasks.

The design of incentive mechanisms to stimulate participation has been addressed in crowd sensing scenarios [23] although similar needs where previously identified in the field of ad hoc or P2P networks [24], which also relay on the participant's willingness, in that case to forward packets. The nature of the incentives provided to crowd sensors allows its classification on the following categories: *economic*, *service-based* and *social.*

The incentives mechanisms in the first category model the problem through a financial approach where crowd sensors get paid or received some kind of credit based on the provided service [12]. However, the lack, until now, of an easy to use, cheap and secure micro-payment system, pushes towards dealing with external entities such as banks or financial institutions that impose too high transaction fees for a practical pay-per-sense solution.

On the other hand, service-based incentives try to foster participation by taking a user-centric approach in which feedback or individual benefits are perceived by the crowd sensors in a way that is relevant to them [5]. However, not all sensing scenarios may fall into such categories since the sensing objective may not be of interest or provide no benefits to the crowd sensors.

Finally, gamification techniques have been used to incentive participation in a wide range of scenarios [22] in what we consider a "social" rewarding. However, apparently the effectiveness of such incentive schemes depends considerably on a large set of interrelated factors from community related (topic, number of participants, etc.) to cultural or motivational factors [7]. This multitude of determinants makes it difficult to assess if the gamification scheme would work in all given MCS scenarios.

2.2 Data Sensing Quality

In MCS systems there is no control over the crowd sensors and we cannot assume that all individuals will behave in the exact same manner or will be equally honestly. Therefore, the overall quality of the sensor readings can see itself deteriorated if counterfeit data is received from malicious users. Hence, the obvious question is how to validate the sensing data that crowd sensors provide to the system. A commonly used approach is to validate the data depending on the trust level of the crowd sensor that reports it [18].

Trust and reputation systems have long been studied to establish trust relations among the members of an on-line community where prior knowledge of the participants may not even be available or where the community is formed by a crowd of anonymous volunteers. In such systems, each user is provided with a reputation score that indicates his or her trustworthiness when acting as information provider. For instance, in [11], Jøsang *et alter* introduce several reputation quantification models while [1] presents a reputation framework based on fuzzy logic in the context of social participatory sensing. Such reputation score can be increased each time a crowd sensor provides valid data.

Nevertheless, a validation scheme based only on the user's reputation as a sensing data reliability measure does not provide a useful method since, in an initial state, the system could not identify any crowd sensor as trustworthy, leading it towards a deadlock state. In order to avoid such situations, combination of reputation systems and collective knowledge can be applied [19].

Data Quality vs Incentive Mechanisms. Combining both reputation systems and incentive mechanisms in the context of a MCS application is especially sensitive. Given the self-interested and possibly selfish nature of individuals, there exists the possibility of crowd sensors acting in a way to maximize their own gains, regardless of the impact that this may have on the overall sensor network. With higher incentives provided for participation, more motivated will be the crowd sensors to increase their personal benefits possibly by providing misleading information. Therefore, improving the design of an incentive mechanism should imply an improvement of the corresponding quality control process and hence, of the reputation system.

2.3 User Anonymity

An important aspect of MCS scenarios is the collection of potentially sensitive information pertaining to individuals. For instance, GPS sensor readings can be used to track users movements and profile them for other purposes besides their crowd sensing tasks. Furthermore, if the MCS application collects "delicate" information such as criminal acts, crowd sensors may be reluctant to provide data without proper anonymity measures for the fear to be collaterally involved in such acts. Hence, it is necessary to preserve the privacy of crowd sensors, but at the same time ensure the usefulness of the MCS application.

A popular approach for preserving users privacy is *anonymization*, which removes any user identifying attribute from the sensing data before sending it to the data collection center. Obviously, this approach can be applied only in those MCS applications that are solely interested in the actual data and no further interaction with the crowd sensors is necessary. Another approach is to use pseudonyms when sending sensing data to the data collection center. These pseudonyms are typically randomly generated and bare no relation with the individual's real identity. In this cases we say that individuals benefit from pseudo-anonymity since we cannot infer their real identity from the pseudonyms, but we can still identify subsequent sensor readings as reported by the same user with the same pseudonym.

Anonymity vs Incentive Mechanisms. Although incentives mechanisms and anonymity have been addressed separately in crowd sensing scenarios, it is still an open problem to address them simultaneously. Crowd sensor should be able to provide sensing data in an anonymous way while still perceiving incentives for that task and at the same time crowd sensor network authorities should be able to ensure that dishonest users cannot earn unlimited credit.

In a recent work, [15], Li and Cao propose an incentive scheme where users are rewarded for their contributions with tokens (credits) that can later be exchanged for additional services or for real-world objects. The incentive scheme relies on a trusted third party to ensure that the anonymity of the users is preserved by applying blind signatures and commitment techniques.

Anonymity vs Reputation Systems. An anonymous reputation system may seem an apparent paradox considering that an anonymous system requires complete unlinkability between the user's identity and the sensed data, while reputation systems claim this link to be existent in order to maintain an updated reputation score for each user.

In [13] the authors acknowledged the importance of a privacy-preserving reputation system. While rarely explored in the context of MCS networks, this has been addressed in peer-to-peer networks, where users are allowed to create multiple pseudonyms, each with its own reputation score, to achieve anonymity [2]. Then, different pseudonyms may be used in different interactions with other entities, forcing adversaries to trace sequences of pseudonyms used by the same individual in order to reveal his or her identity.

Having different pseudonyms, each with its own reputation score, however, is detrimental to the reputation system which should be continuous and applicable transversely to all pseudonyms corresponding to the same user. Miranda and Rodrigues develop this idea in [16] and provide a mechanism that allows users to transfer the reputation information from one pseudonym to another, without disclosing this link or the user's real identity.

However, if not properly performed, reputation transfer between pseudonyms may provide a linkage mechanism. Take for example an individual having the highest reputation score among all pseudonyms. Then, it would be straightforward for an adversary to establish a link between this pseudonym and the pseudonym conserving the same reputation score after the transfer process. Christin *et alter* [6] discuss this issue and provide a solution in the context of participatory sensing. They also acknowledge the need of a trusted third party to ensure that anonymity is preserved in such scenarios. However, the main drawback of their proposal is that users lost reputation in favour of pseudonym unlinkability, but the lost reputation could not be recovered.

Finally, the possibility of changing pseudonyms may allow a Sybil attack, where malicious users may replicate sensor readings under different pseudonyms to earn more reputation or credit, so new measures have to be developed to enforce that every crowd sensor only has a valid pseudonym at a given time.

3 Blockchain Based Cryptocurrencies

The *blockchain* concept was first applied in early 2009 as a core part of Bitcoin and since then it has been adopted by other cryptocurrencies like litecoin or dogecoin. The *blockchain* is a distributed database formed by chained blocks designed to avoid their tampering once they are published. Such distributed approach has different sides: data storage, data transmission, and data confirmation. Data storage is achieved by means of replicating the blockchain information, data transmission uses a P2P network and data confirmation is performed by a competitive open an transparent process, called mining. The main application of the *blockchain* is in the field of cryptocurrencies where the blockchain represents the public non-modifiable ledger of all system transactions that prevents the double-spending problem.

In a blockchain based cryptocurrency, *coins* are not digital objects but an accounting entry in an account. Each account is identified by its address which is tied to a public key pair. Payments in the system are performed through transactions which indicate the source address (the payer) and the destination address (the payee) of the payment. The payment destination can be determined only with the address, a value publicly known, while the source of the payment is validated through a digital signature performed using the private key, a secret value only known by the owner of the source address. Transactions are only valid once they are added to the blockchain through the mining process, a distributed process that can be performed by any user of the network. Mining blocks helps to confirm performed transactions and it has been designed to be a hard task. Using the concept of proof-of-work in order to provide a significant level of security to the cryptocurrency network, the effort of validating blocks is rewarded, mainly, with new coins that are constantly created.

On regards to anonymity, cryprocurrency systems achieves such property by allowing users to create any number of anonymous addresses that will be used in their transactions.

3.1 Cryptocurrencies as a Rewarding Mechanism

Using blockchain based cryptocurrencies as a rewarding system is a straight forward method to reward users in a MCS scenario. For each sensing value crowd sensors provide, they receive a payment as an awarded reward. Users can generate their addresses that will be used as rewarding addresses where the payer, probably the data collection center, using its own address, will send the payments for the sensing task the users perform.

Furthermore, when users' privacy has to be taken into account, all mechanisms used in the system have to provide a certain degree of anonymity. In this case, rewarding users using a blockchain based cryptocurrency is a good strategy due to the anonymity level provided by the majority of such cryptocurrencies. As we already mention, cryptocurrencies anonymity is based on the easiness of anonymous addresses generation that will be used for the payments. For that reason, crowd sensors providing sensing data may generate multiple addresses and the rewarding amount a user has to receive for all his reported senses could be spread over different addresses used for the rewarding payment.

3.2 Cryptocurrencies as a Reputation Annotation Mechanism

As we already mention above, reputation measures may be used to assess the quality of the information that users send to the data collection center. Sensing data may be accepted or discarded based on the reputation value of the user reporting such data. Our approach is to adopt a cryptocurrency coins also as a reputation annotation system, tying the concepts of reward and reputation. Such approach could seem a limitation since the unification of both concepts in a unique value implies that the reward system determines the reputation score and conversely. Nevertheless, from a conceptual point of view, reward and reputation concepts are closely related. Notice that, reward for a given sensed value can be seen as a measure of correctness of such value, since reward should depend on the usefulness of that value. Following such approach, the balance in a specific address will represent both the total awarded amount for the sensing tasks and the reputation obtained for the tasks.

Representing both concepts, reward and reputation, as the balance of an address has an interesting implication. For an address used with this purpose (reward and reputation), each further payment represents a withdrawal of economic funds since coins are transferred to another address, but also implies a reputation reduction. Conversely, the address users may desire to reduce their reputation in exchange for receiving some benefit, in that case the associated withdrawal would be such benefit. At a first glance, it may not seem obvious the need for reputation reduction in a MCS scenario, but as we point out in Sect. 4.3, anonymous reputation schemes using multiple pseudonyms need to reduce users' reputation in order to provide unlinkability of pseudonyms.

4 PaySense: An Integrated Privacy Preserving Solution for Reward and Reputation

In this section we present PaySense, a cryptocurrency based system that provides an integrated mechanism for reward and reputation in MCS applications while preserving crowd sensors' anonymity. Any blockchain based cryptocurrency allowing anonymous address generation can be used in PaySense, but we will focus our practical implementation in bitcoin[1] since it is the most used and stable system nowadays.

4.1 PaySense Entities

The PaySense system is composed by the following entities:

Crowd sensors (CS). Each crowd sensor, CS_i, collects data from her surrounding environment and sends it to the data collection server. They are identified in the PaySense system through multiple bitcoin certified addresses[2], that is $Addr_i^j$ for $j = 1, \cdots, n$. Each bitcoin address has an associated elliptic curve digital signature algorithm (ECDSA) key pair, (Pk_i^j, Sk_i^j), and is constructed from the public portion of such key pair as follows: $Addr_i^j = f(Pk_i^j)$ for a public known cryptographic hash function $f(\cdot)$. Since bitcoin addresses basically result in random alphanumeric characters, crowd sensors can use them as pseudonyms when communicating with other entities in order to preserve their anonymity. For that reason, we use both notation, namely bitcoin address and crowd sensor pseudonym, indistinctly. Furthermore, the use of bitcoin addresses allows crowd sensors to transact with other PaySense entities or even other bitcoin users.

Address Certification Authority (ACA). Each bitcoin address $Addr_i^j$ for $j = 1, \cdots, n$ owned by a crowd sensor CS_i must be certified by the Address Certification Authority (ACA) in order to provide some degree of control over all existing users of the Bitcoin network and avoid a Sybil attack. The Bitcoin address certificates, $Cert(Addr_i^j)$, are issued following the *X.509v3* standard and

[1] The interested reader can refer to [20] for detailed information on the bitcoin system.

[2] Note that the concept of certified address does not exist in the bitcoin system, but a characteristic of PaySense as described in Sect. 4.2.

with the ACA acting as issuer certification authority. The goal of the certification process is twofold: on one hand, it ensures that S_i does not use different bitcoin address in order to change identities and on the other hand, it ensures that bitcoin addresses are renewed periodically given the validity of the certificate, which actually limits the validity of the bitcoin address itself[3].

Data Collection Server (DCS). This entity represents the MCS application server in charge of receiving and processing sensing data sent by the crowd sensors. The DCS must perform a validation process on the received data and based on its correctness, or a data reliability measure, gives rewards to the crowd sensors providing such data. Rewards are provided through bitcoin transaction and for that purpose, the DCS also holds a publicly know bitcoin address, $Addr_{DCS}$ and an ECDSA key pair associated with it. Such address is publicly known by all the crowd sensors and is used exclusively to reward crowd sensors for their readings through the Bitcoin network.

The Bitcoin P2P Network. Although not a real entity of the PaySense architecture, the bitcoin network is used to transfer bitcoins between PaySense certified bitcoin addresses. Furthermore, the transaction information stored in the blockchain will be further used to assess the correctness of the PaySense system, as we discuss further on (*e.g.* crowd sensor reputation query).

4.2 PaySense Interaction Model

In this section we describe the different interactions between PaySense entities that are performed to execute all processes involved in our system.

Crowd Sensor Enrollment. When a crowd sensor CS_i wants to join for the first time the PaySense system, she must request a bitcoin address certificate from the ACA by sending a certificate signing request $CSR(Addr_i^j)$. This certificate signing requests contains the address $Addr_i^j$ in the subject common name field. The real identity of the crowd sensor is required at this step in order to validate that she did not ask for another bitcoin address certificate in the past trying to make a "fresh start" and preventing a Sybil attack. Then, the ACA verifies that the bitcoin address has a *zero-balance* (i.e. does not contain any bitcoins) to ensure that CS_i does not enter the system with a previously assigned reputation score, and issues a new bitcoin address certificate $Cert(Addr_i^j)$. Note that such certificate guarantees that bitcoin payments are being performed only to bitcoin addresses owned by registered crowd sensors.

However, it is straightforward to notice that if CS_i sends her $CSR(Addr_i^j)$ directly, then the ACA could link $Addr_i^j$ to CS_i's real identity. In order to avoid this from happening, we adopt a blind signature scheme as follows:

[3] Although bitcoin addresses do not expire, we apply the concept of expiration to the certificate issued by the ACA. When an address "expires" it cannot be used in the MCS system but it is still a valid and usable bitcoin standard address.

1. CS_i generates n different bitcoin addresses ($j = 1, \cdots, n$) and for each one computes the certificate signing request, $CSR(Addr_i^j)$ and obtains its hash value, $h(CSR(Addr_i^j))$.
2. CS_i blinds[4] each of the n hash values obtained in Step 1, $b_i^j = Blind(h(CSR(Addr_i^j)))$ for $j = 1, \cdots, n$, and sends the n hashed values to ACA together with CS_i real identity.
3. The ACA randomly selects one of the received blinded hash values, namely b_i^k, and requests CS_i both, the unblind factor and the $CSR(Addr_i^j)$ for the rest of the values $j = 1, \cdots, k-1, k+1, \cdots, n$. Then the ACA extracts the $n-1$ bitcoin addresses contained in the received CSR and verifies that all of them has zero balance. Then, the ACA uses the unblinding factors to unblind each b_i^j for $j = 1, \cdots, k-1, k+1, \cdots, n$ and checks that the unblinded values matchs with the hash values of the received CSR values.
4. If all verifications performed by the ACA in Step 3 hold, the ACA signs the binded value b_i^k and sends the result $Sig_{Sk_{ACA}}(b_i^k)$ to CS_i.
5. Upon reception, CS_i unblinds the digital signature $Sig_{Sk_{ACA}}(b_i^k)$ performed by the ACA and uses the unblinded result together with the original $CSR(Addr_i^k)$ value to create the certificate: $Cert(Addr_i^j) = Sig_{Sk_{ACA}}(Addr_i^k)$

Notice that although the ACA is signing a blind certificate, the cut-and-choose technique included in Step 3 ensures that a dishonest crowd sensor cannot obtain an arbitrary signature (for instance with a bitcoin address with non-zero balance) from the ACA with a better probability than $\frac{1}{n}$.

Sensing and Reporting Data. Once in possession of a certified address, crowd sensors can begin to report sensed data to the DCS. Prior to its transmission, the data is digitally signed using the crowd sensor's secret key, that is $sig_{data} = Sig_{Sk_i^j}(data)$. Then, each crowd sensor constructs the following sensing report: $Report = \{data, sig_{data}, Cert(Addr_i^j)\}$

Reports are sent over conventional communication networks to the DCS. Note that the sensed information could even pass through a multi-hop network, if needed, since the source of such information could still be identified by the digital signature.

Verification and Validation of Sensed Data. For each sensor reading received, the DCS performs the following validations:

1. The DCS verifies that the data was sent by a registered crowd sensor. For that purpose, the DCS verifies the correctness of $Cert(Addr_i^j)$.
2. The DCS also validates the source authenticity of the data by verifying the digital signature included in the report, that is $Ver_{Pk_i^j}(sig_{data}) = data$.
3. The DCS should apply a validation process on the data itself to assess its quality.

[4] The binding factor depends on the selected digital signature. Although it has been represented as a function for clarity, the blinding factor is a specific an unrelated value for each different CSR to blind.

If all validations are correct, the DCS provides a job reward and proceeds to update the reputation of the crowd sensor.

Notice that the data validation process performed is obviously application dependent. However, we assume that the reputation value of the crowd sensors who sends the data will involved in the validation process (i.e. crowd sensors with higher reputation are supposed to provide more accurate data). Such information is publicly stored in the bitcoin network and the DCS can perform a crowd sensor reputation query (see next interaction) to determine the level of trustworthiness of the crowd sensor and consequently, the correctness of the contributed data.

Crowd Sensor Reputation Query. The reputation value of a crowd sensor is equivalent to her registered bitcoin address balance. That is to say that the sum of all unspent transactions belonging to a particular bitcoin address is equivalent to the reputation score of the owner of such address. Furthermore, this information is publicly available in the blockchain database of the bitcoin network. However, the DCS not only queries the bitcoin address balance, but also validates that all incoming unspent transactions of $Addr_i^j$ come either from the DCS itself as previous rewards, or from another certified bitcoin address due to a reputation transfer process (see Sect. 4.3).

Job Rewarding and Reputation Update. Once the sensed data has been validated the user has to be rewarded and her reputation updated. Since in PaySense both values are tied as a bitcoin payment, the reputation update will determine the rewarding value. Again, the specific MCS application will define its suitable reputation model which will determine the increasing amount of reputation score when users act properly in the system and provide correct readings. Once the increase on the reputation value has been established, the DCS transfer such reputation to the user by performing a standard bitcoin payment to the reporting address. The value of the payment will be the exact amount of reputation increment.

Withdrawal of Rewarded Coins. At any time, crowd sensors can withdraw the bitcoins received as payments from the DCS since there is no difference between a certified bitcoin address and a standard one, from a transactional perspective. Nonetheless, it is important to stress that, according to the PaySense model of interaction, a withdrawal of funds implies a reduction in the reputation value associated with that bitcoin address.

Transferring Reputation to a New Address. Before the bitcoin address certificate expires, the crowd sensor has to obtain a new certificate from the ACA. Such certification renewal is specially sensitive because the bitcoin address that is certified acts as a crowd sensor pseudonym. So in the process of transferring reputation to a new address, in order to provide a privacy preserving mechanism, different constrains have to be meet:

- the ACA should not learn about the CS identity.
- the ACA should not link the old and the new address of a particular CS.
- the CS should not be able to increase her reputation.

PaySense deals with those constrains by using a bitcoin mixing process, described in the next section.

4.3 Transfer Reputation Protocol

Since PaySense uses bitcoin addresses to store the reputation value for each CS, the reputation transfer is performed through a payment between bitcoin addresses. However, due to the openness of the bitcoin blockchain ledger, standard bitcoin payments disclose both the sender and the receiver of a payment and, according to the constrains enumerated above, they are not suitable for a privacy-preserving reputation transfer.

In order to enhance the anonymity properties of the bitcoin transactions and use them for reputation transfer, we propose the use of mix services, a procedure that shuffles the information in order to hinder the relation between the input and the output values of a transaction. The goal of bitcoin mixing is to allow bitcoin users to send bitcoins from one address to a mix service and receive from the mix service the bitcoins to another address that could not be linked with the original one. Different mixing techniques have been proposed in the recent literature ([3,4,10]) each of one presenting different properties. For implementing PaySense we choose the CoinJoin approach [10] since, although its main drawback is the need of a central server taking part in the mixing protocol, in our scenario, the ACA has to take and active part in the reputation transfer, since the she has to issue new certificates, so she can play the role of such a central server needed in the ConJoin protocol.

The main idea of the PaySense transfer reputation protocol is to build a transaction, that we call *reputation transfer transaction*, where multiple CSs will transfer their reputations jointly from the old bitcoin addresses to their new ones hindering the link between input and output addresses. The greater the number of CSs involved in the protocol, the higher the anonymity reached.

CoinJoin mixing protocol takes advantage of the low level transaction structure in order to create our *reputation transfer transaction* using a multiparty protocol where different actors do not obtain any knowledge that could reveal the link between input and output addresses of such transaction. In order to understand the proposed protocol for the reputation transfer, we have to review some details of the bitcoin transaction structure.

A bitcoin transaction is formed by two basic parts, the inputs block, and the outputs block (see Fig. 1).[5]

When a user wants to perform a bitcoin payment, he should build a transaction including a set of non-spent previous transaction in the inputs block. The total bitcoin amount of the inputs should be at least the amount to be spent. In addition, a list of outputs should be placed in the outputs block, representing the bitcoin addresses to which the user want to pay, and the amount of bitcoins to be paid. The transaction inputs block represents the budget that the user

[5] In a simplest bitcoin transaction, the inputs block and the outputs bock contain exactly one single address.

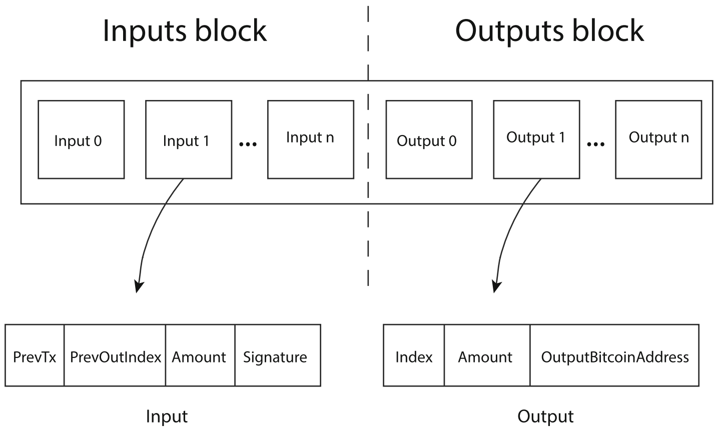


Fig. 1. High level representation of a bitcoin transaction.

has to spend, and reference the previous transactions where the money comes from, and the outputs block represents how the money is spent. Once the inputs and outputs addresses of the transaction are selected the total transaction has to be signed. Standard bitcoin transaction signature is performed by signing the transaction (inputs and outputs) with each private key related to the bitcoin addresses where the funds come from, in other words, from each private key of the bitcoin addresses referenced in the input block. For each input address, the signature is computed over the same transaction value (inputs and outputs). Once all signatures are performed, the resulted signed value will be placed in the corresponding field of each one of the inputs of the inputs block, as it is depicted in Fig. 1.

Notice that with this procedure no link could be established between transaction inputs and outputs, since each signature is performed over the same base information. Furthermore, the base information to be signed (inputs and outputs) has to be build before any transaction signature is performed.

Once the basic bitcoin transaction creation procedure is described we can provide a high level description of our reputation transfer protocol. In such a protocol, different CS create a *reputation transfer transaction* with the help of the ACA. First of all, every CS indicates her bitcoin PaySense certified address as input address and a new bitcoin address as the output. Such new address will be the new certified address for the new period. All this information will be sent to the ACA who would build the base information transaction, the data that each CS has to sign in order to validate the transaction. Then the ACA sends the base information transaction to the CSs and each of them performs the corresponding signature and returns the value to the ACA. Finally, the ACA builds the *reputation transfer transaction* with the base information transaction that he has created plus every signature received from the CSs and sends it to the bitcoin network in order to include it into the blockchain. Then, the ACA certifies all output addresses included in the *reputation transfer transaction*

In Fig. 2, the transfer reputation protocol is depicted. To avoid the disclosure of the link between the inputs and the outputs of the *reputation transfer transaction*, the ACA performs the data collection in three different and independent stages, recollecting different data in each one of them. Moreover, the ACA is accessible to collect data from the CS only through the Tor network [21], being the web application offered by the ACA built as a Tor hidden service.

Transfer Reputation Protocol:

Stage 1: *Output recollection.* The first stage consists in recollecting the outputs of the mixing transaction.

1. The ACA advertises a hidden service for transferring a predetermined and fixed reputation value R between certified ACA addresses and new ones.
2. Crowd sensors with a certified bitcoin address with balance equal or greater than the fixed R may join the protocol by sending the new address that has to be certified as an output transaction with the exact value R. Notice that only one output per crowd sensor is allowed.
3. The ACA ends such output recollection of the first stage after a predefined time. The ACA discards any output with a value different than R.

Stage 2: *Input recollection.* The second stage consists in recollecting all the inputs of the *reputation transfer transaction.*

1. Crowd sensors that have already sent the output in the previous stage send now the input with the ACA certified address. Again, only a single input for each crowd sensor is allowed.
2. *(optional)* In case that the certified bitcoin address of the crowd sensor has balance greater than the fixed value R, the crowd sensor avoids the lost of reputation (and money) by performing a standard bitcoin transaction between the certified address and a new one with the exact R amount. Such new address is the one that will be sent as input transaction.
3. The ACA validates that:
   (a) All input amounts are exactly R.
   (b) The total amount of the input address R comes from either a DCS payment or from a valid ACA certified address (in case of the previous optional step). In this second case, the ACA verifies that the R amount from the certified address is also provided from payments of the DCS.
4. The ACA, using the inputs and outputs received in the first and second stages, constructs the base non-signed *reputation transfer transaction.*
5. The ACA ends the input recollection after a predefined time.

Stage 3: *Signature recollection.* In the third stage, the *reputation transfer transaction* is distributively signed.

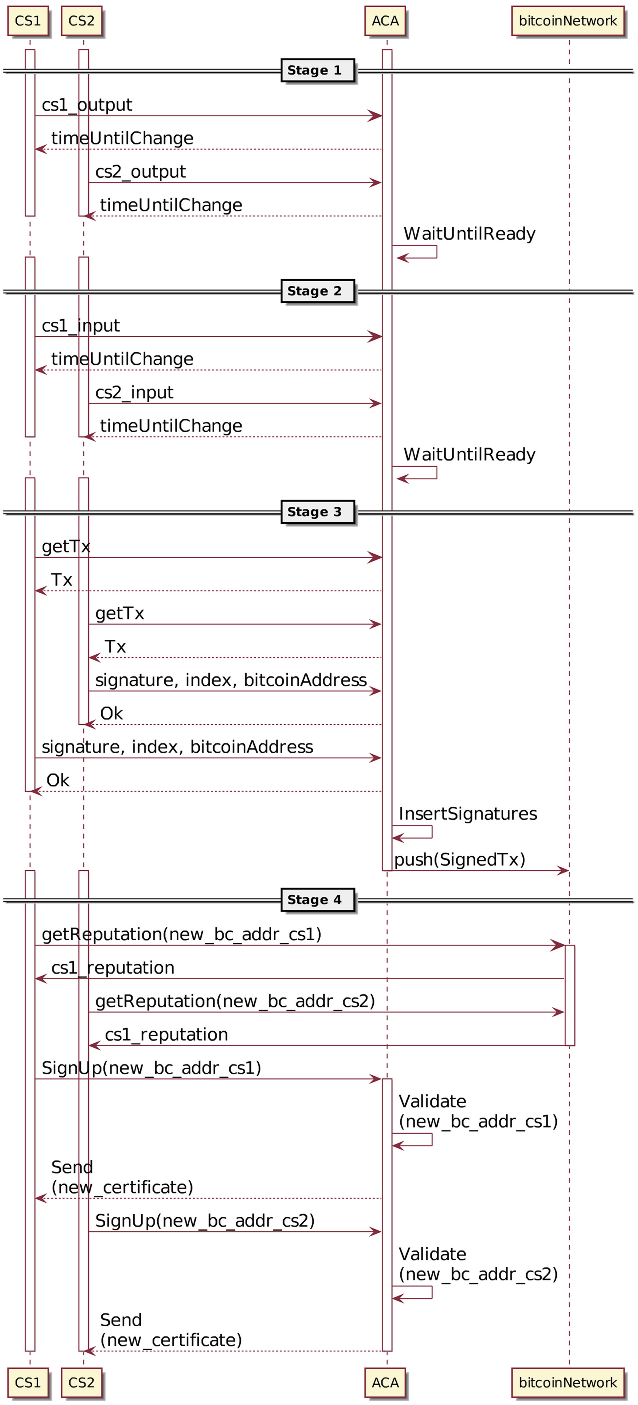


Fig. 2. Transfer reputation protocol with two crowd sensors.

1. Crowd sensors request the base non-signed *reputation transfer transaction* to the ACA.
2. Crowd sensors sign the obtained value and sent the result to the ACA.
3. The ACA compose the *reputation transfer transaction* with the based non-signed transaction that he build in the second Stage and each of the signatures received from each crow sensor.
4. Finally, the ACA pushes the resulting *reputation transfer transaction* to the bitcoin P2P network in order to be included in the blockchain.

Stage 4: *Address certification.* In the last stage, the ACA generates and publishes the new certificates corresponding to the output addresses of the *reputation transfer transaction.*

1. Crowd sensors validate that the *reputation transfer transaction* has been published in the blockchain and that his reputation has effectively been transferred to the new bitcoin address.
2. Crowd sensors send to the ACA the public key corresponding to the bitcoin address where the reputation has been transferred.
3. The ACA verifies that the received public key match one of the bitcoin addresses included as output in the *reputation transfer transaction*, and that the address has only one payment related to it. If the validation is correct sends the new certificate to the crowd sensor.

5 Conclusion and Further Research

With MCS arising as a new sensing paradigm, new singular research challenges are introduced such as fostering participation among the crowd sensors, ensuring the trustworthiness of the contributed data since counterfeit information may be contributed by malicious individuals and, at the same time, preserving the anonymity of the crowd sensors.

We have presented PaySense, a framework that addresses in a practical way all the aforementioned challenges together, using the bitcoin network as an integrating solution. Crowd sensors contribute sensed data using certified bitcoin addresses and reputation and rewards are mapped to a single bitcoin funds transfer, which can later be spent by their owners. Using such approach our system inherits the privacy-preserving properties that bitcoins present. As a main contribution, our proposal solves satisfactory the problem of reputation transfer when dealing with anonymous scenarios. Previous proposals impose a reduction of each user reputation when transferring the reputation between pseudonyms without any other benefit than preserving unlinkability between pseudonyms. In our solution, pseudonym unlinkability comes for free since the reduction of the reputation is transformed in an economical profit thanks to the fact that reputation is expressed directly in bitcoins.

Further extensions of the proposed system may be directed to enable a single ACA shared by multiple MCS applications or even by other applications outside the filed of MCS, enabling users to create a global digital reputation score useful in multiple environments. Also, further research has to be directed to analyse the behaviour of different reputation models when reward and reputation are considered as a whole.

Acknowledgments. This work is partially supported by the spanish ministry under grant number TIN2014-55243-P and the Catalan AGAUR grant 2014SGR-691.

References

1. Amintoosi, H., Kanhere, S.S.: A reputation framework for social participatory sensing systems. Mobile Netw. Appl. **19**(1), 88–100 (2013). http://link.springer.com/10.1007/s11036-013-0455-x
2. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 202–218. Springer, Heidelberg (2008)
3. Bissias, G., Ozisik, A.P., Levine, B.N., Liberatore, M.: Sybil-resistant mixing for bitcoin. In: Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society. WPES '14, NY, USA. ACM, New York (2014)
4. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: anonymity for bitcoin with accountable mixes. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 481–499. Springer, Heidelberg (2014). http://link.springer.com/chapter/10.1007/978-3-662-45472-5_31
5. Brereton, M., Roe, P., Foth, M., Bunker, J.M., Buys, L.: Designing participation in agile ridesharing with mobile social software. In: Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7. OZCHI '09, NY, USA , pp. 257–260 . ACM, New York (2009). http://doi.acm.org/10.1145/1738826.1738868
6. Christin, D., Roßkopf, C., Hollick, M., Martucci, L.A., Kanhere, S.S.: IncogniSense: an anonymity-preserving reputation framework for participatory sensing applications. Pervasive Mobile Comput. **9**(3), 353–371 (2013). http://linkinghub.elsevier.com/retrieve/pii/S1574119213000382
7. Derlath, J.: Gamification - What is the benefit for community management (2013)
8. Dutta, P., Aoki, P.M., Kumar, N., Mainwaring, A., Myers, C., Willett, W., Woodruff, A.: Common sense. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems - SenSys '09, New York, USA, p. 349. ACM, New York, August 2009. http://dl.acm.org/citation.cfm?id=1644038.1644095
9. Eisenman, S.B., Miluzzo, E., Lane, N.D., Peterson, R.A., Ahn, G.S., Campbell, A.T.: BikeNet. ACM Trans. Sens. Netw. **6**(1), 1–39 (2009). http://dl.acm.org/citation.cfm?id=1653760.1653766
10. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world. https://bitcointalk.org/index.php?topic=279249. Aug 2013, 17 Jun 2015
11. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. **43**(2), 618–644 (2007). http://www.sciencedirect.com/science/article/pii/S0167923605000849
12. Koutsopoulos, I.: Optimal incentive-driven design of participatory sensing systems. In: 2013 IEEE Proceedings of INFOCOM, pp. 1402–1410, April 2013

13. Krontiris, I., Maisonneuve, N.: Participatory sensing: the tension between social translucence and privacy. In: Salgarelli, L., Bianchi, G., Blefari-Melazzi, N. (eds.) Trustworthy Internet, pp. 159–170. Springer, Heidelberg (2011). http://link.springer.com/chapter/10.1007/978-88-470-1818-1_12
14. Lane, N.D., Eisenman, S.B., Musolesi, M., Miluzzo, E., Campbell, A.T.: Urban sensing systems: opportunistic or participatory?. In: Proceedings of the 9th Workshop on Mobile Computing Systems and Applications. HotMobile '08, NY, USA, pp. 11–16. ACM, New York (2008). http://doi.acm.org/10.1145/1411759.1411763
15. Li, Q., Cao, G.: Providing privacy-aware incentives for mobile sensing. In: IEEE International Conference on Pervasive, pp. 76–84. No. March, San Diego (2013). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6526717
16. Miranda, H., Rodrigues, L.: A framework to provide anonymity in reputation systems. In: 2006 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops, pp. 1–4, July 2006. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4205248
17. Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems - SenSys '08, New York, USA, p. 323. ACM, New York, November 2008. http://dl.acm.org/citation.cfm?id=1460412.1460444
18. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Commun. ACM **43**(12), 45–48 (2000). http://doi.acm.org/10.1145/355112.355122
19. Tanas, C., Herrera-Joancomartí, J.: When users become sensors: Can we trust their readings? International Journal of Communication Systems: published online in Wiley. Online Library (2013). doi:10.1002/dac.2689
20. The Bitcoin Wiki: A general overview of the bitcoin system and economy (2014). https://en.bitcoin.it/wiki/Introduction
21. torproject: Tor, 17 June 2015. https://www.torproject.org/about/overview.html.en
22. Werbach, K.: For the Win: How Game Thinking Can Revolutionize Your Business. Wharton Digital Press (2012). http://www.amazon.co.uk/For-Win-Thinking-Revolutionize-Business/dp/1613630239
23. Yang, D., Xue, G., Fang, X., Tang, J.: Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, pp. 173–184. Mobicom '12, NY, USA. ACM, New York (2012). http://doi.acm.org/10.1145/2348543.2348567
24. Zhong, S., Li, L.E., Liu, Y.G., Yang, Y.R.: On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks. Wirel. Netw. **13**(6), 799–816 (2006). http://link.springer.com/10.1007/s11276-006-9855-1

Stronger Security for Sanitizable Signatures

Stephan Krenn[1], Kai Samelin[2,3](✉), and Dieter Sommer[2]

[1] AIT Austrian Institute of Technology GmbH, Vienna, Austria
stephan.krenn@ait.ac.at
[2] IBM Research – Zurich, Zurich, Switzerland
{ksa,dso}@zurich.ibm.com
[3] Technical University of Darmstadt, Darmstadt, Germany

Abstract. Sanitizable signature schemes ($\mathcal{SSS}$) enable a designated party (called the *sanitizer*) to alter admissible blocks of a signed message. This primitive can be used to remove or alter sensitive data from already signed messages without involvement of the original signer.

Current state-of-the-art security definitions of $\mathcal{SSS}$s only define a "weak" form of security. Namely, the unforgeability, accountability and transparency definitions are not strong enough to be meaningful in certain use-cases. We identify some of these use-cases, close this gap by introducing stronger definitions, and show how to alter an existing construction to meet our desired security level. Moreover, we clarify a small yet important detail in the state-of-the-art privacy definition. Our work allows to deploy this primitive in more and different scenarios.

1 Introduction

Traditional digital signature schemes such as RSA-PSS require that a signature σ on a message m becomes invalid as soon as a single bit of m is altered [1,2]. Contrary, many use-cases require subsequent changes to the signed data by a semi-trusted third party. As a simple example, consider a driver's license which is signed by the issuing state. To prove majority, the holder wants to remove all information but the date of birth and its picture to preserve his privacy. Obviously, having the data re-signed by the state every time the holder needs to prove its age induces too much overhead to be practical in this scenario. This constellation is widely known as the "digital document sanitization problem" [3].

Sanitizable signature schemes ($\mathcal{SSS}$) [4] address the aforementioned shortcomings. They allow for altering all signer-chosen admissible blocks $m[i]$ of a given message $m = (m[1], \ldots, m[i], \ldots, m[\ell])$ to different bitstrings $m[i]' \in \{0,1\}^*$ by the sanitizer, which holds its own private key. In particular, a sanitization of a message m creates an altered message $m' = (m[1]', \ldots, m[i]', \ldots, m[\ell]')$, where $m[i] = m[i]'$ for every non-admissible block, and a signature σ', verifying under the given public keys.

This work was supported by the Horizon 2020 project PRISMACLOUD under grant agreement no. 644962, and the FP7 projects FutureID and AU2EU under grant agreement nos. 318424 and 611659. Parts of this work were done while the first author was at IBM Research – Zurich.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 100–117, 2016.
DOI: 10.1007/978-3-319-29883-2_7

Application scenarios include secure routing, privacy-preserving handling of patient data, official document disclosure, and blank signatures [4–9].

Organization. This paper is structured as follows. The remainder of this section is devoted for pointing out the problems in current definitions, our contribution and existing related work. The required preliminaries are given in Sect. 2. The security model of $\mathcal{SSS}$s is revised in Sect. 3, while Sect. 4 contains the altered construction. We conclude our work in Sect. 5.

Our Contribution. We review the existing state-of-the-art security model for $\mathcal{SSS}$s and show that it is not sufficient for certain use-cases. We then revise the model, resulting in strictly stronger security definitions. Finally, we sketch how to alter the construction of *Bruszka* et al. [5], such that it satisfies our new definitions. More precisely, we introduce the notions of strong transparency, strong sanitizer- and signer-accountability, strong non-interactive public accountability, and strong unforgeability. Moreover, we show that the original privacy definition may not be clear enough and we provide a clearer definition. To get a first understanding, we briefly describe each security property. The corresponding formal definitions are given in Sect. 3. We want stress that we work in the single signer/single sanitizer model. Extensions to multiple signer/multiple sanitizer environments are straightforward.

(Strong) Unforgeability. In $\mathcal{SSS}$s, there are two key pairs, one for the signer and one for the sanitizer. Unforgeability requires that an adversary, not having access to any of these private keys, cannot generate a signature on messages not endorsed by the signer or the sanitizer.

Our new stronger definition also requires that the adversary cannot generate *any* new signatures on its own, even with fully adaptive oracle access.

Immutability. As already mentioned before, the signer is able to define non-admissible blocks, i.e., blocks which can neither be modified by an outsider nor by the sanitizer. Immutability thus requires that even a sanitizer holding its own private key cannot change non-admissible blocks.

Privacy. A very important property is privacy. It defines that once a message has been sanitized, an outsider cannot derive any information about the original content. Clearly, privacy is necessary to have a meaningful primitive. We clarify a small detail in the current state-of-the-art definition, which may be easily overlooked. Jumping ahead, in the security game for privacy, there are algorithms used whose behavior we clarify, i.e., make explicit.

(Strong) Transparency. The property of transparency is a stronger privacy definition. In particular, if an $\mathcal{SSS}$ is transparent, an outsider not holding any private keys cannot decide whether a signature was generated by the signer or the sanitizer, if the corresponding message was never de-anonymized. In other words,

transparency is the anonymity of the accountable party. Note, the naming is due to historic reasons [4]. Our stronger definition guarantees transparency, as long as no message/signature *pair* was de-anonymized. We want to stress that *Ateniese* et al. already defined strong transparency in [4]. However, *Pöhls* et al. showed that this property is not defined correctly [10]. Hence, our definition does not collide with their definition.

Unlinkability. Unlinkability is also a very strong privacy notion. It prohibits an adversary from deciding from which signature a derived signature was created.

(Strong) Signer-Accountability. As $\mathcal{SSS}$s allow to alter signed data, it must be derivable which party is responsible for a given message/signature pair. Signer-accountability thus requires that the signer cannot deny that the message originated from itself, if it was signed by the signer.

Our stronger definition also accounts for the signature, i.e., even the signer cannot blame the sanitizer for a *signature* which the sanitizer did not generate.

(Strong) Sanitizer-Accountability. Sanitizer-accountability is the counterpart to signer-accountability. In particular, the roles are reversed. This is also true for our stronger definition.

(Strong) Non-Interactive Public-Accountability. In the original definition of $\mathcal{SSS}$s, the accountable party could only be derived if the signer provides a proof π which can be generated if the signer's private key is known [11]. This contradicts legal and application requirements [6,7,12].

Non-interactive public accountability solves this problem by requiring that the accountable party can be derived without requiring any information from either the signer or the sanitizer. This property is mutually exclusive with transparency. As before, our strong definition also takes the signature into account.

Discussion. One may argue that our stronger definitions only make sense in the context of re-randomizable signatures or similar primitives. However, the (contrived) construction given in Sect. 3 shows that this is not true: there are cases which are trivially insecure in our setting, regardless of the signature scheme used. Moreover, we want to explicitly stress that, on first sight, our changes to the existing definitions seem to be rather technical and only addressing details. However, the provided examples prove that our new definitions may open a wider deployment of this primitive. In other words, this paper wants to raise awareness that certain constellations where $\mathcal{SSS}$s are used need extra precautions and guidance on how to avoid these pitfalls.

Motivation. Standard digital signatures normally have the property of existential unforgeability against chosen-message attacks (eUNF-CMA) [2]. Roughly, this definition says that an adversary cannot find a signature σ^* on a new message m^* even it has access to a signing oracle which it can query adaptively on messages of its own choice. Later, a stronger notion named "strong existential unforgeability against chosen-message attacks" (seUNF-CMA) has been introduced [13]. In this definition, an adversary must not be able to find any message/signature pair (m^*, σ^*) which it has not seen before, i.e., the adversary must not be able to generate *any* signature σ^* on its own, even if it already knows arbitrarily many signatures on messages of its own choice, potentially including m^*. This allows for a broader use of digital signatures [13,14]. We follow the same line of research for $\mathcal{SSS}$s. In a nutshell, we now also consider the signatures in the security definitions; as $\mathcal{SSS}$s have three parties, this is an even more subtle task than for standard signatures.

Let us further elaborate: first, for unforgeability, the very same problems as for standard signatures apply. Namely, once an adversary learns a signature σ on some message m, it can potentially generate many new signatures σ_i^* on the same message m, which then would trace back to the sanitizer or signer, respectively, without this party having the chance of denying the authenticity of the message/signature pair.

Next, transparency, i.e., the anonymity of the accountable party, is only guaranteed for message/signature pairs (m_i, σ_i), where the *message* m_i has never been queried to the proof oracle (which informally allows for determining the accountable party; cf. Sect. 2). However, this means that an outsider might be able to determine the accountable party for *all* signatures on messages queried to the proof oracle, which is a problem if certain messages are signed or sanitized more than once. We clarify this statement by providing a construction secure in the old model which is "obviously" insecure in real-world scenarios in Sect. 3.

For all three accountability definitions, similar problems apply. Consider the following examples which clarify our claims:

Examples. Assume that a medical doctor signs a patient record with an $\mathcal{SSS}$. The signed medical record can be sanitized by a server. It removes identifying information from the record before giving it to the accountant, which, in turn, charges the insurance company. This procedure allows to protect the patients' privacy. The current unforgeability definition now guarantees that no party can generate signatures for *new messages*. However, the definition does not guarantee that an outsider cannot generate new signatures on *already seen* signed messages. This means that the accountant may be able to generate new signatures. If the invoice only consists of the amount due and the treatments, the accountant can thus charge more than once if no extra precautions are deployed, as, e.g., an invoice for a treatment of a broken bone is a common scenario.

As we have two "signers" and a verifier, there also exist intermediate stages, namely for all three accountability definitions. In particular, a corrupt server in our hospital example may alter messages in a way that the medical doctor is accountable, while it is not, e.g., by altering cheap treatments to expensive treatments which have existed before. Thus, the insurance company is charged more, while the accountable party for that particular *signature* may not be the party who has generated the signature. The same is true for the medical doctor: if the signer signs fresh messages which have already been output by the sanitizer, the current definition does not guarantee that the signer cannot blame the sanitizer.

For transparency it is even more problematic as for accountability and unforgeability. Again, the current definition only guarantees that transparency only holds for message/signature pairs (m_i, σ_i) for which the *message* m_i has never been "opened", i.e., the signer generated a proof which allows for tracing the accountable party. Depending on the use-case, this may also not always be desired. To be more precise, consider criminal records. In some countries, these records are deleted after ten years. If these records are signed by a national authority using an $\mathcal{SSS}$, a local municipal office can later sanitize the corresponding entries. However, once it comes to dispute over a single record and it is checked whether it had been sanitized or not, potentially all equivalent records can be traced to either the sanitizer or the signer. If, in this slightly simplified scenario, a record only contains the name and the criminal record, the privacy of all citizens with the same name would be at risk. Obviously, this contradicts the very intention of $\mathcal{SSS}$s.

We want to explicitly note that in the scenarios sketched above standard mechanisms such as also signing a unique id or the current timestamp does not help at all: if the $\mathcal{SSS}$ is unlinkable [5,7,15], such a "tag" destroys the unlinkability of the signature. Clearly, it depends on the use-case which security notions are required.

Related Work. $\mathcal{SSS}$s have originally been introduced by *Ateniese* et al. [4]. *Brzuska* et al. formalized most of the current security properties in [11]. These have been later extended for unlinkability [5,7,15] and non-interactive public accountability [6,7]. See Sect. 3 for the definitions. Some properties discussed in [11] have then been refined in [16]; namely they also consider the admissible blocks in the security games. Recently, several extensions such as limiting the sanitizer to signer-chosen values [10,17–19], trapdoor $\mathcal{SSS}$s (which additionally allow to add new sanitizers after signature generation by the signer) [20,21], multi-sanitizer and -signer environments [7,22,23], and sanitization of signed and encrypted data [24] have been considered. Currently, the only work considering $\mathcal{SSS}$s and data-structures more complex than lists is [10]. Our results directly carry over to the aforementioned extended settings with only minor adjustments. Real implementations and the corresponding performance measurements of $\mathcal{SSS}$s have also been presented [6,7,25].

There exists much additional related work, e.g., proxy signatures [26]. *Ahn* et al. [27] and *Demirel* et al. [28] provide a comprehensive overview.

2 Preliminaries

Here, we present some basic notation and the general framework for $\mathcal{SSS}$s.

Notation. By $\lambda \in \mathbb{N}$ we denote the main security parameter. All algorithms implicitly take 1^λ as their first input. We write $a \leftarrow A(x)$ if a is assigned the output of algorithm A with input x. For a message $m = (m[1], \ldots, m[\ell])$, where $m[i] \in \{0,1\}^*$, we call $m[i]$ a block, while $\ell \in \mathbb{N}$ denotes the number of blocks in a message m. We call an algorithm efficient if it runs in probabilistic polynomial time in λ. All algorithms may return an exception $\perp \notin \{0,1\}^*$.

A function $\nu : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible*, if it vanishes faster than every inverse polynomial. That is, for every $k \in \mathbb{N}$ there exists an $n_0 \in \mathbb{N}$ such that $\nu(n) \leq n^{-k}$ for all $n > n_0$.

2.1 Sanitizable Signature Schemes

Here, we introduce the algorithms required for $\mathcal{SSS}$s and the corresponding state-of-the-art security model. The definitions are compiled from [6,7,11]. The security model is discussed and altered in Sect. 3.

Definition 1 (Sanitizable Signatures). *A sanitizable signature scheme $\mathcal{SSS}$ consists of seven efficient algorithms (KGen_{sig}, KGen_{san}, Sign, Sanit, Verify, Proof, Judge) such that:*

1. *Key Generation: There is one key generation algorithm for the signer and one for the sanitizer. Both create a key pair; a private key and the corresponding public key, w.r.t. the security parameter λ:*

$$(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda), (\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$$

2. *Signing: The Sign algorithm takes as input a message m, sk_{sig}, pk_{san}, as well as a description $\textsc{adm}$ of the admissibly modifiable blocks. $\textsc{adm}$ contains the set of indices of the modifiable blocks, as well as the number ℓ of blocks in m, while we assume that $\textsc{adm}$ can be uniquely and unambiguously derived from any valid message/signature pair. We write $\textsc{adm}(m) = 1$, if $\textsc{adm}$ is valid w.r.t. m, i.e., $\textsc{adm}$ contains the correct ℓ and all indices are in m. If $\textsc{adm}(m) = 0$, this algorithm returns $\perp$. It outputs a signature σ:*

$$\sigma \leftarrow \mathsf{Sign}(m, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \textsc{adm})$$

3. *Sanitizing: Algorithm Sanit takes a message m, modification instruction $\textsc{mod}$, a valid signature σ, pk_{sig}, and sk_{san}. It modifies the message m according to the modification instruction $\textsc{mod}$, which is a set containing pairs $(i, m[i]')$ for those blocks that shall be modified, meaning that $m[i]$ is replaced with $m[i]'$. We write $\textsc{adm}(\textsc{mod}) = 1$, if $\textsc{mod}$ is valid w.r.t. $\textsc{adm}$, meaning that the indices to be modified are contained in $\textsc{adm}$. Sanit calculates a new signature σ' for the modified message $m' \leftarrow \textsc{mod}(m)$. Then, Sanit outputs m' and σ':*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(m, \textsc{mod}, \sigma, \mathrm{pk}_{sig}, \mathrm{sk}_{san})$$

4. *Verification: The Verify algorithm outputs a decision $d \in \{\mathtt{true}, \mathtt{false}\}$ verifying the correctness of a signature σ for a message m w.r.t. the public keys pk_{sig} and pk_{san}:*

$$d \leftarrow \mathsf{Verify}(m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san})$$

5. *Proof: The* Proof *algorithm takes as input* sk_{sig}*, a message* $m = (m[1], m[2], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a valid signature* σ *and a set of (polynomially many) additional message/signature pairs* $\{(m_i, \sigma_i)\}$ *and* pk_{san}*. It outputs a string* $\pi \in \{0,1\}^*$ *which can be used by the next algorithm (*Judge*) to derive the accountable party of a given message/signature pair:*

$$\pi \leftarrow \mathsf{Proof}(\mathrm{sk}_{sig}, m, \sigma, \{(m_i, \sigma_i) \mid i \in \mathbb{N}\}, \mathrm{pk}_{san})$$

6. *Judge: Algorithm* Judge *takes as input a message* m*, a valid signature* σ*, both public keys and a proof* π*. Note, this means that once a proof* π *is generated, no additional action is required with the signer or the sanitizer. It outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}\}$ *indicating whether the message/signature pair has been created by the signer or a sanitizer:*

$$d \leftarrow \mathsf{Judge}(m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}, \pi)$$

We require the usual correctness requirements to hold [11].

3 Revisiting the Security Properties

Here, we introduce the stronger security framework for $\mathcal{SSS}$s. The non-strong definitions are compiled from [5–7,11]. Our strong definitions are also based on the work done in [5–7,11]. We do not restate the properties we strengthen; we will highlight the differences in each definition. If one wants to also consider ADM as a part which needs to be protected, one needs to add the alterations presented in [16].

Next, we discuss the shortcomings of some of the security definitions and provide strictly stronger variants. In particular, we review and revise the following properties and introduce their strengthened/clarified definitions:

– Unforgeability
– Privacy
– Transparency
– Signer Accountability
– Sanitizer Accountability
– Non-Interactive Public Accountability

In a nutshell, for almost all of the aforementioned security definitions, we also consider the signature in the security definitions. As already argued, this seems to be a small technical detail, but has a significant impact on the resulting security. For transparency, which we think has the largest impact, we clarify this statement by introducing a (contrived) scheme which is trivially insecure in our setting, while it achieves the weaker state-of-the-art transparency notion.

3.1 Security of Sanitizable Signatures Re-Revisited

Strong Unforgeability. The first notion we want to revisit is unforgeability. The original definition given in [11] allows the adversary to derive new signatures

```
Experiment $\mathsf{SUnforgeability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
  $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot), \mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot, sk_{\text{san}}), \mathsf{Proof}(sk_{\text{sig}}, \cdot,\cdot,\cdot,\cdot)}(pk_{\text{sig}}, pk_{\text{san}})$
    for $i = 1, 2, \ldots, q$ let $(m_i, pk_{\text{san},i}, \text{ADM}_i)$ and $\sigma_i$ index the queries/answers to/from Sign
    for $j = 1, 2, \ldots, q'$ let $(m_j, \sigma_j, pk_{\text{sig},j}, \text{MOD}_j)$ and $(m'_j, \sigma'_j)$ index the queries/answers to/from Sanit
  if $\mathsf{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk_{\text{san}}) = \mathtt{true}\ \wedge$
    $\forall i \in \{1, 2, \ldots, q\} : (pk_{\text{san}}, m^*, \underline{\sigma^*}) \neq (pk_{\text{san},i}, m_i, \underline{\sigma_i})\ \wedge$
    $\forall j \in \{1, 2, \ldots, q'\} : (pk_{\text{sig}}, m^*, \underline{\sigma^*}) \neq (pk_{\text{sig},j}, m'_j, \underline{\sigma'_j})$
      return 1
  return 0
```

Fig. 1. Strong Unforgeability

on already queried messages, as the winning condition of the security game requires that the adversary outputs a message that was not queried before.

As already argued, this is not desired in all use-cases, very similar to the case of standard digital signatures. Hence, we alter the definition in such a way that the adversary cannot even generate any new signatures on its own, highlighted by the additional underlined conditions. In other words, in the strong definition of unforgeability, the adversary also wins if it can generate *any* new valid signature σ^* on its own.

Definition 2 (Strong Unforgeability). *An $\mathcal{SSS}$ is strongly unforgeable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr[\mathit{SUnforgeability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] \leq \nu(\lambda),$$

where the experiment is defined in Fig. 1.

We now obtain the following separation result:

Theorem 1. *Every scheme which is strongly unforgeable, is also unforgeable. The converse is not true.*

Proof. The proof is straightforward. Let $\mathcal{A}$ be an adversary winning the standard unforgeability game. We can then construct an adversary $\mathcal{B}$ which uses $\mathcal{A}$ internally to break the strong unforgeability of an $\mathcal{SSS}$. $\mathcal{B}$ proceeds as follows:

1. $\mathcal{B}$ receives both public keys of its own challenger.
2. $\mathcal{B}$ passes the public keys to $\mathcal{A}$.
3. $\mathcal{B}$ simulates all oracles of $\mathcal{A}$ using its own oracles without any modifications.
4. Eventually, $\mathcal{A}$ outputs (m^*, σ^*). $\mathcal{B}$ also outputs (m^*, σ^*) as its own forgery.
5. As for all queries i to the signing oracle $(pk_{\text{san}}, m^*) \neq (pk_{\text{san},i}, m_i)$ yields and for all queries j to the sanitization oracle we also have $(pk_{\text{sig}}, m^*) \neq (pk_{\text{sig},j}, m'_j)$, (m^*, σ^*) clearly breaks the strong unforgeability of $\mathcal{SSS}$ with the same probability as $\mathcal{A}$ wins its own game.

The other direction is also easy: for each signature generation, we append a 0. For processing a signature, the last bit is removed. An adversary can exchange the 0 with a 1, while the signature verification is not affected. $\square$

```
Experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\lambda})$
  $(m^*, \sigma^*, pk^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathrm{sig}}, \cdot, \cdot), \mathsf{Proof}(sk_{\mathrm{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\mathrm{sig}})$
    for $i = 1, 2, \ldots, q$ let $(m_i, pk_{\mathrm{san},i}, \mathrm{ADM}_i)$ index the queries to Sign
  return 1, if
    $\mathsf{Verify}(m^*, \sigma^*, pk_{\mathrm{sig}}, pk^*) = \mathtt{true} \wedge$
    $(\forall i \in \{1, 2, \ldots, q\} : pk^* \neq pk_{\mathrm{san},i} \ \vee\ m^* \notin \{\mathrm{MOD}(m_i) \mid \mathrm{MOD} \text{ with } \mathrm{ADM}_i(\mathrm{MOD}) = 1\})$
```

Fig. 2. Immutability

Jumping back to our example in Sect. 1, the new definition now prohibits the attacks: the accountant is bound to the signatures it receives from the server and cannot generate any new ones.

Immutability. As already aforementioned, a sanitizer should only be able to alter admissible blocks defined by ADM. Hence, also deleting or appending blocks must be prohibited. As usual, the adversary is given full oracle access, while it is also allowed to generate the sanitizer key pair.

Definition 3 (Immutability). *An $\mathcal{SSS}$ is immutable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr[\mathsf{Immutability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 2.

Privacy. Privacy is related to the indistinguishability of ciphertexts. In particular, the adversary is allowed to input two messages with the same ADM which are sanitized to the exact same message. Then, the adversary has to decide which message was used to generate the sanitized one. Again, the adversary receives full adaptive oracle access.

Definition 4 (Privacy). *An $\mathcal{SSS}$ is private, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\left|\Pr[\mathsf{Privacy}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 3.

Discussion. Compared to the original definition given in [11], we additionally check whether ADM "matches" *both* messages. This has a severe impact, depending on how one interprets the constraints given in [11]. Namely, [11] only requires that MOD_0 and MOD_1 are compatible with ADM, while "the resulting modified messages are identical for both tuples" [11]. However, it is not clear what this concretely means if ADM does not match one of the *messages*. More precisely, the adversary may proceed as follows: it chooses $m_0 = (1, 2)$

```
**Experiment** $\mathsf{Privacy}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\lambda)$
  $(pk_{\mathrm{san}}, pk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathrm{sig}},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{san}}),\mathsf{Proof}(sk_{\mathrm{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{LoRSanit}(\cdot,\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{sig}},sk_{\mathrm{san}},b)}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    where oracle LoRSanit on input of $m_0, \textsc{mod}_0, m_1, \textsc{mod}_1, \textsc{adm}$
      if $\textsc{mod}_0(m_0) \neq \textsc{mod}_1(m_1)$ $\underline{\vee\ \textsc{adm}(m_0) \neq \textsc{adm}(m_1)}$, return $\perp$
      let $\sigma \leftarrow \mathsf{Sign}(m_b, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \textsc{adm})$
      return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, \textsc{mod}_b, \sigma, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
  return 1, if $a = b$. Else, return 0.
```

Fig. 3. Privacy

with $\textsc{adm} = \{\{1,2\}, \ell = 2\}$. Let the other message be $m_1 = (1)$. Clearly, $\textsc{adm}(m_1) = 0$, i.e., $\textsc{adm}$ is not valid w.r.t. the *message* m_1. Further consider $\textsc{mod}_0 = \textsc{mod}_1 = \{(2,3)\}$. Depending on how one implements the modification algorithm *in detail*, we may have $\textsc{mod}_0(m_0) = \textsc{mod}_1(m_1) = (1,3)$, as it is not specified whether $\textsc{mod}$ returns $\perp$ for $\textsc{mod}_1(m_1)$. However, now Sign returns $\perp$ for $b = 1$, but a signature if $b = 0$. Clearly, the adversary can easily derive b from the information it sees. We assume that prior art, e.g., [5,11], implicitly requires that $\textsc{mod}(m)$ returns $\perp$ if $\textsc{adm}(\textsc{mod}) = 0$. In our definition, we made this explicit.

Strong Transparency. As mentioned earlier, the standard transparency definition only guarantees that the identity of the accountable party of a message m remains anonymous as long as no signature on this message m has ever been opened, i.e., no proof π has ever been created for that m. That is, as soon as the issuer of a signature on a certain message has been revealed, all previous and future signatures on the same message, regardless whether they are freshly signed or sanitized, may potentially be de-anonymized as well. As already argued, this might have undesirable side effects in practice. Our strong transparency definition in Fig. 4 prohibits such attacks. Considering our example, even if it comes to a dispute over a sanitized criminal record, all other records remain transparent.

Definition 5 (Strong Transparency). *An $\mathcal{SSS}$ is strongly proof-restricted transparent, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\left|\Pr[\mathsf{STransparency}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 4.

Discussion. Even though a bit more subtle (as anonymity is only potentially affected per message), the difference between the original definition and Definition 4 is comparable to the difference between, e.g., CPA- and CCA-anonymity for group signature schemes [29]. For CPA-anonymity the anonymity of the issuer of a signature is only guaranteed as long as *no* signature has never been

```
Experiment $\mathsf{STransparency}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\lambda)$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,sk_{\mathrm{sig}},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{san}}),\mathsf{Proof}(sk_{\mathrm{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}(\cdot,\cdot,\cdot,sk_{\mathrm{sig}},sk_{\mathrm{san}},b)}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    where oracle Sanit/Sign on input of $m$, MOD, ADM:
      $\sigma \leftarrow \mathsf{Sign}(m, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \mathrm{ADM})$
      $(m', \sigma') \leftarrow \mathsf{Sanit}(m, \mathrm{MOD}, \sigma, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
      if $b = 1$:
        $\sigma' \leftarrow \mathsf{Sign}(m', sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \mathrm{ADM})$
      return $(m', \sigma')$
  if $\mathcal{A}$ has queried any $(m', \underline{\sigma'})$ output by Sanit/Sign to Proof, return a random bit
  return 1, if $a = b$. Else, return 0.
```

Fig. 4. Strong Transparency

inspected, i.e., de-anonymized, but all signers might be at risk once a single signature was inspected. However, for CCA-anonymity privacy is guaranteed for each signature as long as this specific signature has not been de-anonymized.

To clarify this statement in the context of $\mathcal{SSS}$s, we derive a (contrived) construction which is transparent, but trivially does not meet our strong transparency definition. Let $\mathcal{E} = \{\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a secret-key CCA2-secure encryption scheme. We require that the secret-key and message space of $\mathcal{E}$ contains $\{0,1\}^\lambda$. Moreover, let $\mathcal{E}' = \{\mathsf{EKGen}', \mathsf{Enc}', \mathsf{Dec}'\}$ denote a public-key CCA2-secure encryption scheme. Let $\mathcal{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ be a pseudo-random function, and $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be a random oracle.

Let $\mathcal{SSS} = (\mathsf{KGen}_{\mathrm{sig}}, \mathsf{KGen}_{\mathrm{san}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ be a sanitizable signature scheme according to the already existing standard definitions. We now construct a contrived sanitizable signature scheme $\mathcal{SSS}'$ from $\mathcal{SSS}$ which still fulfills the standard definitions but not strong transparency. For brevity, we skip obvious checks.

Construction 1. *We define* $\mathcal{SSS}' = (\mathsf{KGen}'_{sig}, \mathsf{KGen}'_{san}, \mathsf{Sign}', \mathsf{Sanit}', \mathsf{Verify}', \mathsf{Proof}', \mathsf{Judge}')$ *as follows:*

1. *Key Generation: For the key generation, we alter the algorithms as follows. We start with* KGen'_{sig}*: create a key* $\kappa \leftarrow \{0,1\}^\lambda$ *for the* $\mathcal{PRF}$ *at random. Return* $(\mathrm{pk}_{sig}, \mathrm{sk}'_{sig})$*, where* $\mathrm{sk}'_{sig} = (\kappa, \mathrm{sk}_{sig})$*, and* $(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda)$ *is generated as in the original* $\mathcal{SSS}$*.* KGen'_{san} *generates an additional key pair of the public-key encryption scheme, i.e.,* $(\mathrm{pk}', \mathrm{sk}') \leftarrow \mathsf{EKGen}'(1^\lambda)$*. It returns* $((\mathrm{pk}_{san}, \mathrm{pk}'), (\mathrm{sk}_{san}, \mathrm{sk}'))$*, where* $(\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$*, i.e., the same keys as in the original* $\mathcal{SSS}$*.*
2. *Signing:* Sign' *computes* $\sigma \leftarrow \mathsf{Sign}(m, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \mathrm{ADM})$*. Let* $k \leftarrow \mathcal{PRF}(\kappa, \mathcal{H}(\mathrm{pk}_{san}))$*, and* $k' \leftarrow \mathcal{PRF}(k, \mathcal{H}(m))$*. If* $\sigma = \perp$*, return* $\perp$*. Then,* Sign' *returns* $\sigma' = (\sigma, c_1, c_2)$*, where* $c_1 \leftarrow \mathsf{Enc}(k', 0^\lambda)$ *and* $c_2 \leftarrow \mathsf{Enc}'(\mathrm{pk}', k)$*.*
3. *Sanitizing: Parse* σ *as* (σ', c_1, c_2)*. Let* $k \leftarrow \mathsf{Dec}'(\mathrm{sk}', c_2)$*. Calculate* $(m', \sigma'') \leftarrow \mathsf{Sanit}(m, \mathrm{MOD}, \sigma', \mathrm{pk}_{sig}, \mathrm{sk}_{san})$*. If* $\sigma' = \perp$*, return* $\perp$*. Let* $k' \leftarrow \mathcal{PRF}(k, \mathcal{H}(m')))$*,* $c'_1 \leftarrow \mathsf{Enc}(k', 1^\lambda)$*, and* $c'_2 \leftarrow \mathsf{Enc}'(\mathrm{pk}', k)$*. Finally, return* $(m', (\sigma'', c'_1, c'_2))$*.*

4. *Verification: Parse σ as (σ', c_1, c_2). Return $\mathsf{Verify}(m, \sigma', \mathrm{pk}_{sig}, \mathrm{pk}_{san})$.*
5. *Proof: Parse σ as (σ', c_1, c_2) and each (m_i, σ_i) as $(m_i, (\sigma'_i, c_{1,i}, c_{2,i}))$. Let $\pi \leftarrow \mathsf{Proof}(\mathrm{sk}_{sig}, m, \sigma', \{(m_i, \sigma'_i)\}, \mathrm{pk}_{san})$. If $\pi = \perp$, return $\perp$. Else, let $k \leftarrow \mathcal{PRF}(\kappa, \mathcal{H}(\mathrm{pk}_{san}))$, and $k' \leftarrow \mathcal{PRF}(k, \mathcal{H}(m))$. Return $\pi' = (\pi, k')$.*
6. *Judge: Parse π as (π', k') and σ as (σ', c_1, c_2). Return $\mathsf{Judge}(m, \sigma', \mathrm{pk}_{sig}, \mathrm{pk}_{san}, \pi')$.*

Clearly, all security properties are still preserved. However, as Proof now also returns the secret key to decrypt c_1, an adversary can now decide for each *message* which has been queried to Proof how the corresponding signature was generated by decrypting, which is contained in the *signature* σ. We want to stress that [25] also defines transparency in a similar way. However, their scheme does not fulfill this definition, as the authors do not use strongly unforgeable signatures. In [6,7] transparency is defined similarly; however, as their schemes fulfill non-interactive public accountability and not transparency, these errors seem to be typos.

Theorem 2. *Every scheme which is strongly transparent, is transparent. The converse is not true.*

Proof. The claim immediately follows from the construction above.

Unlinkability. Unlinkability prohibits an adversary to decide how a signature was generated, i.e., from which signature a sanitized signature was derived. We introduce the stronger definition from [7], where even the signer can be malicious. This game is similar to privacy with the same constraints. However, compared to the privacy game, the adversary can also input the signatures and the modification instruction. Again, it receives full oracle access; the signing and proof oracles can be simulated by the adversary.

Definition 6 (Unlinkability). *An $\mathcal{SSS}$ is unlinkable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\left| \Pr[\mathsf{Unlinkability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] - \frac{1}{2} \right| \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 5.

Next, we revise the three accountability definitions. As already argued, these definitions capture the intermediate cases, i.e., when either the sanitizer or the signer is adversarial. This is captured within the next three definitions.

Strong Signer-Accountability. For strong signer-accountability, a signer should not be able to blame a sanitizer if the sanitizer is actually not responsible for a given message/signature *pair* never generated by the sanitizer. Hence, the adversary has to generate a proof π^* which makes Judge to decide that the sanitizer is accountable, if it is not for a message/signature pair (m, σ). Here, the adversary gains access to all oracles related to sanitizing.

```
**Experiment** $\mathsf{Unlinkability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{san}}),\mathsf{LoRSanit}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{san}},b)}(pk_{\mathrm{san}})$
    where oracle $\mathsf{LoRSanit}$ on input of $m_0, \textsc{mod}_0, \sigma_0, m_1, \textsc{mod}_1, \sigma_1, pk_{\mathrm{sig}}$:
      if $\textsc{adm}_0 \neq \textsc{adm}_1 \vee \textsc{mod}_0(m_0) \neq \textsc{mod}_1(m_1) \vee \textsc{adm}_0(\textsc{mod}_0) \neq \textsc{adm}_1(\textsc{mod}_1) \vee$
      $\mathsf{Verify}(m_0, \sigma_0, pk_{\mathrm{sig}}, pk_{\mathrm{san}}) \neq \mathsf{Verify}(m_1, \sigma_1, pk_{\mathrm{sig}}, pk_{\mathrm{san}})$, return $\perp$
      return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, \textsc{mod}_b, \sigma_b, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
  return 1, if $a = b$. Else, return 0.
```

Fig. 5. Unlinkability

```
**Experiment** $\mathsf{SSig\text{-}Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $(pk^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\mathrm{san}})}(pk_{\mathrm{san}})$
    for $i = 1, 2, \ldots, q$ let $(m'_i, \sigma'_i)$ and $(m_i, \textsc{mod}_i, \sigma_i, pk_{\mathrm{sig},i})$ index the answers/queries from/to $\mathsf{Sanit}$
  if $\mathsf{Verify}(m^*, \sigma^*, pk^*, pk_{\mathrm{san}}) = \mathtt{true} \wedge$
    $\forall i \in \{1, 2, \ldots, q\} : (pk^*, m^*, \underline{\sigma^*}) \neq (pk_{\mathrm{sig},i}, m'_i, \underline{\sigma'_i}) \wedge$
    $\mathsf{Judge}(m^*, \sigma^*, pk^*, pk_{\mathrm{san}}, \pi^*) = \mathtt{San}$
      return 1
  return 0
```

Fig. 6. Strong Signer Accountability

Definition 7 (Strong Signer Accountability). *An $\mathcal{SSS}$ is strongly signer accountable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr[\mathsf{SSig\text{-}Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 6.

Theorem 3. *Every scheme which is strongly signer-accountable, is also signer-accountable. The converse is not true.*

Proof. Similar to the proof of Theorem 1.

Strong Sanitizer-Accountability. Strong sanitizer-accountability is similar to standard sanitizer-accountability. However, compared to the original definition, the adversary now also wins if it can generate any signature σ^* which has not been generated by the signer. As usual, the adversary gains access to all signer-related oracles.

Definition 8 (Strong Sanitizer-Accountability). *An $\mathcal{SSS}$ is sanitizer accountable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr[\mathsf{SSan\text{-}Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 7.

```
Experiment $\mathsf{SSan\text{-}Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^{\lambda})$
  $b \leftarrow \{0,1\}$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot), \mathsf{Proof}(sk_{\text{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\text{sig}})$
    for $i = 1, \ldots, q$ let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and $\sigma_i$ index the queries/answers to/from $\mathsf{Sign}$
  $\pi \leftarrow \mathsf{Proof}(sk_{\text{sig}}, m^*, \sigma^*, \{(m_i, \sigma_i) \mid 0 < i \leq q\}, pk^*)$
  if $\mathsf{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \mathtt{true}\ \wedge$
    $\forall i \in \{1, 2, \ldots, q\} : (pk^*, m^*, \underline{\sigma^*}) \neq (pk_{\text{san},i}, m_i, \underline{\sigma_i})\ \wedge$
    $\mathsf{Judge}(m^*, \sigma^*, pk_{\text{sig}}, pk^*, \pi) = \mathtt{Sig}$
      return 1
  return 0
```

Fig. 7. Strong Sanitizer Accountability

Theorem 4. *Every scheme which is strongly sanitizer-accountable, is also sanitizer-accountable. The converse is not true.*

Proof. Similar to the proof of Theorem 1.

Strong Non-Interactive Public Accountability. Strong non-interactive public accountability is also similar to the standard definition of non-interactive public accountability. However, this definition now also covers the case where the adversary could generate new signatures.

Definition 9 (Strong Non-Interactive Public Accountability). *An $\mathcal{SSS}$ is strongly non-interactive publicly accountable, if for any efficient adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr[\mathit{SPubaccountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the experiment is defined in Fig. 8.

Theorem 5. *Every scheme which is strongly non-interactive publicly accountable, is also non-interactive publicly accountable. The converse is not true.*

Proof. Similar to the proof of Theorem 1.

Discussion. Clearly, all three definitions above fix the problems given in the running example: the server cannot generate any signature which has not been endorsed by the medical doctor, and vice versa.

Moreover, some known implications of the existing security properties from [5–7,11] carry over to our setting. Namely, strong sanitizer-accountability and strong signer-accountability together imply strong unforgeability, while strong non-interactive public accountability implies both strong sanitizer-accountability and strong signer-accountability. As strong transparency implies transparency, it also implies (proof-restricted) privacy. The proofs for this statement are essentially the same as given for the non-strong ones [5–7,11] and are therefore omitted. We leave it as open work to prove if other implications and separations are still valid in our setting.

```
Experiment $\mathsf{SPubaccountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^{\lambda})$
  $(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^{\lambda})$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \cdot, \cdot), \mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot, sk_{\text{san}})}(pk_{\text{sig}}, pk_{\text{san}})$
    for $i = 1, 2, \ldots, q$ let $(m_i, \text{ADM}_i, pk_{\text{san},i})$ and $\sigma_i$ index the queries/answers to/from $\mathsf{Sign}$
    for $j = 1, 2, \ldots, q'$ let $(m_j, \text{MOD}_j, \sigma_j, pk_{\text{sig},j})$ and $(m'_j, \sigma'_j)$ index the queries/answers to/from $\mathsf{Sanit}$
  if $\mathsf{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk^*) = \mathtt{true} \land$
    $\forall i \in \{1, 2, \ldots, q\} : (pk^*, m^*, \underline{\sigma^*}) \neq (pk_{\text{san},i}, m_i, \underline{\sigma_i}) \land$
    $\mathsf{Judge}(m^*, \sigma^*, pk_{\text{sig}}, pk^*, \perp) = \mathtt{Sig}$
      return 1
  if $\mathsf{Verify}(m^*, \sigma^*, pk^*, pk_{\text{san}}) = \mathtt{true} \land$
    $\forall j \in \{1, 2, \ldots, q'\} : (pk^*, m^*, \underline{\sigma^*}) \neq (pk_{\text{sig},j}, m'_j, \underline{\sigma'_j}) \land$
    $\mathsf{Judge}(m^*, \sigma^*, pk^*, pk_{\text{san}}, \perp) = \mathtt{San}$
      return 1
  return 0
```

Fig. 8. Strong Public Accountability

4 Constructions

We now sketch how to achieve our new notions. As (strong) non-interactive public accountability and (strong) transparency are mutually exclusive, we need two distinct constructions.

Construction with Strong Transparency. The main idea of the construction given in [5] is to sign all non-admissible blocks, the sanitizer's and the group's public key, and ADM using a strongly unforgeable deterministic signature scheme $\mathcal{S}$. Finally, the complete message, and the signer's public key are signed using a group signature scheme $\mathcal{GS}$ [5,30]. All standard properties of the $\mathcal{SSS}$ follow in a straightforward manner from the strong unforgeability and the group signature's security properties.[1] To achieve our stronger definitions, one needs a stronger group signature scheme. Namely, the anonymity, non-frameability and traceability security definitions [5,30] of the group signature scheme need to be adjusted. In particular, as done in Sect. 3, we need alter the games in the following way:[2]

– For anonymity, the adversary must never submit a pair (m, σ) for signature σ received for m from the LoR-Sign oracle to the opening oracle (instead of only m).
– For non-frameability and traceability, the adversary must never have received σ^* for m^* (instead of only not asking for a signature on m^*) from the signing oracle.

We also require that all other information in the full signature is non-malleably attached to it, e.g., by also signing them.

We leave a formal proof of security of the sketched construction as open work.

[1] Note, their scheme only achieves a weaker form of unlinkability; also the signer's key pair is generated honestly. The adversary gains oracle access to Sign and Proof [5].
[2] Due to space requirements, we assume the reader is familiar with the security definitions of group signatures. References [5,30] contain all required definitions.

Construction with Strong Non-Interactive Public Accountability. The construction given in [7] achieves non-interactive public accountability. The basic idea of their scheme is the same as in the aforementioned construction, but instead of using a group signature $\mathcal{GS}$, they use another deterministic and unique signature scheme $\mathcal{S}$. Due to the determinism and uniqueness of the signature schemes deployed, their scheme already fulfills our strong definitions without any modifications. It depends on the use-case if a deterministic $\mathcal{SSS}$ offers the required properties from a real-world perspective.

5 Conclusion

We have shown that the state-of-the-art security definitions of $\mathcal{SSS}$s are not sufficient for some use-cases we have identified. We introduced new strictly stronger definitions, accounting for these additional use-cases. In particular, we have introduced the notions of strong unforgeability, strong transparency, strong sanitizer-accountability, strong signer-accountability and strong non-interactive public accountability and clarified the property of privacy. Finally, we have sketched how to alter an existing construction to achieve our enhanced notions.

References

1. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
2. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**, 281–308 (1988)
3. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital documents sanitizing problem. Technical report 195, Institute of Electronics, Information and Communication Engineers (2003)
4. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
5. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
6. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-interactive public accountability for sanitizable signatures. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 178–193. Springer, Heidelberg (2013)
7. Brzuska, C., Pöhls, H.C., Samelin, K.: Efficient and perfectly unlinkable sanitizable signatures without group signatures. In: Katsikas, S., Agudo, I. (eds.) EuroPKI 2013. LNCS, vol. 8341, pp. 12–30. Springer, Heidelberg (2014)
8. Derler, D., Hanser, C., Slamanig, D.: Blank digital signatures: optimization and practical experiences. In: Camenisch, J., Fischer-Hübner, S., Hansen, M. (eds.) Privacy and Identity 2014. IFIP AICT, vol. 457, pp. 201–215. Springer, Heidelberg (2015)
9. Hanser, C., Slamanig, D.: Blank digital signatures. In: Asia CCS, pp. 95–106. ACM (2013)

10. Pöhls, H.C., Samelin, K., Posegga, J.: Sanitizable signatures in XML signature — performance, mixing properties, and revisiting the property of transparency. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 166–182. Springer, Heidelberg (2011)
11. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
12. Höhne, F., Pöhls, H.C., Samelin, K.: Rechtsfolgen editierbarer signaturen. Datenschutz und Datensicherheit **36**(7), 485–491 (2012)
13. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, p. 83. Springer, Heidelberg (2002)
14. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
15. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with rerandomizable keys. ePrint 395 (2015)
16. Gong, J., Qian, H., Zhou, Y.: Fully-secure and practical sanitizable signatures. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 300–317. Springer, Heidelberg (2011)
17. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
18. Derler, D., Slamanig, D.: Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In: Au, M.-H., et al. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 455–474. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26059-4_25
19. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
20. Canard, S., Laguillaumie, F., Milhau, M.: *Trapdoor* sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
21. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)
22. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Sanitizable signatures: How to partially delegate control for authenticated data. In: Proceedings of BIOSIG. LNI, vol. 155, pp. 117–128. GI (2009)
23. Canard, S., Jambert, A., Lescuyer, R.: Sanitizable signatures with several signers and sanitizers. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 35–52. Springer, Heidelberg (2012)
24. Fehr, V., Fischlin, M.: Sanitizable signcryption: sanitization over encrypteddata (full version). Cryptology ePrint Archive, Report 2015/765 (2015). http://eprint.iacr.org/
25. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: Scope of security properties of sanitizable signatures revisited. In: ARES, pp. 188–197 (2013)
26. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: CCS 1996, pp. 48–57 (1996)

27. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. Cryptology ePrint Archive, Report 2011/096 (2011). http://eprint.iacr.org/
28. Demirel, D., Derler, D., Hanser, C., Pöhls, H.C., Slamanig, D., Traverso, G.: PRISMACLOUD D4.4: overview of functional and malleable signature schemes. Technical report, H2020 Prismacloud (2015). www.prismacloud.eu
29. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
30. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)

Some Remarks and Ideas About Monetization of Sensitive Data

Ania M. Piotrowska and Marek Klonowski(✉)

Department of Computer Science, Faculty of Fundamental Problems of Technology, Wrocław University of Technology, Wrocław, Poland
Marek.Klonowski@pwr.edu.pl

Abstract. One of the emerging problems on the border of privacy protection research and e-commerce is the monetization of sensitive data. More precisely, a client would like to obtain some statistical data about users' personal information in exchange for a reward. To satisfy both parties, a monetization protocol should ensure that users' privacy is not violated and the data utility is preserved at the same time.

During ESORICS 2014 Bilogrevic et al. presented a novel and promising approach to monetization of aggregated sensitive data. In our paper, we point some flaws and shortcomings of the presented protocol. We also make some general methodological remarks to explain why some auspicious directions of data monetization might be futile. Finally, we propose a simple scheme for a secure data aggregation based on sharing trust between different non-collaborating parties.

Keywords: Monetization · Data aggregation · Privacy preserving

1 Introduction

Online services and profiles on social networks store an invaluable collection of personal information of various users. Often, they can be analyzed to provide useful and valuable information both for research purposes and e-commerce related issues. The simplest example is profiled advertising addressed to a group of users. The advertiser would like to know potential customers and is often ready to pay a revenue for knowledge about them. Similarly, the statistical information about users might be used by companies and research centers for analysis. These economic and scientific aspects contributed to fast development of the *monetization* of users data (i.e., exchange of personal information for a reward). In recent papers [17,20] it was reported that the price can range from \$0.0005 to \$33 per a single attribute of a single user. That is, such data can be very valuable. On the other hand, the individual users revealing their data might be exposed to many threats.

Even though we concentrate on online web services, the same (or similar) problem (i.e., monetization data and ensuring privacy at the same time)

Supported by Polish National Science Center DEC-2013/09/B/ST6/02251.

may emerge in many scenarios from various distributed systems ranging from sensor field collecting some environmental measurements to social networks with personal information like age or number of friends. In every case we would like to trade with some general (yet, still useful) information about the users/sensors/devices without exposing too much about individuals.

Two main concern in design of privacy-protecting data monetization schemes is how to provide useful statistical information without compromising privacy of any individual and how to evaluate the data before revealing them. During ESORICS 2014 Bilogrevic at el. presented the first privacy-preserving aggregation scheme for personal data monetization [3] (extended in [4]). In their approach, the data of individual users is aggregated before monetization and sold to a customer. The scheme presented in [3] assumes that an aggregator is untrusted and there is no single trusted third party. The proposed scheme combines two techniques to solve the users' data exchange problem. First, authors present the data aggregation protocol based on [24] for the model with untrusted aggregator. Thanks to methods introduced in [24] the protocol should guarantee **provable** privacy of an individual user, formally expressed in terms of distributed differential privacy metric (that metric is a "distributed counterpart" of widely accepted Dwork's *differential privacy* [9]). For more details see also [10]. Next, authors introduce a novel monetization framework of statistical information, in order to find a fair price for the aggregated data. The main advantage of the proposed scheme is that it can be efficiently used in many real-life scenarios, even when the computational power is relatively small (e.g., in a system of constrained devices).

In our paper, we point some flaws and shortcomings of the protocol from [3] - among others we show that the aggregator (or even some outer, adversarial party) can reveal much more information about individuals than declared. Next, we show that in some cases the way of evaluating data seems not to be adequate. We also present some general methodological remarks about constructing similar protocols - in particular on evaluating data. Finally, we present several methods that can be used to improve security/functionality of the original and possibly some similar protocols.

Despite we present our and previous results in terms of trading with data taken from users profiles, very similar mechanisms can be applied for a wide class of distributed systems processing sensitive data (including i.e., sensor networks).

1.1 Paper Organization

The rest of the paper is organized as follows. In the rest of this section, we recall some previous related works about secure data aggregation and data monetization. In Sect. 2, we recall the Bilogrevic et al.'s scheme from [3]. Section 3, is focused on the analysis of the scheme presented in [3]. We discuss advantages of this protocol and point some its flaws and shortcomings. Section 3.6, is devoted to presentation of a some methods that can be used to improve security and usability of some data aggregation protocols. In Sect. 4, we conclude and discuss some future work.

1.2 Related Works

The data aggregation problem is widely examined by different research areas in different context, including data security, computational and communication complexity (e.g., [25]). It has been investigated for general networks, nevertheless most of the recent results refer to different types of sensor networks [15,22]. Security related issues (including privacy, authenticity and accessibility) of aggregated data in sensor networks were discussed in [12–14]. Different data aggregation protocols with an untrusted aggregator have been investigated in [19,21,24]. In the model presented in [24] authors introduced a protocol, which guarantees strong privacy for individual participants. It ensures that the aggregator is *oblivious* i.e., can learn only a desired statistics of the aggregated data without gaining any additional knowledge about individuals. Even, if a subset of participants form a coalition with the aggregator and share their private data, the data of the remaining participants is still protected. The presented protocol combines the properties of homomorphic encryption and carefully parametrized statistical obfuscation methods, that ensure privacy of individual users and the utility of the aggregated data.

Papers [3] as well as [24] express privacy of individual users using the concept of *differential privacy* introduced by Dwork in [8]. Let us remark that Shi et al. in [24] use a *distributed differential privacy* notion, described by [7] and used extensively in further works. Note that this notion is in fact a natural counterpart of original Dwork's definition adapted to distributed systems. There is a long list of other papers that use this or similar definitions to evaluate privacy of different systems (i.e. [5–7,19]) ranging from data bases to RFID systems.

2 Bilogrevic et al.'s Scheme

In this section we recall the protocol presented in [3]. The idea behind it is based on the extended algorithm described in [24] combined with mechanisms for monetization of users' data. The aim is to construct an approximation of the distribution of the attributes of a group of users and finally sell it to the customer.

2.1 System Architecture

In the Bilogrevic et al.'s scheme there are three entities: a data aggregator $\mathbb{A}$, a set of n users $\mathbb{U} = \{u_1, u_2, ..., u_n\}$ and a customer $\mathbb{C}$. At the beginning of the protocol each user gets the individual, random secret key s_i and the aggregator gets its own secret key s_A. It is assumed that $s_A + s_1 + s_2 + \ldots + s_n = 0$. Each user is represented by its profile $P_i = \{x_{i1}, x_{i2}, \ldots x_{il}\}$, where x_{ij} is a value of a certain attribute, such that $x_{ij} \in \{m_j, m_j + 1, \ldots, M_j - 1, M_j\}$. It is assumed that all x_{ij} are represented by elements of a cyclic group $\mathbf{Z}_p$ for a prime p. The aggregator $\mathbb{A}$ is untrusted and should learn about the users' attributes nothing but specified statistics. The protocol works as follows: a customer $\mathbb{C}$ sends a query to the aggregator to obtain particular data. The query contains the information about the type of information which customer wants to obtain and the users. The aggregator $\mathbb{A}$ forwards the query to the users. Each user follows the obfuscation and encryption steps described in the next paragraph and sends its secured (encrypted and obfuscated by adding noise) data (i.e., values of chosen attributes) to the aggregator. Then, $\mathbb{A}$ combines all feedback messages and decrypts the result, obtaining some statistical data. For each attribute defined by the computed statistic, $\mathbb{A}$ determines the *sensitivity* of this information and the cost of sale. Next, $\mathbb{A}$ sends the statistics together with the message about each attribute containing: the sensitivity and the cost. $\mathbb{C}$ selects the aggregates to purchase. After the exchange between $\mathbb{A}$ and $\mathbb{C}$, the aggregator is responsible for sharing the total sale revenue among the users (Fig. 1).

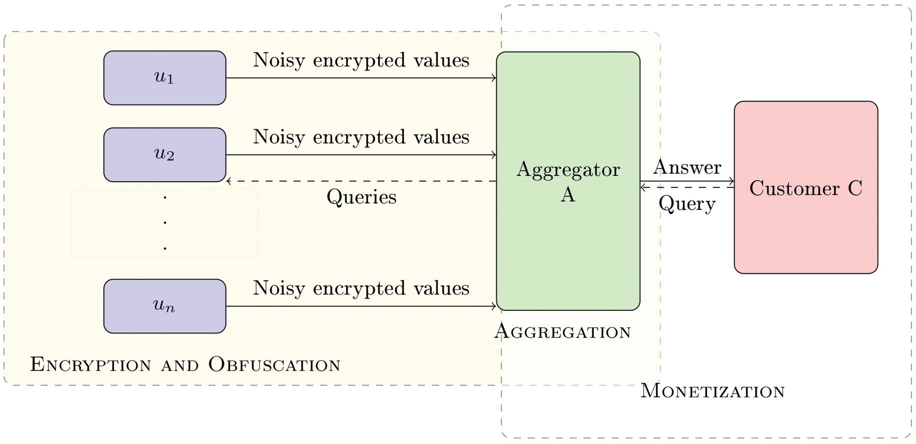


Fig. 1. System architecture

Aggregation Model. The aggregation model is an extension of the protocol described in [24]. Each user defines its own *privacy-sensitivity* value $0 \leq \lambda_{ij} \leq 1$ for each attribute $j \in \{1, \ldots, l\}$ from its profile. Values of user's attributes are unknown for the aggregator.

OBFUSCATION: First, each user generates two random values r_{ij}, o_{ij} in order to "obfuscate" its attributes.

$$\begin{aligned} \widehat{x_{ij}} &= x_{ij} + r_{ij} \mod p, \\ \widehat{x^2_{ij}} &= x^2_{ij} + o_{ij} \mod p. \end{aligned} \tag{1}$$

ENCRYPTION: Each user u_i encrypts the values $\widehat{x_{ij}}, \widehat{x^2_{ij}}$ using its own secret key s_i as follows

$$\begin{aligned} c_{ij} &= g^{\widehat{x_{ij}}} H(t)^{s_i}, \\ c_{ij}^2 &= g^{\widehat{x_{ij}^2}} H(t)^{s_i}, \end{aligned} \tag{2}$$

where t is a public timestamp. Next, the user sends to the aggregator $\mathbb{A}$ two vectors with encrypted values from its profile $c_i = [c_{i1}, c_{i2}, c_{i3}, \ldots, c_{ik}]$, $c_i^2 = [c_{i1}^2, c_{i2}^2, c_{i3}^2, \ldots, c_{ik}^2]$.

Aggregation: The aggregator combines all feedback messages from the users and decrypts the result as follows:

$$\begin{aligned} V_j &= H(t)^{s_A} \cdot c_1 \cdot c_2 \cdot \ldots \cdot c_n = H(t)^{s_A} \cdot \prod_{i=1}^{n} g^{\widehat{x_i}} H(t)^{s_i} \\ &= g^{\sum_{i=0}^{n} \widehat{x_i}} H(t)^{s_A + s_1 + \ldots + s_n} = g^{\sum_{i=0}^{n} \widehat{x_i}}, \\ W_j &= H(t)^{s_A} \cdot c_1^2 \cdot c_2^2 \cdot \ldots \cdot c_n^2 = H(t)^{s_A} \cdot \prod_{i=1}^{n} g^{\widehat{x_i^2}} H(t)^{s_i} \\ &= g^{\sum_{i=0}^{n} \widehat{x_i^2}} H(t)^{s_A + s_1 + \ldots + s_n} = g^{\sum_{i=0}^{n} \widehat{x_i^2}}. \end{aligned} \tag{3}$$

In [3] authors assume that these values are sufficiently small. Because g is a public parameter $\mathbb{A}$ can obtain values $\sum_{i=0}^{n} \widehat{x_i}$ and $\sum_{i=0}^{n} \widehat{x_i^2}$ computing the discrete logarithm of base g. Now, the aggregator $\mathbb{A}$ has the statistic of participants' values, without having exact knowledge about values of single attributes.

2.2 Data Monetization Model

The obtained tuple (V_j, W_j) is used by the aggregator to estimate the probability density function of analyzed data. $\mathbb{A}$ computes the discrete logarithm base g of (V_j, W_j) to obtain values $(\widehat{\mu_j}, \widehat{\sigma_j^2})$:

$$\begin{aligned} \widehat{\mu_j} &= \frac{\log_g V_j}{N} = \frac{\sum_{i=1}^{N} \widehat{x_{ij}}}{N}, \\ \widehat{\sigma_j^2} &= \frac{\sum_{i=1}^{N} \widehat{x_{ij}^2}}{N} - \widehat{\mu_j}^2. \end{aligned}$$

Using the *Central Limit Theorem*, the authors of [3] claim that the distribution of attribute j can be approximated by the normal distribution $\mathcal{N}_j$ with parameters $\widehat{\mu_j}, \widehat{\sigma_j^2}$. This assumption is used to compute the distance d_j between the obtained normal approximation $\mathcal{N}_j$ and the uniform distribution $\mathcal{U}$. The "distance" between discretized $\mathcal{N}_j$ and $\mathcal{U}$ is measured by the *Jensen-Shannon divergence* (or JS, for short). To decide which attribute can be shared with $\mathbb{C}$, the aggregator compares d_j with users sensitivity parameters λ_{ij}. If $d_{ij} \leq 1 - \lambda_{ij}$, this means that the user i is willing to share this attribute j. $\mathbb{A}$ uses the majority rule to decide if he can offer the statistic about attribute j to $\mathbb{C}$.

Cost function and sharing the revenue: The aggregator computes the monetary price *Cost*(j) of the attribute j depending on the distance d_j, the number of the contributing users and the monetary value p_j that users assign to each attribute. $\mathbb{A}$ takes as the salary the commission percentage (fixed at the beginning of the protocol) for each attribute and splits the rest of the revenue evenly among the users.

3 Flaws, Shortcomings and Remarks

Despite clear merits of the protocol recalled in the previous section, we need to point some of its flaws and shortcomings. In this section, we also discuss some assumptions that seems to be disputable in many practical settings. Some of our remarks refer solely to the Bilogrevic at el.'s scheme (like the parts described in Subsects. 3.1 and 3.4) while some other can be applied to wider class of similar protocols (Subsects. 3.2, 3.3 and 3.5).

3.1 Information Leakage

In this section we show that the protocol from [3] can reveal significant information about the data aggregated by an individual user. Let us recall, that in order to "aggregate" values x_{ij}, x_{ij}^2 user u_i sends to the aggregator $\mathbb{A}$ the following values

$$\begin{aligned} c_{ij} &= g^{\widehat{x_{ij}}} H(t)^{s_i}, \\ c_{ij}^2 &= g^{\widehat{x_{ij}^2}} H(t)^{s_i}, \end{aligned} \tag{4}$$

as j-th elements in vectors c_i and c_j^2, respectively (see formulas (1), (2)). User u_i hopes, that the only information which $\mathbb{A}$ can learn are the sums of x_{ij} and x_{ij}^2 over significantly many i (i.e., over at least $\gamma n - 1$ other users). Moreover, any other party which might observe the communication between $\mathbb{A}$ and the users is believed to be unable to learn any partial information about values being aggregated.

Recall, that g is a public parameter. For that reason, **any** party (including $\mathbb{A}$) which intercepts values c_{ij}, c_{ij}^2 can perform the following operations

$$\begin{aligned} \frac{c_{ij}^2}{c_{ij}} &= \frac{g^{\widehat{x_{ij}^2}} H(t)^{s_i}}{g^{\widehat{x_{ij}}} H(t)^{s_i}} = \frac{g^{\widehat{x_{ij}^2}}}{g^{\widehat{x_{ij}}}} = g^{\widehat{x_{ij}^2} - \widehat{x_{ij}}}, \\ v := \log_g \left(\frac{c_{ij}^2}{c_{ij}} \right) &= \widehat{x_{ij}^2} - \widehat{x_{ij}} = x_{ij}^2 - x_{ij} + o_{ij} - r_{ij}. \end{aligned} \tag{5}$$

Let us argue, that computing a discreet logarithm is feasible. First, let us recall that values x_{ij} belong to a known and relatively small domain. Indeed, as described in previous section, each value $x_{ij} \in \{m_j, \ldots, M_j\}$, where $m_j, M_j \in \mathbf{Z}_p$ for prime p. The only problem may be caused by the values r_{ij} and o_{ij} used as a noise to "obfuscate" the real values. Authors of [3] do not explicitly specify how values r_{ij}, o_{ij} are chosen, but give a reference to the protocol described in [24]. According to [24], values r_{ij}, o_{ij} are computed using following distribution

$$r_{ij} \leftarrow \begin{cases} Geom(\alpha) & \text{with probability } \beta, \\ 0 & \text{with probability } 1-\beta \end{cases}$$

where $Geom(\alpha)$ is the *symmetric geometric distribution*. The values α and β are chosen in such a way so as to ensures the distributed differential privacy of users after revealing **aggregated** (not a single) values and depend on other parameters. Namely, $\alpha = \exp\left(\frac{\epsilon}{\Delta}\right)$ and $\beta = \frac{1}{\gamma n} \log \frac{1}{\delta}$, where $\Delta = M_j - m_j$ and $0 < \gamma < 1$ is a fraction of honest users. Finally $0 < \delta < 1$ and $\epsilon > 0$ are values governing privacy (they are used in the definition of (ϵ, δ)-differential privacy. Note, that the value of β is small even for very small δ. Since r_{ij}, o_{ij} are chosen independently then with probability at least $(1-\beta)^2$ we get $r_{ij} = o_{ij} = 0$. In such case computing a discrete logarithm of v requires at most Δ operations. Moreover, the equation is reduced to $v = x_{ij}^2 - x_{ij}$. The Fundamental Theorem of Algerbra (see [11]) implies that this equation (over $\mathbf{Z}_p$) has at most two solutions as a polynomial of variable x_{ij}.

In the case when $r_{ij} \neq 0$ or $o_{ij} \neq 0$ the equation has also at most two solutions, but it is dependent on added values (unknown for the attacker). Note however, that for a typical (discussed and simulated in [24]) parameter $\epsilon = 0.5$ and **arbitrary** δ both values r_{ij} and o_{ij} are with high probability distributed over a small subset of $\mathbf{Z}_p$. Thus the value of x_{ij} can be found with significant probability.

Note, that this attack cannot be used against the original scheme [24]. Indeed we utilize relations between $\widehat{x_{ij}^2}$ and $\widehat{x_{ij}}$, while in the protocol described in [24] users send only an encrypted version of $\widehat{x_{ij}}$.

Improving the Security. One may be tempted to repeal this problem using o_{ij}, r_{ij} from other distribution getting values from a bigger interval with high probability. Indeed, that would improve the protection of the privacy of individuals. Informally, the data would be "more obfuscated". Such a modification however would dramatically decrease the utility of the aggregated data. Moreover, retrieving the statistics by $\mathbb{A}$ (during the decryption in AGGREGATION step) would be computationally much more demanding.

More promising direction to patch the scheme might be using different t's for every single attribute value j and it's square. This protects the data from the attack described above. However, still a similar attack can be performed using relations between $c_{ij} = Enc(\widehat{x_{ij}})$ and $c_{ij'} = Enc(\widehat{x_{ij'}})$ for $j \neq j'$. Note, that we cannot use different values t for every user i and attribute j, because then $\mathbb{A}$ will not be able to decrypt the data (i.e., cannot use the property of summing the secret keys s_i to 0).

This approach would also increase the complexity of the scheme and requires $\Theta(k)$ hashings and performing additional, heavy operations (exponentiation) for both the aggregator and every single user.

3.2 Pricing via Approximation by Normal Distribution

In this section we concentrate on the method of establishing the price of the aggregated data presented in [3]. To evaluate the values of computed statistics the distribution of the data is approximated by the normal distribution $\mathcal{N} = \mathcal{N}(\widehat{\mu}, \widehat{\sigma^2})$. This approximation is motivated by *Central Limit Theorem* (CLT), which **roughly** stays that the arithmetic mean of a sufficiently large sample of independent and identically distributed random variables tends to normal distribution (with probability). Then, the aggregator computes d_j - the Jensen Shannon divergence between normal $\mathcal{N}$ and uniform distribution $\mathcal{U}$ for each attribute j. The value d_j is then used for evaluating the aggregated data (measuring how much information is released and fixing a price).

The CLT allows to estimate the distribution of the mean of a sample. However, it does not allow us to say much about the distribution of the random variables from the sample (except some cases -e.g. when we know in advance that the original data is sampled from a normal distribution). Despite common opinion, usually encountered data is not normally distributed. Hence, one should not approximate the distribution of the actual attributes using the normal distribution. In Table 1, we present the results of two standard normality tests (χ^2-square, Kolmogorov-Smirnov) conducted for a sample of size 50 from a dataset of 3348 users from DataFerret (Note, that in Subsects. 3.2, 3.3 we use the same source of data for experiments as authors of [3]). We compare the distribution of attributes: Income and Education with the normal distribution. The results show, that the distribution of a certain attribute is very far from the normal distribution. This explains, that even if the mean value is normally distributed, we cannot approximate the original distribution with $\mathcal{N}$. Plots of the values of analyzed attributes show, that the distribution function significantly deviates from the Gaussian bell curve.

Table 1. Normality tests results

		Incomes	Education
Kolmogorov-Smirnov normality test	p-value	1.067e-05	9.83e-05
	D	0.2081	0.1904
Pearson χ^2-square normality test	p-value	8.742e-08	6.106e-08
	P	46	46.8

To depict more formally, why the method presented in [3] based on normalization and JS divergence may lead to false conclusions about the distribution of investigated attributes, we present an illustrative example. In short, it shows that there are distributions[1] arbitrary far from uniform distribution, such that after approximation by the normal distribution, can be very close to uniform distribution (in the sense of JS divergence metric).

Example 1. Let $\mathcal{D}_N$ be a random variable deified in the following way $\Pr[D_N = N] = \Pr[D_N = -N] = 1/2$. Let us fix N. Directly from CLT one can prove that the arithmetic mean of a sample of size n chosen from the same distribution as D_N converges with probability for growing n to the normal distribution $\mathcal{N}(0, N^2)$. Let $\mathcal{N}_N$ be an uniform distribution on the set $\{-N, -N+1, \ldots, N-1, N\}$. One can easily prove that $\lim_{N\to\infty} JS(\mathcal{U}_N, \mathcal{N}(0, N^2)) = 0$, but in the same time $\lim_{N\to\infty} JS(D_N, \mathcal{U}_N) = 1$.

3.3 Sensitivity of the Distributions

In this section, we show that distributions of the data investigated in typical scenarios are very sensitive even to small changes. Namely, adding/changing a single factor to a query may completely change the type of the distribution of the focused data. In real-life scenarios, the customers want to obtain more precise information. For example, when $\mathbb{C}$ asks for the incomes of the users, the queries might be depend on many factors like: highest obtained education degree, state of residence, industry code, sex or race. In such case, the particular factor has significant impact on the result and the analyzed sample of users cannot be generalized to the whole data, and vice versa. Below, we present the plots of incomes of the USA citizens. The analyzed data is from Census Bureau [1]. The plots depict the incomes of sampled users, specified by selected factors in comparison with the randomly chosen sample of users. The mean value of the random sample is marked by the solid line, whereas the dashed line represents the mean values of the sample determined by a factor (Fig. 2).

Our simulations show, that attributes are very sensitive to different factors, which can sway them significantly. One can observe, that education, branch of industry and occupational field, state of living or even sometimes factors like sex and race determine personal income (Fig. 3).

We conducted this simulation to show that if a potential customer wants to obtain a specific piece of information about users and the aggregator shares the mean value of a certain attribute computed based on a whole dataset, the results will be strongly inaccurate. Note, that in the protocols, where the statistic is computed based on all users, the customers can learn only the general information which in many scenarios can be useless. For that reason, one may doubt if it could be possible to construct a single, widely accepted method of evaluating this type of data using a fixed distribution to compare the results (Figs. 4 and 5).

[1] In some formulas for the sake of simplicity we identify a random variable with its distribution.

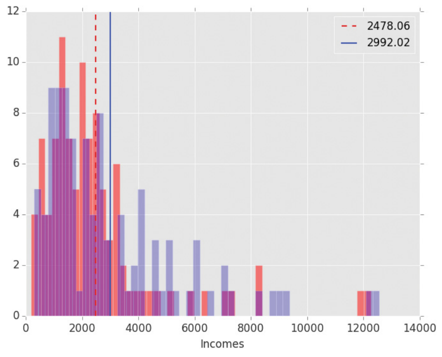


Fig. 2. Mean of incomes of the random citizens of the USA citizens and those who live in Arizona. Sample size 100.

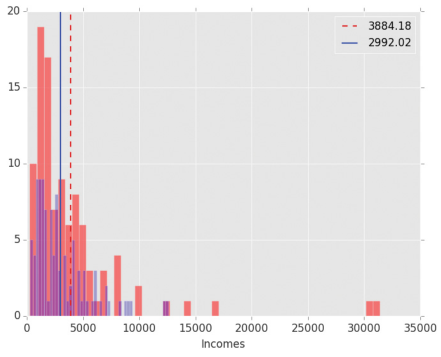


Fig. 3. Mean of incomes of the random citizens of the USA citizens and those who live in New York. Sample size 100.

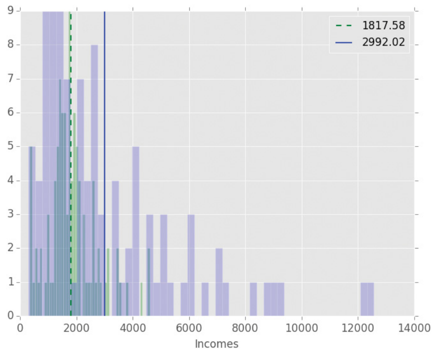


Fig. 4. Mean of incomes of the random citizens of the USA and those who graduated High School. Sample size 100.

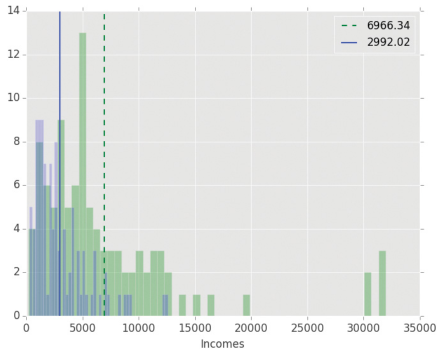


Fig. 5. Mean of incomes of the random citizens of the USA and those who graduated Master's degree. Sample size 100.

3.4 Other Shortcomings and Remarks

There are some other issues of the proposed solution that significantly limits the range of applicability of the Bilogrevic et al's protocol. Note that most of them are inherited from [24] (Figs. 6 and 7).

Key distribution necessity - it is assumed that each user u_i is given a private key s_i such that $s_1 + \ldots + s_n + s_A = 0$ (where s_A is the aggregator's key). For that reason users need a trusted third party or a key establishment protocol eliminating trusted dealer in the manner similar to [18]. The latter would require significant volume of communication/interaction. For that reason such approach seems not to be adequate for dynamic groups of users. Moreover, if at least one user declines taking part in the protocol any aggregated values cannot be obtained.

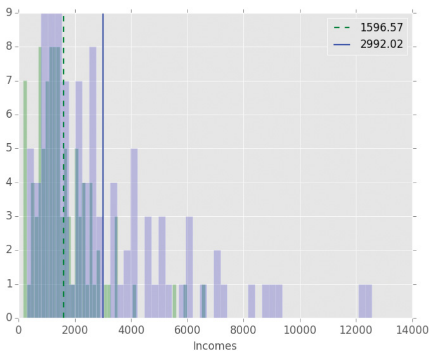


Fig. 6. Mean of incomes of the random citizens of the USA and those who work in the food service.

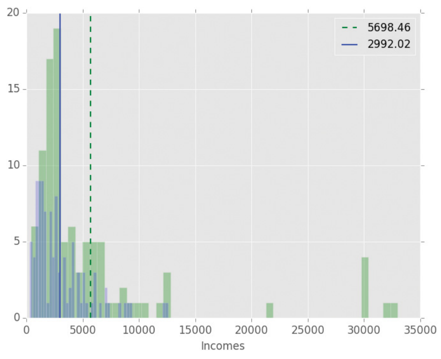


Fig. 7. Mean of incomes of the random citizens of the USA and those who work in the hospitals.

Data poisoning attack - every single user may add a fake value to completely change the final value of data aggregation. In addition, one can easily see that any user may launch a DoS-type attack by giving a huge value instead x_{ij} to make decoding procedure (3) impossible. Note, that without any special countermeasures (like zero knowledge proofs) such attacks are undetectable. On the other hand, all countermeasures we have in mind cause a significant increase of computations and/or intensive interactions between users.

Limited range of aggregated data - there is a very limited range of values that can be efficiently aggregated. If there is no knowledge about the a priori distribution of x_{ij}, the aggregator may be forced to perform the number of exponentiations that is proportional to possible size of a range of the possible data. Moreover, the scheme allows to deal with numerical values, only.

Correlation between statistics - At the end of the aggregation phase the aggregator reveals $S_1 = \sum_{i=0}^{n} \widehat{x_i^2}$ and $S_2 = \sum_{i=0}^{n} \widehat{x_i}$. The proof given in [24] guarantees privacy protection of all honest users after revealing S_1 and after revealing S_2 but not both. The problem is in the fact that S_1 and S_2 are correlated and reveal more information that each of them individually. Nevertheless, despite the lack of a formal proof of the security we doubt if S_1 and S_2 can be used for mounting any successful attack in real life settings.

3.5 To Pay for What?

Authors of [3] handled very important issue - how to evaluate (and fix a fair price) for the data and preserve privacy of individual users at the same time? This seems to be the most problematic issue related to the monetization protocols. In short, authors of [3] proposed the following strategy - the aggregated data is used for construction of an approximation $\mathcal{N}$ of the probability distribution of the collected data $\mathcal{D}$. Then, the "distance" d between $\mathcal{N}$ and uniform distribution $\mathcal{U}$ (over the same range as $\mathcal{D}$) is computed. In the protocol [3] the Jensen-Shannon divergence is used as a distance function. The final price is computed as the product $c \cdot n \cdot d$, where c is a constant depending on a type of data (see further information in [2,16,17]) and n is the number of users.

We suppose that in many cases such approach cannot be justified, let alone the fact that the way of computing $\mathcal{N}$ as an approximation of $\mathcal{D}$ presented in [3] leads to some paradoxes as described in Sect. 3.2.

– It is not clear, why the customer $\mathbb{C}$ should have an incentive to pay more, if the distance between $\mathcal{N}$ and $\mathcal{U}$ is big. In various real-life scenarios such value does **not** *estimate the amount of valuable information (i.e., sensitivity) that each attribute leaks* (as claimed in [3]). This is because of the fact that in many cases the distribution of an examined value is not expected to have the uniform distribution. Moreover, the information that those values have a statistic typical for uniform distribution can indicate a serious anomaly and information about this fact may be worthless. Thus, it is hard to agree that uniform distribution does not *reveal actionable information.*
– More natural approach seems to be paying for the "difference" (i.e. measured using JS divergence, total variation distance etc.) between the actual distribution of the aggregated data and an *a priori* distribution. That seems to be a good model of how much the customer may be surprised by the new data. On the other hand, one can easily see that confirming that the actual distribution of the data is close to its *a priori* distribution can be very valuable as well. Moreover, in the case with many types of data it is not feasible to determine easily an *a priori* distribution at hand.

One can show many examples that a value of the information (and its price) highly depends on the context. This is even more challenging in the scenarios wherein information is being processed (in particular, in order to protect users privacy). A customer wants to buy the utility of data, while a user sells the privacy. In real life systems, both privacy and utility usually strongly depend on a very complex context. Thus, we believe that the approach for a data evaluating presented in [3] can be very promising, however only for a limited range of applications.

3.6 Possible Patches

In this paragraph, we propose a set of methods which might be applied to improve the flexibility of the privacy-preserving protocol introduced in [3,24] and also prevent the information leakage presented in the previous section. We only outline the presented methods, which still need further investigation, both in terms of security and efficiency. We believe that the presented ideas can be applied in real-life scenarios to the original scheme from [3] as well as some other similar protocols. Let us recall some techniques that can be used to mitigate problems described in our paper.

Secret sharing. The main concern on which we focused is how to increase a flexibility of the previously presented protocol. One of the possible solutions might be the use of the secret sharing introduced by [23] to allow the customer to ask for a more accurate and better specified group of users and does not rely on the availability of all of them.

First, let as recall the idea of the (k,n)-threshold secret sharing. Having a secret S, one can "divide" it into n *shares* $Z = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$. The (k,n)-threshold secret sharing ensures that if one knows at least k out of n values from the set Z, the value of secret S can be easily computed. However, knowledge of any $k-1$ or less shares does not give any information about S.

Distributed key generation. The users generate a pair of keys (K, D) using a distributed key generation schema, which does not rely on the trusted third party. This can be one of MPC (Multiparty Computation - type) protocols.

Additional layer of encryption. Another issue is how to prevent the information leakage described previously. One of the possible solutions might be to add an extra layer of encryption by using a random noise generated from the uniform distribution $\mathcal{U}$ - such data appears to the adversary purely random. If such layer is removed, thus it does not influence the error of aggregated data.

These three methods can be applied to the privacy-preserving protocol presented in previous sections. As previously, we consider a protocol with a set of n users $\mathbb{U} = \{u_1, \ldots, u_n\}$, an untrusted aggregator $\mathbb{A}$, who is now responsible only for joining the data received from the user into one (encrypted) statistic and a customer $\mathbb{C}$. Additionally, we have also the auxiliary servers $\mathbb{S}$ and $\mathbb{P}$ and the data trader $\mathbb{T}$, who is responsible for forwarding the query to the users, pricing the statistical information and sharing the revenue for users. Servers $\mathbb{S}$ and $\mathbb{P}$ gather the information from the users which is necessary to decrypt the obtained data.

After the customer $\mathbb{C}$ sends a query to $\mathbb{T}$ with the information about selected users and data which he wants to obtain, $\mathbb{T}$ forwards the customer's query to all users. After receiving the query, users generate a pair of keys (K, D) using the distributed key generation process. Each user has its own key pair (K, ζ_i). Public key K is used to encrypt the individuals information. Those users, who want to reveal their private information, by responding to this query, obfuscate, encrypt their data with key K and add one more secure layer by using a random noise generated z_{ij} from the uniform distribution $\mathcal{U}$ as below

$$c_{ij} = Enc_K(x_{ij} + r_{ij}) \cdot b^{z_{ij}}, \tag{6}$$

where b is a public parameter. Next, each user u_i sends its secured data to the aggregator $\mathbb{A}$. Value ζ_i is forwarded to server $\mathbb{S}$ and the noise value z_{ij} to $\mathbb{P}$. Aggregator $\mathbb{A}$ combines the encrypted data received from the users and forwards the obtained statistic (for example: sum) to the data trader $\mathbb{T}$. Server $\mathbb{P}$ computes a sum of all values from the users who respond to the query as $\sum_{i=1} z_{ij}$ and forwards this sum to $\mathbb{T}$. At the same time, server $\mathbb{S}$ forwards to $\mathbb{T}$ permuted values ζ_i, which he received from the users. After receiving messages from the aggregator and the servers $\mathbb{S}$ and $\mathbb{P}$, the data trader $\mathbb{T}$ is able to remove the additional layer of encryption from the encrypted statistic by dividing the value by $b^{\sum_{i=1} z_{ij}}$ and rebuild D (based on the secret sharing schema) to decrypt the obtained statistics.

The main advantage of this type approach is the fact that the protocol can be successfully completed even if some users cannot (or do not want) to participate in the data aggregation process. On the other hand, if the statistics are finally released and sold to the customer, every single user can be sure that its data is aggregated with at least $k-1$ other values. The users can decide after each query how large the parameter k should be to ensure that the sensitive data is protected. Let us note that the security parameters should depend on the threshold parameter k.[2] Clearly, the privacy of users is protected as long as other actors do not cooperate together against them.

The main drawback of this model is an increased amount of communication and computations. However, we believe that further investigations can improve this flaw and bring promising results.

4 Conclusion

We believe that constructing a useful and privacy-oriented method of data monetization still remains a big challenge.

In our paper, we pointed some shortcomings of monetization scheme from [3] and proposed a different one. Clearly, our approach is still not adequate for many realistic scenarios. However, it is hard to imagine a single "universal" method of monetization of statistics of users' data that can meet all reasonable requirements. In particular, finding the price of processed (for protecting privacy) data seems to be the most difficult issue that strongly depends on many unpredictable factors.

We hope that some of our remarks can be used for constructing different privacy-preserving monetization protocols.

Acknowledgments. We would like to thank the anonymous referees for their very important comments and for providing directions for further work.

References

1. U.s. census bureau. dataferrett analysis and extraction tool
2. Aperjis, C., Huberman, B.A.: A market for unbiased private data: Paying individuals according to their privacy attitudes. CoRR (2012)
3. Bilogrevic, I., Freudiger, J., De Cristofaro, E., Uzun, E.: What's the Gist? privacy-preserving aggregation of user profiles. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 128–145. Springer, Heidelberg (2014)
4. Bilogrevic, I., Freudiger, J., Cristofaro, E.D., Uzun, E.: What's the gist? privacy-preserving aggregation of user profiles. IACR Cryptology ePrint Archive, 2014:502 (2014)

[2] Value k/n has similar influence on privacy as the parameter γ in [24].

5. Chan, T.-H.H., Shi, E., Song, D.: Privacy-Preserving Stream Aggregation with Fault Tolerance. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 200–214. Springer, Heidelberg (2012)
6. Chen, R., Reznichenko, A., Francis, P., Gehrke, J.: Towards statistical queries over distributed private user data. In: NSDI (2012)
7. Cynthia, D., Krishnaram, K., Frank, M., Ilya, M., Moni, N.: Our data, ourselves: Privacy via distributed noise generation. In: Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques (2006)
8. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
10. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
11. Fine, B., Rosenberger, G.: The Fundamental Theorem of Algebra. Springer, New York (1997)
12. Hu, L., Evens, D.: Secure aggregation for wireless networks. In: SAINT-W 2003 (2003)
13. Jose, J., Jose, J., Princy, M.: A survey on privacy preserving data aggregation protocols for wireless sensor networks. J. Comput. Inf. Technol. (2014)
14. Klonowski, M., Koza, M., Kutylowski, M.: Efficient and robust data aggregation using untrusted infrastructure. In: Elçi, A., Gaur, M.S., Orgun, M.A., Makarevich, O.B. (eds.) The 6th International Conference on Security of Information and Networks, SIN 2013, Aksaray, Turkey, November 26–28, pp. 123–130. ACM (2013)
15. Krishnamachari, B., Estrin, D., Wicker, S.B.: The impact of data aggregation in wireless sensor networks. In: 22nd International Conference on Distributed Computing Systems, Workshops (ICDCSW 2002) July 2–5, Vienna, Austria, Proceedings, pp. 575–578. IEEE Computer Society (2002)
16. Malheiros, M., Preibusch, S., Sasse, M.A.: "Fairly Truthful": the impact of perceived effort, fairness, relevance, and sensitivity on personal data disclosure. In: Huth, M., Asokan, N., Čapkun, S., Flechais, I., Coles-Kemp, L. (eds.) TRUST 2013. LNCS, vol. 7904, pp. 250–266. Springer, Heidelberg (2013)
17. Pablo, C.J., Christopher, R., Vijay, E., Mauro, C., de Oliveira, R.: Your browsing behavior for a big mac: Economics of personal information online. In: Proceedings of the 22Nd International Conference on World Wide Web, WWW 2013 (2013)
18. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
19. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010 (2010)
20. Riederer, C., Erramilli, V., Chaintreau, A., Krishnamurthy, B., Rodriguez, P.: For sale: Your data: By: You. In: Proceedings of the 10th ACM Workshop on Hot Topics in Networks (2011)
21. Rieffel, E.G., Biehl, J.T., van Melle, W., Lee, A.J.: Secured histories: computing group statistics on encrypted data while preserving individual privacy. CoRR, abs/1012.2152 (2010)

22. Roy, S., Conti, M., Setia, S., Jajodia, S.: Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact. IEEE Trans. Inf. Forensics Secur. **9**(4), 681–694 (2014)
23. Shamir, A.: How to share a secret. Commun. ACM **22**, 612–613 (1979)
24. Shi, E., Chan, T.-HH., Rieffel, E., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS (2011)
25. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: Wang, J.T. (ed.) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD, Vancouver, BC, Canada, June 10–12, 2008, pp. 567–580. ACM (2008)

A Novel Approach to Data Revocation on the Internet

Olga Kieselmann(✉), Nils Kopal, and Arno Wacker

Applied Information Security, University of Kassel,
Pfannkuchstr. 1, 34121 Kassel, Germany
{olga.kieselmann,nils.kopal,arno.wacker}@uni-kassel.de
https://ais.uni-kassel.de

Abstract. After publishing data on the Internet, the data publisher loses control over it. However, there are several situations where it is desirable to remove published information. To support this, the European Union proposed the General Data Protection Regulation (GDPR) which states that providers must remove the data when the corresponding owner requests it. However, the data might already have been copied by third parties. Therefore, Article 17 of the GDPR includes the regulation that the provider must also inform all third parties about the users request. Hence, the providers would need to track every access, which is hard to achieve. This technical infeasibility is a gap between the legislation and the current technical possibilities. To close this gap, we propose a novel service which gives the data owner the possibility to inform simultaneously all providers about her removal request.

Keywords: Privacy · Data revocation · GDPR · Internet service

1 Introduction

The vast variety of different Internet services existing today entails that people knowingly or unknowingly share more and more personal data. Until recently, most people had no problem with loosing control over the processing and dissemination of their own data once published on the Internet. Lately, however, this attitude has begun to change and Internet users start to exercise more caution about which data they share. The main reasons are the recently disclosed privacy incursions by government agencies and companies. Internet users became aware of massive surveillance and data storage programs. Furthermore, there were also recent revelations that some service providers misuse private user data [23].

Besides that, people are emotional beings and it is typical for a person to change her opinion and views over the course of her life. This is also true for data published on the Internet. As she changes her opinion over time, she might regret having published data which does not reflect her current opinion. Therefore, she wants to extinguish this data from the Internet in the present. For instance, she might have published a party-picture on a social network in her youth.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 134–149, 2016.
DOI: 10.1007/978-3-319-29883-2_9

At that time, she did not care much about her public appearance. Years later, after finishing her studies, she applies for a job in a leading position. While her opinion and attitude changed over time, the published picture did not and can still be found. Even worse, it might influence the decision of the recruiting agent. To prevent this, she would like to completely remove this picture from the Internet. However, today it is almost impossible to identify all locations and servers where the picture has already been copied and stored – she has lost control over her own data.

In general, Internet users want more control over their data. An example for this is the impact from the judgment by the Court of Justice of the European Union in case 131/12 from May, 2014 [22]. This case states that natural persons have the right to opt out from the results of search engines. Since this judgment was published, Google has to deal with thousands of user requests for deletion every day [14]. Although removing links from search results does not actually delete the data, people use this possibility due to the lack of better solutions.

Since 2011, the European Commission is working on a General Data Protection Regulation (GDPR) which will be mandatory for all countries in the European Union (EU). It will also be binding for foreign companies which have their subsidiaries in the EU. The GDPR literally states that *"the proposed data protection regulation leaves no legal doubt that no matter where the physical server of a company processing data is located, non-European companies, when offering services to European consumers, must apply European rules"* [5].

The principal aim of this regulation is underpinning the right of individuals to have their data deleted on the Internet. A draft of the proposed GDPR was already published in 2012 [6]. However, the technical feasibility of this draft was questioned by many experts. For example, in [9], the authors considered the technical means to enforce this right in information systems and identified technical challenges. In [17], we also discussed problems for service providers with respect to this draft. Specifically, according to Article 17, service providers must *"take 'reasonable steps', including technical, to inform third parties of the fact the individual wants the data to be deleted"*. Thus, the main challenge for the service providers is to track every access in order to inform all third parties. However, there is currently no technical solution to fulfill this legal requirement with reasonable effort.

Fully deleting data on the Internet is a complex issue. Therefore, we propose to divide it into sub-problems, which we can solve independently. We start in this paper with a solution only for the case where a user publishes data objects on the Internet self-willed and consciously. Hence, we don't consider cases where third parties publish or collect data about a user. For this, we present an Internet-wide data revocation service in this paper. With this service, on the one hand, the service providers can automatically verify whether there are removal requests. On the other hand, the user can notify all service providers that her data objects should be deleted. Thus, our solution offer both parties a technical instrument to support Article 17.

The rest of this paper is organized as follows: We first discuss existing approaches for removing previously published data from the Internet in Sect. 2. In Sect. 3, we present our assumptions, and in Sect. 4 we motivate our requirements for a technical data revocation solution. Based on those, we discuss the design rationale of our approach in Sect. 5 and its implementation possibilities in Sect. 6. Afterwards, we analyze our approach with respect to our requirements. Finally, we conclude our paper in Sect. 8 with a brief summary and provide an outlook on future research.

2 Related Work

In the past, there have been several technical approaches for controlling the availability of data objects after publishing them on the Internet. The ideal solution are self-destructing data objects which disappear or become useless after a specified time or on owner's demand. The main challenge with such self-destructing data objects is the hostile host problem [16]. Generally, the hostile host problem states that it is theoretically impossible to force a foreign host into performing specific actions if its administrative owner disagrees and interferes. Despite this theoretical impossibility, there are practical solutions which make it very hard for the administrative owner to interfere. The more secure ones rely on additional tamper resistant hardware like High Security Modules (HSMs) or Smartcards.

In general, all those solutions follow the same principle: first, the protected data object is encrypted. Then, the encrypted data object is published, and the key is handed to a key management service. Anyone who wants access to the data object must get this key. When the data object is to be "deleted", the key is removed from the key management service. This way, it is no longer possible to decrypt the data object, and thus the object has been effectively deleted. The critical parts for this solution are the key and the (decrypted) representation of the data object. Anyone who can intercept the key has no need to request the key again and thus can decrypt the data object at any time in the future. The protection can also be circumvented if it is possible to make a copy of the actual data object while it is decrypted. The most known example based on this principle are Digital Rights Management (DRM) systems [19]. Even though some of the DRM systems rely on tamper resistant hardware, they get circumvented eventually – it is usually just a matter of time.

While DRM systems usually aim for protecting content owned by industry, there are similar systems for arbitrary users. There is X-pire! [2] which is a pure software solution for protecting pictures on the Internet. The decryption is done with a web browser add-on. As the browser environment isn't protected from its own administrator, it took only a short time for the first successful attack against it [24]. To solve this problem in X-pire 2.0, the developers require a trusted execution environment (TEE) for requesting and storing the decryption keys as well as for displaying the decrypted content [3]. Hence, this approach is limited to TEE-enabled devices. A similar concept are "self-destructing" data objects in Vanish [12] and EphPub [4]. In Vanish, the data object is also encrypted, but the key is stored in a distributed hash table (DHT). This key gets lost after some time, and the data automatically becomes inaccessible. However, Wolchok et al. showed in [29] that the design of Vanish is insecure and can be circumvented even under the stricter assumptions made by the Vanish authors. To overcome the attacks on Vanish, EphPub uses the world-wide domain name system (DNS) [21] to store the key. A common aspect of those approaches is that they are designed to prevent access to data objects after expiration time. Especially, they do not offer DRM-like mechanisms which prohibits anyone from copying or republishing content before the expiration time. Thus, there is an implicit assumption that also providers do not circumvent the system before a data object is expired. Additionally, to decrypt the protected data objects, they all require some software installed on the users computer (e.g. browser extension). In our approach, we use the same assumptions to protect data objects after their expiration time. However, we do not use any encryption, but rely on the cooperation of the providers. This reduces the complexity of our approach and does not require any changes on the computer of any Internet user (e.g. no browser extension required).

A different approach for controlling the availability of data objects is practiced by Internet search engines. Nowadays, we find most of our information on the Internet with the help of search engines. Thus, if some data objects (here mostly websites) are removed from the result list of search engines, they can no longer be found by most Internet users. Google currently uses this approach to "delete" websites on request. The drawbacks of this approach are twofold: first, the search engine provider must manually verify each request and decide according to the local law if a "deletion" is acceptable or not. Second, even if the search provider removes the site from its result list, the site itself does still exist and can be accessed with the correct URL. Even worse, it might still be found by a search engine not obeying this law.

In summary, the existing attempts for deleting data objects on the Internet offer neither a real protection, nor a possibility to prevent the data propagation.

3 System Model

Nowadays, arbitrary data is easily spread on the Internet. On the one hand, there are data *owners* who publish their data objects with one or more *providers*. Hence, publishing a data object determines its owner. Within our system, other persons have no claim of ownership for this data object. A *data object* can be an arbitrary information, e.g., a picture, a video file or a document. Providers offer the published data through different services. On the other hand, there are arbitrary Internet-*users* who retrieve this published data with service-specific clients, e.g., web browsers. Furthermore, the published data might also be replicated by providers. Usually, different providers are under different adminstration controls, i.e., one provider cannot control the stored data of another.

Similarly to the real world, there are also legal regulations on the Internet. In particular, the proposed GDPR of the EU regulates the right of an owner to delete her previously published data. More specifically, the providers are required to delete data objects upon request of the owner. We, therefore, assume that all providers bound to this law within the EU act correctly and delete the data upon request. Providers outside of this law's jurisdiction might not obey it. Furthermore, users and owners might act maliciously – independently of any legal regulations. For instance, some users might try to steal data from data owners and distribute it. Even more, they might assume ownership of the stolen data.

4 Requirements

In the following, we describe and motivate the requirements for our data revocation system.

Availability: The system must allow owners to revoke their own data on the Internet at any time, i.e., the revocation system must be able to process requests for data revocation from owners at any time. It is acceptable that the actual data revocation through the providers is performed at a later time. This time can be a fixed timespan or whenever a user requests the respective data object in the future.

Scalability: A data revocation system must scale with respect to the number of data objects under its control. We assume that in the future almost all data objects on the Internet are under the control of this system. Therefore, it must cope with a huge amount of data objects.

Usability: One of the most important acceptance factors of this system is the usability for all participants. Ideally, there is no change for any user accessing a service on the Internet. Additionally, the burden for the data owner must be kept to a bare minimum when publishing new data objects. The same must be true for the provider with respect to software upgrades and maintenance. Finally, also the system itself should work with a minimum of resources, financial as well as human. In ideal case, the system works out-of-the-box with zero configuration.

No Censorship: We require that our system does not provide a way for censorship, e.g. by government authorities. This means that only the data owner herself is able to delete her own data – and nobody else. Thus, it should not be possible for governments, even with legal jurisdiction over the providers, to use our service for selectively revoking data objects.

Privacy: A crucial requirement is to protect the user's privacy and anonymity on the Internet. Specifically, it must be impossible to deduce the owner from the data objects with our system, i.e., our system must not introduce *new* ways to deduce the owner of a data object. Clearly, if the owner is already known to the provider or can be deduced from the content of the data object, our system cannot (and is not intended to) prevent this. Furthermore, the system must not provide a way for any entity to find all data objects of a certain owner. In general, our system must not introduce *new* privacy risks for anyone.

5 Design Rationale

Due to the hostile host problem [16], the data owner is unable to delete data objects on the hosts of the providers. However, the providers are required by law to delete the data objects on demand of the owner. Additionally, the provider is required to inform all other parties who retrieved the owner's data object in the past. However, due to the nature of the Internet, it is usually very hard for the providers to identify all other parties who accessed the respective data object. Hence, the main issue for applying the GDPR is the notification of other parties about the owner's demands. In our approach, we propose to solve this issue with a distributed service which notifies every party (e.g., other providers) about the owner's demands.

The main idea of our approach is an Internet-wide data revocation service, similar to the Online Certificate Status Protocol (OCSP) [11] but for arbitrary data objects. To achieve this, we assign a unique identifier to each protected data object. We use this identifier as a unique reference to this data object within the data revocation service. To delete her own data objects, the owner notifies this service about her demands. Every provider publishing such a protected data object is required to check with the data revocation service whether the data object can still be published or must be deleted.

To ensure that only the owner is able to revoke her data objects, we require the data revocation service to resemble a distributed database without a central authority. Hence, no single company has exclusive control, i.e., the ability to revoke arbitrary data objects willfully. Furthermore, only the owner must be able to revoke her own data objects. Thus, we require an authentication mechanism for the owner which complies with our privacy-requirement.

In general, our approach works as follows (for details, see Sect. 6): Before publishing a protected data object, the owner embeds a unique identifier into the data object and registers this identifier with the data revocation service. During the registration, she sets the status of this data object to "active". The owner (and only she) can change this status later to "revoked". Afterwards, she publishes the data object using an arbitrary service provider. For any protected data object, the service provider must request its status from the data revocation service before delivering the data object to the user. As long as a data object is not revoked, it may be delivered by the provider. In case the status of a protected data object is "revoked", the provider must not deliver this data object. Additionally, the provider must delete the data object from her own data storage if this data storage is in her own administrative domain.

Furthermore, we assume that the system participants behave within the law, i.e., they do not remove the unique identifiers from protected data. Thus, our goal is not to technically enforce the usage of our service or to delete data. Rather, we rely on law enforcement to use our service and provide a mechanism for the law-abiding participants in order to follow Article 17 of the GDRM.

6 Data Revocation

To implement our approach, we suggest to build the data revocation as an Internet service, i.e., the *Data Revocation Service (DRS)*. Besides the DRS, the unique identifier for data objects and the owner authentication with DRS are significant components in our approach. Hence, we first describe these base components of our system and afterwards we provide details of the underlying protocol.

6.1 Unique Identifier

In our system, we assign to each protected data object a unique identifier (*ID*). This *ID* must be embedded in the data object to allow the service providers to identify protected data objects. For example, this can be achieved by adding the *ID* as meta-information to the corresponding data object. For instance with images, the *ID* could simply be another tag in the already available exchangeable image file format (Exif) data of the image [1]. Another example is the integration of the *ID* as an additional meta-tag in the header of a website.

A simple possibility to implement a unique *ID* would be to take a sufficient large *ID* space, e.g., 512 bits, and choose for each data object a random value from this space. With such a large *ID* space, the probability of a collision, i.e., two data objects with the same *ID*, is very low. This method with randomized IDs is used, for instance, for peer-IDs in Peer-to-Peer (P2P) networks. However, with this approach, the *ID* is independent of the data object's content. This results in the possibility to assign different IDs to the same data object. We, therefore, choose to derive the *ID* directly from the content of the data object, e.g., by hashing its contents. In order to fulfill the required uniqueness property of the *ID*, we can use a cryptographic hash function. With a cryptographic hash function, a single changed bit in the data object would yield a different *ID*. While this is the intended behaviour in most cases, there are situations when similar data objects should be identified with similar or even identical IDs. This applies for instance to images which might have been modified. In this specific case, we can use a robust hash function [26] to identify all modified derivatives of an image with the same *ID*.

By deriving the *ID* directly from the data contents, we can avoid adding it as additional meta-information to the data object. However, the service provider still needs to be able to distinguish between protected and unprotected data objects. Therefore, we still must add meta-information to the protected data object. In this case, the meta-information can simply be a flag or a set of instructions which define the algorithm for deriving the *ID*.

6.2 Data Revocation Service

As already mentioned, we require the DRS to resemble some kind of distributed database which scales to a multitude of simultaneous requests. This could be achieved by using a hierarchical structure, similar to the well-known DNS [21] for the Internet. The drawback of this approach is that such a system usually has to be set up and maintained manually by administrators. In contrast, we envision a system which can be started with zero configuration and it maintains itself in a self-organizing way. In general, P2P networks are known to be self-organizing and have the potential to cope with a multitude of requests. Thus, we propose to realize the DRS as a DHT based on a structured P2P network, e.g., Chord [25] or Kademlia [20]. Although DHTs are no real distributed databases, their functionality is sufficient for our system. Generally, DHTs offer the following two operations:

– put (key, value) - for adding a key-value pair or modifying the value of a certain key,
– get (key) - for retrieving the value for a certain key.

In our system, the *ID* of a data object is the key for the DHT, and the status of this object is stored under the corresponding value. Specifically, the owner of a data object uses the put operation to store or modify the status of a protected data object in the DHT. Afterwards, the service provider uses the get operation to retrieve this status when the corresponding data object is requested.

With a DHT, the responsibility for the set of all keys is distributed among all participating peers. Hence, each peer is responsible for a certain subset of keys. However, this means that the values of those keys could be manipulated arbitrarily by the responsible peer. In an open system, we must consider that some of the peers might act maliciously by manipulating the values under their control. We, thus, require that for each key there are k responsible peers. Hence, each operation (put, get) always operates on k peers. If the get operation retrieves different values, we use majority voting. With this approach, an attacker would need at least $\lfloor k/2 + 1 \rfloor$ subverted peers to successfully alter a value. We used this mechanism to create an access control for DHTs in [27].

6.3 Authentication

In our system, only the owner of a data object must be able to change its status. Hence, we require an access control mechanism for accessing the DRS. More specifically, we must protect only the put operation of the DHT since only the owner must be allowed to update the data object's status. In contrast, the get operation does not require any access control, as everyone is allowed to readout the status of a data object.

We can achieve such an access control in different ways. The simplest approach is to authenticate the owner with username and password. In this case, each user who wants to publish protected data objects must first register with the DRS. However, this would be a contradiction to our anonymity requirement as it identifies the owner of a data object. Even if we use pseudonyms as usernames, the DRS would still be able to identify all data objects of the same user. Therefore, this approach is not a suitable access control mechanism for the DRS.

Another possibility is the usage of shared keys. In this case, the owner stores a different shared secret for each data object registered with the DRS. For all subsequent put operations regarding a specific data object, the DRS needs to verify whether the user provides the correct shared secret. Instead of storing the shared secret itself, it would be enough to store its cryptographic hash value – similar to storing a password when using authentication with username and password. However, with this approach malicious peers have access to the shared secret and could steal the ownership of this data object. Those peers can impersonate the owner and change the status of the corresponding data object at will.

We can achieve the owner authentication with public-key cryptography similarly to our previously published approach for user authentication with a DHT [27]. To do so, the owner has to create a public/private key pair for each protected data object registered with the DRS. She stores the public key corresponding to a protected data object with the DRS and keeps the private key as her secret. For a subsequent authentication, the owner needs to sign all put operations with her private key. Thus, the DRS can verify this signature with the public key stored for a specific data object. This way, both issues from the first two approaches can be solved: By using different public/private key pairs for each data object, the anonymity of the owner is preserved, and the DRS has no possibility to identify all data objects of the same user. Furthermore, individual malicious peers from the DRS have no way to steal the ownership of a data object, as they do not have access to the private key. Obviously, this only works if the responsibility for keys in the DRS resides with many different peers, i.e., k as described in the previous section.

An alternative to public-key cryptography is the usage of a zero-knowledge proof [13]. A zero-knowledge proof is a challenge-response proof system for two parties, where the prover proves to a verifier that he posses knowledge about a secret without revealing any information about this secret. As a specific zero-knowledge proof, we can use the Feige-Fiat-Shamir protocol [10] to authenticate the owner. This protocol consists of two phases, namely the key generation phase and the authentication phase. In our system, the key generation phase corresponds to the data object registration with the DRS, and the authentication phase corresponds to the owner authentication. Furthermore, the prover is the data owner and the verifier is the DRS. To register a protected data object using the Feige-Fiat-Shamir protocol, the owner creates a large random number and stores the square of this number with the DRS. The random number itself is the owner's secret S. For subsequent authentication, the owner has to use S for responding to several challenges from the DRS. She is authenticated after responding correctly to all of them. The main advantage of this solution is the lightweight registration of a protected data object with the DRS, i.e., the user only needs to generate a random number and square it. In comparison, generating a public/private key pair for any public key algorithm is more complex. The downside of using a zero-knowledge proof is the authentication phase which is more complex in comparison with a public key algorithm. With a zero-knowledge proof, the owner is required to respond to several challenges before the DRS accepts the user as authenticated. However, we expect that this is a sound trade-off since registering protected data objects will happen many times more than revoking data objects. Therefore, we propose to use the zero-knowledge proof for authenticating the data owner with the DRS.

6.4 Protocol

We use SSL/TLS [8] to secure all communications with the DRS. This prevents man-in-the-middle attacks during the registration of new data objects. Obviously, we cannot enforce encryption for the communication between the Internet user and the provider.

We depict the message flow of our protocol in Fig. 1. Generally, we can divide it in three phases: the registration phase, the publication phase, and the usage phase. Furthermore, we can split the usage phase in two processes, namely the distribution of a data object and updating its status. In the following, we describe each phase briefly.

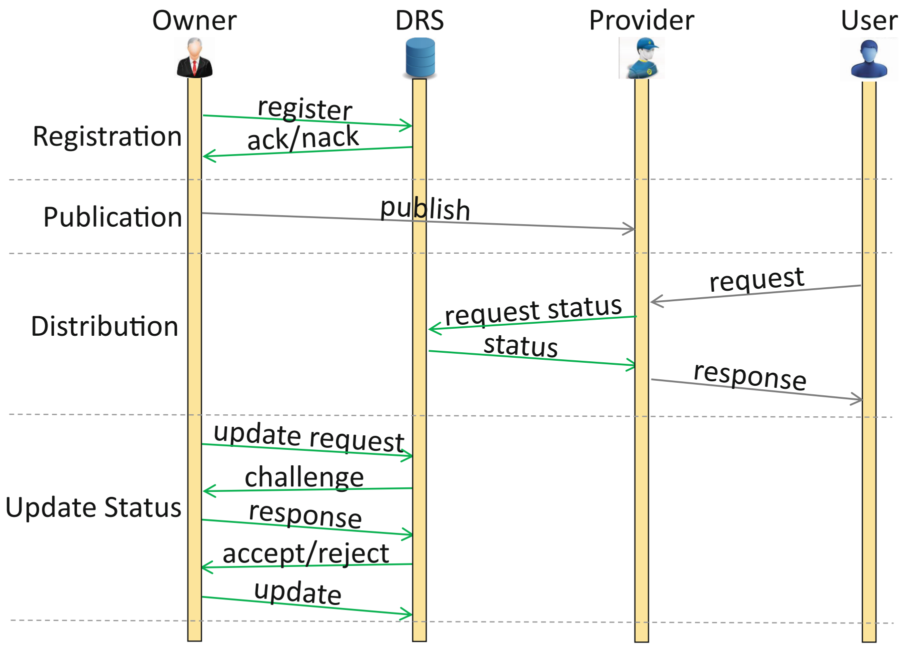


Fig. 1. Protocol overview

Registration Phase: In this phase, the owner registers a protected data object by storing a dataset with the DRS. This dataset is identified by the data object's *ID* and consists of its status (*stat*) and the so-called authenticator (*auth*) which is the square number used for the owner authentication with a zero-knowledge proof (cf. Sect. 6.3). Hence, the owner executes "put(*ID*, *stat*, *auth*)". Upon receiving this put operation, the DRS verifies whether there is already a dataset stored with the provided *ID*. If for this *ID* there is already a value present, the registration is rejected. Otherwise, the DRS completes the registration by storing the new dataset under the given *ID*.

This results in a fully anonymous registration as no information from the stored dataset includes any personal information about the corresponding owner. First, the *ID* is independent from its owner. Second, the value *stat* could be a flag determining whether the provider should deliver the corresponding object or not. Its value could simply be 'active' or 'revoked' and it should be set initially to 'active' – otherwise it would not make sense to publish the data object in the first place. Alternatively, the status could be an expiry date specifying how long the requested data object may be delivered. Finally, the value *auth* is chosen randomly by the owner and, therefore, it also does not provide any information about her.

Publication Phase: After registering a new data object with the DRS, the owner must embed the metainformation about the *ID* into the data object. Afterwards, she can publish this protected data object with an arbitrary provider on the Internet. To do so, she is free to use any way offered by the provider, i.e., this is independent from our data revocation service.

Usage Phase: As mentioned above, during this phase, we differentiate between the distribution of a data object by the provider and updating its status by the owner.

Whenever a user requests a protected data object, the provider has to retrieve the status for this object from the DRS. Depending on the retrieved status, the provider delivers the requested data object to the user or not. This is the most expensive part of our approach with respect to the number of requests. Thus, we designed the DRS as a P2P network with high scalability in mind – as described in Sect. 6.2. Additionally, the provider could cache status information for some time. During this caching time, the provider does not need to contact the DRS, thereby reducing the number of requests. However, the provider might deliver already revoked data objects during the caching time.

The second process of the usage phase is updating the status of a data object. For this, the requester must prove to the DRS that she is the owner of the corresponding data object. To do so, she has to use the zero-knowledge proof as described in Sect. 6.3 to authenticate the subsequent put-operation.

7 Discussion

In general, a P2P network is available as long as there are enough peers online. Hence, assuming enough volunteers (i.e., peers) to support our systems, the data owner can access the DHT at any time for changing the status of her data objects or registering new data objects. Furthermore, also providers can access the DHT at any time in order to verify the status of a data object.

Similarly, the requirement for a scalable service can be fulfilled by the underlying P2P network. The bottleneck of our approach are the status requests for protected data objects from the provider to the DRS. Those requests must be processed and answered by the DRS within a few milliseconds, since they directly influence the delay experienced by the Internet user. A P2P network can be built in such a way that often retrieved status is automatically cached on many peers and can therefore be delivered quickly (e.g., [7,15,28]). Additionally, by using an expiry date as the data objects' status (cf. Sect. 6.4), the provider can cache the result and only retrieve it again after the specified caching time. This reduces the amount of requests substantially. Therefore, responses within a few milliseconds are possible, even under heavy load. Furthermore, P2P networks gracefully scale with the number of peers. Hence, any number of requests can be handled if there are enough peers online.

We also achieved a high usability of our system: First, by using a P2P network for the DRS, we can keep the configuration effort for setting up this service to a minimum. In most cases, this is just a matter of installing the software on a peer and executing it. The self-organizing nature of the P2P network makes it possible to start such a service with nearly zero configuration. Second, the burden on the owner can be kept to minimum. The data owner has to store a secret S for each published data object. Additionally, she has to generate the unique ID and embed this into the data object. However, the entire process can be automated by a software running on her computer. Even easier, this could be solved with a browser plugin: whenever she uploads some data object, the plugin transparently performs all necessary steps for her. One might argue, that storing a secret for each protected data object still poses a burden for its owner. However, this can be mitigated by deriving the secret from the data object ID and a randomly generated master key K (e.g., a 256-bit key). In this case, the owner uses a hash-based message authentication code (e.g., HMAC [18]) with the master key and the data object ID to generate the required secret, i.e., $S = \mathrm{HMAC}_K(ID)$. By doing so, all secrets can be derived from the master key, hence only the master key needs to be stored. To prevent data loss, this master key can be stored – in an encrypted form – on some cloud storage, or even with the DRS itself. Finally, there is no burden at all for the common Internet user – she just browses the Internet as before, without the need to use additional hardware or install any additional software.

With our system, only the data owner in possession of the correct secret can trigger the removal of a data object from all providers who serve it. Assuming that the DRS is distributed worldwide, it is also impossible for governments or agencies to trigger the removal of specific data objects, i.e., to use our DRS for censorship. The peers responsible for the status information of a data object might not be located within the jurisdiction of a certain government. Since we require that each status information is always stored on k distinct peers, any government would need to gain access to at least $\lfloor k/2 + 1 \rfloor$ peers. This can be made arbitrary difficult by increasing k. Clearly, a government could contact the providers within their jurisdiction directly and request certain data to be removed. However, this is outside the scope of our approach as it can also be done today without our service. Thus, our service does not introduce any new means for censorship of data objects.

Finally, our service does not import any new privacy risks. Since the owner is authenticated by the means of a secret random number, this does not leak any information about the owners identity. Additionally, it is impossible to find or identify all objects from the same owner since the owner uses a different secret for each data object. Thus, the public available information stored in the DRS does not provide any new information.

Note that our system does not prevent any malicious entity to download a protected data object, remove the *ID* and to re-upload it. Since there is no *ID* with this "new" data object, the provider will not request its status from the DRS and thus will not delete it on the original owner's request. In this case, the original owner must contact the provider directly and use the classical way to delete her data object – if necessary with a court order based on the GPDR. Even though there will be cases in which our system is circumvented by malicious users, it will reduce the amount of manual requests for data removal for any provider.

8 Conclusion and Outlook

Nowadays, there is a great demand from privacy-aware users for technical solutions to remove previously published data objects from the Internet. This is complemented by the proposed GDPR of the EU which regulates that any provider must delete data objects on owner's demand. Even more, the GDPR demands that providers must also inform all third parties who accessed those data objects. However, due to the openness of the Internet, it is in general very hard for the providers to track every access to all data objects. Hence, providers might simply be unaware of all parties who may have stored copies. This makes it impossible for them to comply with the regulation.

We additionally argued in this paper that self-destructing data objects are a theoretical impossibility due to the hostile host problem. This means, there is no reliable way for a data owner to exercise any control over her data object once it is under the administrative control of one or more providers. This implies that data owners always require the cooperation of the providers. Assuming that the GDPR becomes effective, we can infer that at least the providers in the EU will cooperate. Hence, it is sufficient that the data owner reliably informs all providers about her removal request. We, therefore, proposed the Data Revocation Service as a new possibility for data owners to simultaneously inform all providers whether her data objects can still be used or must be deleted. With this service, there is no need for the provider to track every data access in order to potentially inform third parties about the deletion of the data object. Every provider can retrieve the status of any data object on her own. Thus, this service closes the gap between the legislation (i.e., inform third parties) and the technical possibilities (i.e., hard to track every access).

Furthermore, our service can be used by search engine providers or forwarding providers (e.g., proxies). With the DRS, search engines can automatically filter the search results by omitting all "revoked" data objects (here mostly websites) from the final list delivered to the user. Clearly, this can only be used for data objects where the data owner is the one requesting the data revocation. It cannot help in cases where data revocation is requested due to other circumstances by others than the data owner. However, at least in the first case, the time and effort for the data owner and search engine provider can be reduced to a minimum since it can be automatized. In contrast, currently every single data "deletion" must be processed and verified manually by the search engine provider, and the owner must proof her right, potentially in a legislative court.

Similar to search engines, pure forwarding providers like proxy services can refrain from forwarding data objects which have been revoked. In ideal case, the data would not be delivered from the provider who actually stores the document. However, since it is likely that not every single service provider will be law-abiding, there will remain some providers who simply ignore the data owner's demand. Nevertheless, if search engine providers, proxies, and most of the providers who store the actual data object comply with the data owners wishes, the data object will eventually get extinct and increasingly harder to be found on the Internet. Hence, in some way the Internet "forgets" revoked data objects.

In the future, we are going to further generalize our approach with respect to the type of data objects supported. Here, we investigate which functions are suitable for the generation of the unique *ID* when considering arbitrary data objects. Additionally, we look into how to further strengthen the DHT against denial of service attacks. Last but not least, we are currently in the process of evaluating the DRS with respect to scalability and resilience. For this, we use simulations and also develop a prototypical implementation. Furthermore, we evaluate the overhead for the providers in terms of additional messages and latency.

Acknowledgment. This work has been co-funded by the Social Link Project within the Loewe Program of Excellence in Research, Hessen, Germany.

References

1. Camera & Imaging Products Association Exchangeable Image File Format for Digital Still Cameras: Exif Version 2.3. Technical report, CIPA DC-008-2010 & JEITA CP-3451B Standard (2010)
2. Backes, J., Backes, M., Dürmuth, M., Gerling, S., Lorenz, S.: X-pire!-a Digital Expiration Date for Images in Social Networks. arXiv preprint arXiv: 1112.2649 (2011)
3. Backes, M., Gerling, S., Lorenz, S., Lukas, S.: X-pire 2.0: a user-controlled expiration date and copy protection mechanism. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 1633–1640. ACM (2014)
4. Castelluccia, C., De Cristofaro, E., Francillon, A., Kaafar, M.-A.: Ephpub: toward robust Ephemeral Publishing. In: 2011 19th IEEE International Conference on Network Protocols (ICNP), pp. 165–175. IEEE (2011)
5. European Commission. Factsheet on the "Right to be Forgotten" Ruling (C-131/12). http://ec.europa.eu/justice/data-protection/files/factsheets/factsheet_data_protection_en.pdf. Accessed 20th May 2015

6. European Commission. Proposal for a Regulation of the European Parliament, of the Council on the Protection of Individuals with Regard to the Processing of Personal Data, on the Free Movement of Such Data (General Data Protection Regulation), January 2012. http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf. Accessed 20th May 2015
7. Dabek, F., Li, J.,Sit, E., Robertson, J., Kaashoek, M.F., Morris, R.: Designing a DHT for low latency and high throughput. In: NSDI, vol. 4, pp. 85–98 (2004)
8. Dierks, T., Rescorla, E.: RFC 5246: The Transport Layer Security (TLS) Protocol. The Internet Engineering Task Force (2008)
9. Druschel, P., Backes, M., Tirtea, R.: The Right to Be Forgotten - Between Expectations and Practice (2011). http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/the-right-to-be-forgotten. Accessed 20th May 2015
10. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptology **1**(2), 77–94 (1988)
11. Galperin, S., Santesson, S., Myers, M., Malpani, A., Adams, C.: X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSpP (2013)
12. Geambasu, R., Kohno, T., Levy, A.A., Levy, H.M.: Vanish: increasing data privacy with self-destructing data. In: USENIX Security Symposium, pp. 299–316 (2009)
13. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, pp. 291–304. ACM (1985)
14. Google European Privacy Requests for Search Removals (2014). http://www.google.com/transparencyreport/removals/europeprivacy/?hl=en-US. Accessed 20th May 2015
15. Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B., Keleher, P.: Adaptive replication in peer-to-peer systems. In: Proceedings of the 24th International Conference on Distributed Computing Systems, pp. 360–369. IEEE (2004)
16. Hine, J.H., Dagger, P.: Securing distributed computing against the hostile host. In: Proceedings of the 27th Australasian Conference on Computer Science, vol. 26, pp. 279–286. Australian Computer Society Inc. (2004)
17. Jandt, S., Kieselmann, O., Wacker, A.: Recht auf Vergessen im Internet - Diskrepanz zwischen rechtlicher Zielsetzung und technischer Realisierbarkeit? Datenschutz und Datensicherheit (DuD) **37**(4), 235–241 (2013)
18. Krawczyk, H., Canetti, R., Mihir Bellare, H.: Keyed-hashing for Message Authentication (1997). https://tools.ietf.org/html/rfc2104. Accessed 20th May 2015
19. Liu, Q., Safavi-Naini, R., Sheppard, N.P.: Digital rights management for content distribution. In: Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers, vol. 21, pp. 49–58. Australian Computer Society Inc. (2003)
20. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
21. Mockapetris, P.: RFC 1034: Domain Names - Concepts and Facilities (1987)
22. Court of Justice of the European Union. Judgment in Case C-131/12. Press Release No 70/14, May 2014
23. Rainie, L., Kiesler, S., Kang, R., Madden, M.: Anonymity, privacy, and security online. Pew Research Center (2013)
24. Ruef, M.: Erfolgreicher Angriff gegen X-pire!, January 2011. http://www.scip.ch/?labs.20110131. Accessed: 20th May 2015

25. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev. **31**(4), 149–160 (2001)
26. Venkatesan, R., Koon, S.-M., Jakubowski, M.H., Moulin, P.: Robust image hashing. In: 2000 Proceedings of the International Conference on Image Processing, vol. 3, pp. 664–666. IEEE (2000)
27. Wacker, A., Schiele, G., Schuster, S., Weis, T.: Towards an authentication service for peer-to-peer based massively multiuser virtual environments. Int. J. Adv. Media Commun. **2**(4), 364–379 (2008)
28. Wang, Q., Daudjee, K., Özsu, M.T.: Popularity-aware prefetch in P2p range caching. Peer-to-Peer Netw. Appl. **3**(2), 145–160 (2010)
29. Wolchok, S., Hofmann, O.S., Heninger, N., Felten, E.W., Halderman, J.A., Rossbach, C.J., Waters, B., Witchel, E.: Defeating vanish with low-cost sybil attacks against large DHTs. In: NDSS (2010)

PerfectDedup: Secure Data Deduplication

Pasquale Puzio$^{1,2(\boxtimes)}$, Refik Molva^{2}, Melek Önen^{2}, and Sergio Loureiro^{1}

^{1} SecludIT, Sophia Antipolis, France
{pasquale,sergio}@secludit.com
^{2} EURECOM, Sophia Antipolis, France
{puzio,molva,onen}@eurecom.fr
http://www.secludit.com, http://www.eurecom.fr

Abstract. With the continuous increase of cloud storage adopters, data deduplication has become a necessity for cloud providers. By storing a unique copy of duplicate data, cloud providers greatly reduce their storage and data transfer costs. Unfortunately, deduplication introduces a number of new security challenges. We propose PerfectDedup, a novel scheme for secure data deduplication, which takes into account the popularity of the data segments and leverages the properties of Perfect Hashing in order to assure block-level deduplication and data confidentiality at the same time. We show that the client-side overhead is minimal and the main computational load is outsourced to the cloud storage provider.

Keywords: Cloud · Storage · Deduplication · Confidentiality · Encryption · Security · Perfect hashing

1 Introduction

Cloud storage providers constantly look for techniques aimed to minimize redundant data and maximize space savings. We focus on deduplication, which is one of the most popular techniques and has been adopted by many major providers such as Dropbox[1]. The idea behind deduplication is to store duplicate data only once. Thanks to such a mechanism, space savings can reach 70 % [7] and even more in backup applications. On the other hand, along with low costs, users also require the confidentiality of their data through encryption. Unfortunately, deduplication and encryption are two conflicting techniques. A solution which has been proposed to meet these two conflicting requirements is Convergent Encryption (CE) [4] whereby the encryption key is the result of the hash of the data segment. However, CE unfortunately suffers from various well-known weaknesses [9] including dictionary attacks. We propose to counter the weaknesses

Partially supported by the TREDISEC project (G.A. no 644412), funded by the European Union (EU) under the Information and Communication Technologies (ICT) theme of the Horizon 2020 (H2020) research and innovation programme.

[1] https://www.dropbox.com.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 150–166, 2016.
DOI: 10.1007/978-3-319-29883-2_10

due to CE by taking into account the popularity [10] of the data segments. Data segments stored by several users, that is, popular ones, are only protected under the weak CE mechanism whereas unpopular data segments that are unique in storage are protected under semantically-secure encryption. This declination of encryption mechanisms lends itself perfectly to efficient deduplication since popular data segments that are encrypted under CE are also the ones that need to be deduplicated. This scheme also assures proper security of stored data since sensitive thus unpopular data segments enjoy the strong protection thanks to the semantically-secure encryption whereas the popular data segments do not actually suffer from the weaknesses of CE since the former are much less sensitive because they are shared by several users. Nevertheless, this approach raises a new challenge: the users need to decide about the popularity of each data segment before storing it and the mechanism through which the decision is taken paves the way for a series of exposures very similar to the ones with CE. The focus of schemes based on popularity then becomes the design of a secure mechanism to detect the popularity of data segments.

In this paper we suggest a new scheme for the secure deduplication of encrypted data, based on the aforementioned popularity principle. The main building block of this scheme is an original mechanism for detecting the popularity of data segments in a perfectly secure way. Users can lookup for data segments in a list of popular segments stored by the Cloud Storage Provider (CSP) based on data segment identifiers computed with a Perfect Hash Function (PHF). Thanks to this technique, there is no information leakage about unpopular data segments and popular data segments are very efficiently identified. Based on this new popularity detection technique, our scheme achieves deduplication of encrypted data at block level in a perfectly secure manner. The advantages of our scheme can be summarized as follows:

– our scheme allows for storage size reduction by deduplication of popular data;
– our scheme relies on symmetric encryption algorithms, which are known to be very efficient even when dealing with large data;
– our scheme achieves deduplication at the level of blocks, which leads to higher storage space savings compared to file-level deduplication [7];
– our scheme does not require any coordination or initialization among users;
– our scheme does not incur any storage overhead for unpopular data blocks;

2 Secure Deduplication Based on Popularity

Given the inherent incompatibility between encryption and deduplication, existing solutions suffer from different drawbacks. CE was considered to be the most convenient solution for secure deduplication but it has been proved that is vulnerable to various types of attacks [9]. Hence, CE cannot be employed to protect data confidentiality and thus stronger encryption mechanisms are required.

We point out that data may need different levels of protection depending on its popularity [10] a data segment becomes "popular" whenever it belongs to more than t users (where t is the popularity threshold). The "popularity" of a block is viewed as a trigger for its deduplication. Similarly, a data segment is considered to be unpopular if it belongs to less than t users. This is the case for all highly sensitive data, which are likely to be unique and thus unlikely to be duplicated.

Given this simple distinction, we observe that popular data do not require the same level of protection as unpopular data and therefore propose different forms of encryption for popular and unpopular data. For instance, if a file is easily accessible by anyone on the Internet, then it is reasonable to consider a less secure protection. On the other hand, a confidential file containing sensitive information, such as a list of usernames and passwords, needs much stronger protection. Popular data can be protected with CE in order to enable source-based deduplication, whereas unpopular data must be protected with a stronger encryption. Whenever an unpopular data segment becomes popular, that is, the threshold t is reached, the encrypted data segment is converted to its convergent encrypted form in order to enable deduplication.

We propose to encrypt unique and thus unpopular data blocks (which cannot be deduplicated) with a symmetric encryption scheme using a random key, which provides the highest level of protection while improving the computational cost at the client. Whenever a client wishes to upload a data segment, we propose that she should first discover its popularity degree in order to perform the appropriate encryption operation. The client may first lookup for a convergent encrypted version of the data stored at the CSP. If such data segment already exists, then the client discovers that this data segment is popular and hence can be deduplicated. If such data segment does not exist, the client will encrypt it with a symmetric encryption scheme. Such a solution would greatly optimize the encryption cost and the upload cost at the client. However, a standard lookup solution for the convergent encrypted data segment would reveal the convergent encrypted data segment ID, that is the digest of the data computed under an unkeyed hash function like SHA-3, which would be a serious breach. Secure lookup for a data segment is thus a delicate problem since the ID used as the input to the lookup query can lead to severe data leakage as explained in [9,17]. Therefore, in such a scenario the main challenge becomes how to enable the client to securely determine the popularity of a data segment without leaking any exploitable information to the CSP. Also, the client needs to securely handle the "popularity transition", that is the phase triggered by a data segment that has just reached the popularity threshold t. More formally, the popularity detection problem can be defined as follows: given a data segment D and its ID ID_D, the client wants to determine whether ID_D belongs to the set P of popular data segment IDs stored at an untrusted CSP. It is crucial that if $ID_D \notin P$, no information must be leaked to the CSP. More generally, this problem can be seen as an instance of the Private Set Intersection (PSI) problem [26]. However, existing solutions are known to be costly in terms of computation and communication, especially when dealing with very large sets. Private Information Retrieval (PIR) [25] may also be a solution to this problem. However, using PIR raises two main issues: first, it would incur a significant communication overhead; second, PIR is designed to retrieve a single element per query, whereas an efficient protocol for the popularity check should allow to check the existence of multiple data segment IDs at once. Hence instead of complex cryptographic primitives like PSI and PIR we suggest a secure mechanism for popularity detection based on a lightweight building block called Perfect Hashing [11]. We aim at solving this problem by designing a novel secure lookup protocol, which is defined in next section, based on Perfect Hashing [11].

3 Basic Idea: Popularity Detection Based on Perfect Hashing

The popularity detection solution we propose makes use of the Perfect Hashing process which, given an input set of n data segments, finds a collision-free hash function, called the perfect hash function (PHF), that maps the input set to a set of m integers (m being larger than n by a given load factor). The CSP can run this process in order to generate the PHF matching the IDs of the convergent encrypted popular blocks that are currently stored at the CSP. The resulting PHF can be efficiently encoded into a file and sent to the client. Using the PHF received from the CSP, the client can lookup for new blocks in the set of encrypted popular block IDs stored at the CSP, as illustrated in Fig. 1. For each new block D, the client first encrypts the block to get $CE(D)$, he then computes the ID thereof using an unkeyed hash function h like SHA-3. Finally, by evaluating the PHF over ID, the client gets the lookup index i for the new block. The integer i will be the input of the lookup query issued by the client. Once the CSP has received the lookup query containing i, he will return to the client the convergent encrypted popular block ID stored under i. At this point, the client can easily detect the popularity of his data segment by comparing the ID he computed with the one received from the CSP: if the two IDs match, then D is popular. As mentioned above, it is a crucial requirement to prevent the CSP from discovering the content of the block D when it is yet unpopular. We achieve so by introducing an enhanced and secure version of Perfect Hashing, which makes the generated PHF one-way, meaning that the CSP cannot efficiently derive the input of the PHF from its output i. This also implies that the PHF must yield well-distributed collisions for unpopular blocks.

However, even though the client is now able to securely detect the popularity of a block, he still needs to handle the popularity transition, that is the phase in which a block reaches the threshold t and the convergent encrypted block needs to be uploaded to the CSP. Since the client cannot be aware of other copies of the same block previously uploaded by other users, a mechanism to keep track of the unpopular data blocks is needed. Clearly, the client cannot rely on the CSP for this task, as the CSP is not a trusted component. Therefore, we propose to introduce a semi-trusted component called Index Service (IS), which is responsible for keeping track of unpopular blocks. If the result of a popularity check is negative, then the client updates the IS accordingly by sending the popular convergent encrypted block ID and the ID of the symmetrically encrypted block. As soon as a block becomes popular, that is reaches the threshold t, the popularity transition is triggered and the client is notified in order to let him upload the convergent encrypted block, which from now on will be deduplicated by the CSP. Upon a popularity transition, the IS will delete from its storage any information related to the newly popular block. Regarding the popularity threshold, we point out that users do not have to be aware of its value, since the popularity transition is entirely managed by the IS, that is responsible for determining the current value for t. For instance, the value of t may be either static or dynamic, as proposed in [15]. Indeed, our scheme is completely independent of the strategy used for determining the value of the popularity threshold.

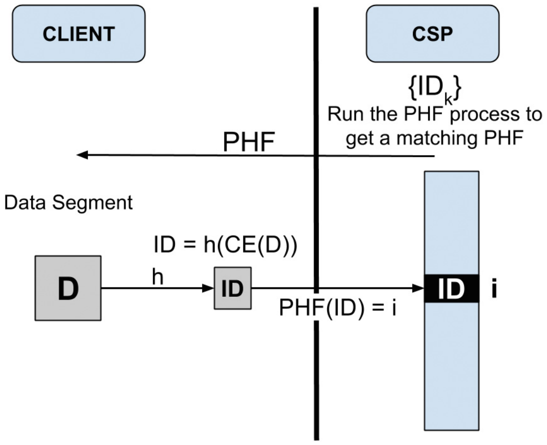


Fig. 1. The secure PHF allows users to detect popular blocks while preventing the CSP from discovering unpopular blocks

4 Background

4.1 Convergent Encryption

The idea of convergent encryption (CE) [4] is to derive the encryption key from the hash of the plaintext. A basic implementation of convergent encryption can be defined as follows: a user computes the encryption key using the message by applying a secure hash function H over M: $K = H(M)$; the message can then be encrypted with this key using a block cipher E: hence, $C = E(K, M) = E(H(M), M)$. Thanks to this technique, two users with two identical plaintexts will obtain two identical ciphertexts since the encryption key is the same and the encryption algorithm is deterministic. Despite its practicality, CE is known to be vulnerable to several weaknesses which undermine its capability of protecting confidential data and allow an attacker who has access to the storage server to perform offline dictionary attacks and discover predictable files. As shown in [9], CE is unfortunately exposed to the two following attacks: confirmation-of-a-file (COF) and learn-the-remaining-information (LRI). These attacks exploit the deterministic relationship between the plaintext and the encryption key and therefore can be successful in the verification whether a given plaintext has already been stored.

4.2 Perfect Hashing

A Perfect Hash Function (PHF) maps a set of arbitrary entries into a set of integers without collisions. Authors in [11] proposed a new algorithm that allows finding a perfect mapping for very large sets in a very efficient way. This algorithm, which is called CHD (Compress, Hash and Displace), achieves linear space and computational complexities (with respect to the size of the set). The main idea behind this algorithm is to split the input set into several buckets (subsets) with a few elements and find a collision-free mapping for each of these buckets separately. This approach has proved to be much more scalable than previous approaches. The mean number of elements per bucket is a parameter that can be tuned upon executing the generation algorithm. CHD also allows choosing a load factor, which is the fraction of non-empty positions in the hash table.

Although perfect hashing is widely adopted for efficient indexing in the field of relational databases [19], it has some desirable properties which make it an appropriate building block for our scheme. First, the computational complexity to build the PHF is linear and the PHF can be evaluated in constant time. Thanks to these properties, the system is scalable since the PHF generation remains feasible when dealing with very large datasets. In addition to that, the main computational load is outsourced to the CSP, while the client only has to perform very simple and lightweight operations such as evaluating the PHF on block IDs and symmetrically encrypting data blocks. Second, thanks to a special encoding and compression mechanism, the size of the PHF file is small and therefore it can easily be transferred to the client. Therefore, the performance impact is minimal and this approach can easily scale up to sets of millions of elements. Third, the resulting hash table is collision-free with respect to the elements of the input set (popular block IDs), meaning that any index is associated to at most one element of the input set. On the other hand, if the PHF is evaluated over the rest of the domain (unpopular block IDs) then collisions are well-distributed. This property is an important starting point to build our secure lookup protocol which must guarantee that an attacker is not able to determine on what input the PHF has been evaluated. Indeed, while an index in the hash table corresponds to a unique popular block ID, many unpopular block IDs are mapped to the same index. Therefore, given an index in the hash table, the CSP cannot determine the corresponding block ID. In our solution we propose to extend the existing PHF by replacing the underlying hash function with a one-way secure hash function such as SHA-3 [24]. Indeed, for the security of the scheme, it is crucial that the hash function used by the algorithm is one-way, meaning that it is easy to compute on a given input, but hard to invert given the image of a random input.

5 Our Solution

5.1 Overview

We consider a scenario where users want to store their data (files) on a potentially untrusted Cloud Storage Provider (CSP) while taking advantage of source-based block-level deduplication and protecting the confidentiality of their data at the same time. Users run a client C which is a lightweight component with respect to both storage and computational capacity. CSP is assumed to be honest-but-curious and thus correctly stores users' data while trying to disclose the content thereof. Prior to uploading its data, C runs a secure lookup protocol to check whether the data are popular. The CSP is responsible for the generation of the PHF over the popular blocks and the storage of the resulting collision-free hash table. The proposed protocol introduces a trusted third party called Index Service (IS) which helps the client to discover the actual number of copies of a yet unpopular block. We stress the fact that IS only stores information on unpopular blocks and once a block becomes popular, all corresponding information are removed from its database, hence this component does not need to have a significant storage capacity.

The proposed solution is described under three different scenarios:

- Unpopular data upload (Scenario 1): if C finds out that the data is yet unpopular, it performs the upload to the CSP and updates the IS;
- Popularity transition (Scenario 2): if C finds out that the popularity degree of the data is $t - 1$ (where t is the popularity threshold), then it performs the appropriate operations to upload the newly popular data. IS removes all information with respect to this specific data and CSP deletes all the encrypted copies previously stored;
- Popular data upload (Scenario 3): C only uploads metadata since it has detected that the requested data is popular, therefore deduplication can take place.

CSP stores a hash table for popular block IDs which is constructed with the previously introduced PHF. Each element of the hash table is defined by the couple $(PHF(h(CE(b_i))), h(CE(b_i)))$ where $h(CE(b_i))$ is the unkeyed secure hash of the convergent encrypted block. Before any operation, given the current set of popular blocks, CSP creates a corresponding secure PHF. This PHF is updated only when CSP needs to store new popular blocks. In the next sections, we first present the popularity check phase which is common to all three scenarios and then explain the following phases.

5.2 Popularity Check (Scenarios 1, 2 and 3)

Before uploading a file F, C splits F into blocks $F = \{b_i\}$, encrypts each of them with CE and computes their IDs. We point out that our scheme is completely independent of the underlying data-chunking strategy used for determining block boundaries, which is a problem that is out of the scope of this paper. The client fetches the PHF from the CSP and evaluates it over $\{h(CE(b_i))\}$. The result of this operation is a set of indices $I = \{PHF(h(CE(b_i)))\}$, where each index represents the position of the potentially popular block ID in the hash table stored at the CSP. These indices can be used to perform the popularity check without revealing the content of the blocks to the CSP. Indeed, given a set of indices obtained as above, the client can retrieve the corresponding block IDs stored in the hash table and then compare them with his own block IDs. Any block b_i such that $h(CE(b_i))$ is equal to the popular block ID retrieved from the CSP, is considered as popular, hence will be deduplicated. The index does not reveal any exploitable information on the block.

5.3 Popularity Transition (Scenarios 1 and 2)

If the popularity check reveals that a block is not popular, C needs to check whether it is going to trigger a popularity transition. A block becomes popular as soon as it has been uploaded by t users. In order to enable C to be aware of the change of the popularity status and perform the transition, C sends an update to the IS whenever the popularity check has returned a negative result for a given block ID. IS stores a list of block IDs and owners corresponding to each encrypted copy of the yet unpopular block. When the number of data owners for a particular block reaches t, the popularity transition protocol is triggered and IS returns to C the list of block IDs. In order to complete this transition phase, CSP stores the convergent-encrypted copy, removes the corresponding encrypted copies and updates the PHF. From now on, the block will be considered popular, therefore it will be deduplicated. We point out that this operation is totally transparent to the other users who uploaded the same block as unpopular. Indeed, during their upload phase, users also keep encrypted information about the convergent encryption key. This allows them decrypting the block when it becomes popular.

5.4 Data Upload (Scenarios 1, 2 and 3)

Once the client has determined the popularity of each block, he can send the actual upload request. The content of the request varies depending on the block status. If the block is unpopular, C uploads the block symmetrically encrypted with a random key. If the block is popular, C only uploads the block ID, so that the CSP can update his data structures. Optionally, in order to avoid to manage the storage of the encryption keys, C may rely on the CSP for the storage of the random encryption key and the convergent encryption key, both encrypted with a secret key known only by the client.

6 Security Analysis

In this section, we analyze the security of the proposed scheme, the CSP being considered the main adversary. The CSP is "honest-but-curious", meaning that it correctly performs all operations but it may try to discover the original content of unpopular data. We do not consider scenarios where the CSP behaves in a byzantine way. We assume that CSP cannot collude with the IS since this component is trusted. Since the goal of the malicious CSP is to discover the content of unpopular blocks, we analyze in detail whether (and how) confidentiality is guaranteed for unpopular data in all phases of the protocol. However, if the user wants to keep a file confidential even when it becomes popular, he may encrypt the file with a standard encryption solution and upload it to the cloud without following the protocol steps. Finally, we also analyze some attacks that may be perpetrated by users themselves and propose simple countermeasures against them.

Security of Blocks Stored at the CSP. By definition, an unpopular block is encrypted using a semantically-secure symmetric encryption. The confidentiality of unpopular data segments thus is guaranteed thanks to the security of the underlying encryption mechanism.

Security During Popularity Check. The information exchanged during the Popularity Check must not reveal any information that may leak the identity of an unpopular block owned by the user. The identity of an unpopular block is protected thanks to the one-wayness of the secure PHF: the query generated by the client does not include the actual unpopular block ID but an integer i that is calculated by evaluating the secure PHF on the block ID. Simple guessing by exploring the results of the secure hash function embedded in the PHF is not feasible thanks to the one-wayness of the underlying secure hash function (SHA-3 [24]). In addition to that, when the PHF is evaluated over an unpopular block ID, there is definitely a collision between the ID of the unpopular block and the ID of a popular block stored at the CSP. These collisions serve as the main countermeasure to the disclosure of the unpopular block ID sent to the CSP during the lookup. With a reasonable assumption, we can also consider that the output of the underlying secure hash function (SHA-3) is random. In case of a collision between an unpopular block ID and the ID of a popular block stored at the CSP, thanks to the randomness of the underlying secure hash function, the output of a PHF based on such a hash function is uniformly distributed between 0 and m. In the case of such a collision, the probability that the CSP guesses the unpopular block ID used as input to the PHF by the client thus is:

$$\frac{m}{|\bar{P}|} = \frac{|P|}{|\bar{P}| * \alpha} \tag{1}$$

where P is the set of popular block IDs stored at the CSP, $\bar{P}$ is the rest of the block ID domain including all possible unpopular block IDs, α is the load factor of the PHF such that $m = \frac{|P|}{\alpha}$.

Assuming that the cardinality of the entire domain is much larger than the cardinality of the set of popular block IDs (which is the case if popular block IDs are the result of a secure hash function), we can state that the number of collisions per index is large enough to prevent a malicious CSP from inferring the actual block ID used as input to the PHF. In a typical scenario using a

PHF based on a secure hash function like SHA-3, whereby the complexity of a collision attack would be 2^{256}, and a popular block ID set with 10^9 elements, this probability will be ($\alpha = 0.81$):

$$\frac{10^9}{(2^{256} - 10^9) * 0.81} \approx 1.06 * 10^{-68} \tag{2}$$

Hence collisions can effectively hide the identity of unpopular blocks from an untrusted cloud provider while keeping the lookup protocol extremely efficient and lightweight for the users.

Security Against Potential Protocol Vulnerabilities. We now consider a few additional attacks that may be perpetrated by the CSP. For each of them, we propose simple but effective countermeasures, which are easy to implement and do not significantly increase the computational and network overhead. First, we consider that the CSP may pre-build a PHF based on some specific data (derived for example from a dictionary) which have not been yet uploaded by users. Within such a scenario, clients would detect their requested block to be popular although it has never actually been uploaded by any user; such a block will then be stored with a lower level of protection. As a countermeasure to such an attack, we propose that the IS attaches a signature to each popular block ID upon the Popularity Transition. Therefore, the IS will sign popular block IDs before being stored at the CSP, enabling clients to verify the authenticity of these blocks when running the popularity check. Such a countermeasure would have a minimal impact on the performance of the system. Another attack we consider is related to the confirmation-of-file attack to which convergent encryption is also vulnerable [9]. Indeed, upon a Popularity Check, the CSP may compare the sequence of indices sent by the client with the sequence produced by a given popular file F. If the two sequences match, then there is a chance that the client is actually uploading F. In order to hide this information from the CSP, the client may add a number of random indices to the list of indices being sent upon the Popularity Check. Thanks to the resulting noise included in the index list, the identification of the target file by the CSP will be prevented. This countermeasure also prevents the CSP from running the learn-the-remaining-information attack. Moreover, the overhead due to this countermeasure is negligible both in terms of bandwidth and computation.

Security Against Users. Users may force a popularity transition by repeatedly uploading random or targeted blocks. As a countermeasure, the popularity threshold may be set to a value $t' = t + u$, where u is the expectation of the maximum number of malicious users. As opposed to the proposal of [10], the threshold can be dynamically updated at any time of the system life. Indeed, this parameter is transparent to both users and the CSP, hence the Index Service can update it depending on the security needs. Users may also perpetrate a DoS attack by deleting random blocks stored at the cloud. This may happen upon a popularity transition: the client is asked to attach a list of block IDs that may not be the actual encrypted copies of the block being uploaded. We suggest making the Index Service sign the list of block IDs to be deleted so that the cloud can verify whether the request is authentic. This signature does not significantly increase the overhead since several schemes for short signatures [22] have been proposed in the literature.

7 Performance Evaluation

7.1 Prototype Implementation

In order to prove the feasibility of our approach, we implemented a proof-of-concept prototype consisting of the three main components, namely, the Client, the IS and the CSP. All components have been implemented in Python. Cryptographic functions have been implemented using the pycrypto library[2]. Both the Client and the IS run on an Ubuntu VM hosted on our OpenStack platform, while the CSP runs on an Ubuntu VM hosted on Amazon EC2 (EU Region). The IS uses REDIS[3] in order to store the information on unpopular blocks, which are encoded as lists. Metadata (block IDs, file IDs, files structures, encrypted keys) are stored in a MySQL database. Perfect Hashing has been implemented using the CMPH library[4] at both the Client and the CSP. In order to achieve one-wayness, we customized CMPH by replacing the internal hash function with SHA256 [20]. We stress the fact that this is a proof-of-concept implementation, therefore for the sake of simplicity the CSP has been deployed on a VM where data blocks are stored locally. In a production environment, the CSP service may be deployed on a larger scale and any storage provider such as Amazon S3[5] may be employed to physically store blocks.

We consider a scenario where the client uploads a 10 MB file to the CSP pre-filled with 10^6 random blocks. We propose to first evaluate the computational overhead of each single component and measure the total time a client needs to wait during each phase until the data upload has been completed. We then analyze the network overhead of the proposed solution. Our analysis considers the three previously described scenarios:

- Scenario 1 (Unpopular File): the file to be uploaded is still unpopular;
- Scenario 2 (Popularity Transition): the file has triggered a popularity transition hence is going to become popular;
- Scenario 3 (Popular File): the file to be uploaded is already popular.

7.2 Computational Overhead

In this section we present our measurements of the computational overhead at each component and then show the total time a client takes to upload a file.

[2] https://pypi.python.org/pypi/pycrypto.
[3] http://redis.io.
[4] http://cmph.sourceforge.net/.
[5] https://aws.amazon.com/s3.

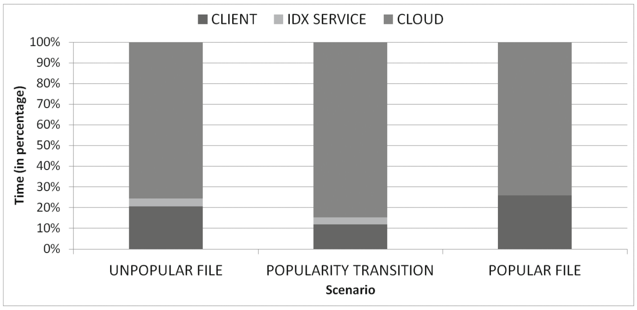


Fig. 2. Portion of the total computation time spent at each component in each scenario

Figure 2 shows an aggregate measure of all computation-intensive operations each component performs. The results prove that, as expected, the computational overhead introduced in the CSP is much higher than the one affecting the client. Also, since the operations performed by the IS are extremely simple, its computational overhead is negligible.

Figure 3 shows more detailed results by highlighting which operations introduce a higher computational overhead. The results prove that:

– Symmetric encryption introduces a negligible computational overhead, hence it does not affect the system performance;
– The client-side Popularity Check is extremely lightweight and thus introduces a negligible computational overhead;
– The most computation-intensive operations (PHF generation, hash table storage, upload processing) are performed by the CSP, hence a big fraction of the computational overhead is outsourced to the CSP.

Figures 4 and 5 show the results of an in-depth study on the performance of the Perfect Hashing algorithm, both in terms of storage space and computation time for the generation of the PHF. The generation time also includes the time needed to store the hash table. We measured these quantities on a dataset of 10^6 random block IDs while varying the load factor and the bucket size. The former is a coefficient indicating the fraction of non-empty positions in the final collision-free hash table; the latter is the mean number of elements in each subset of the input set (see [11] for further details). As we can observe from Figs. 4 and 5, the optimal bucket size is between 3 and 4 and the load factor should not be greater than 0.8. These parameters can be tuned depending on the scenario (e.g. bandwidth) in order to achieve the best performance.

Furthermore, as mentioned earlier, in order to improve the security of our novel lookup protocol, we replaced the default hash function employed by the CMPH library (Jenkins [21]) with SHA-3. This improvement is required for the following reason: using a non-secure hash function would allow an adversary such as the CSP to easily enumerate all block IDs mapped to a given index of the hash table. Such a threat may compromise the security of the whole system and make the popularity check protocol insecure.

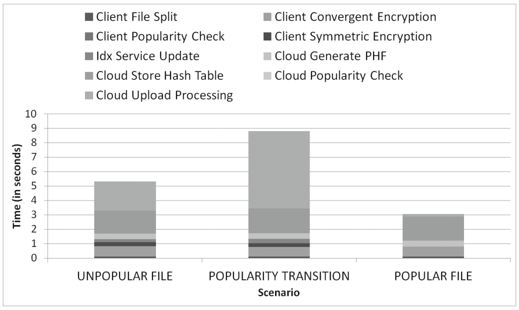


Fig. 3. Total time spent during each phase of the protocol in each scenario

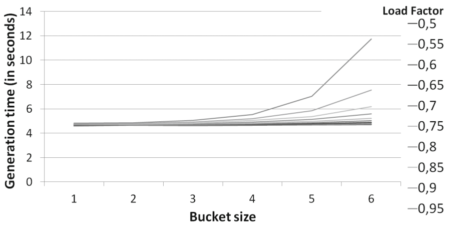


Fig. 4. Analysis of PHF generation time with varying parameters for a set containing 10^6 elements

Conclusion. Figure 6 summarizes all measurements by showing the total time spent during each phase of the upload protocol within the three scenarios. These results show that despite the delay introduced by the Popularity Check phase, the user achieves a throughput of approximately 1MB per second even when a file does not contain any popular block.

7.3 Communication Overhead

In this section we analyze the communication overhead of our scheme considering the same scenarios. The upload has been split into multiple sub-operations: PHF Download, Popularity Check, Index Service Update (not performed in Scenario 2) and the Upload. For each of these operations we analyze the size of all messages exchanged (both requests and responses). Table 1 regroups all the results expressed in MB. The PHF Download response size is linear with respect to the set of popular block IDs. The larger the set, the larger the response will be. However, as shown in [11], the size of PHF file is about 1.4 bits per popular block ID; hence this operation does not introduce a significant delay even when dealing with very large datasets. We point out that the PHF file does not have to be downloaded at every request, since the user can cache it. Furthermore, the size of the Popularity Check request and response is linear with respect to the number of blocks in the file that is being uploaded. The Popularity Check request contains a list of indices (one integer per block), while the response contains a list of block IDs (one per index) of 32 bytes each. The Index Service Update

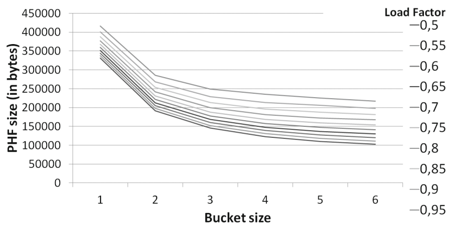


Fig. 5. Analysis of PHF size with varying parameters for a set containing 10^6 elements

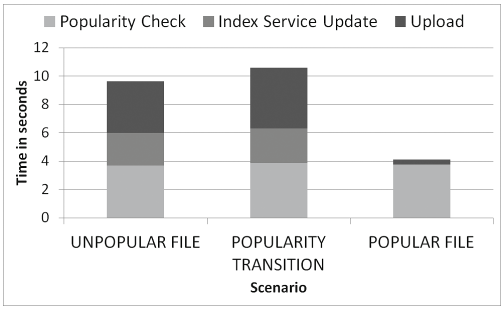


Fig. 6. Total time spent by all components when uploading a file (including Popularity Check) in each scenario

request is only sent for unpopular blocks. The request consists of two block IDs (32 bytes each) per block. The response size varies depending on whether the popularity transition occurs. If the file has triggered a popularity transition, then the response includes a list of block IDs, otherwise it is empty. As we can see from Table 1, requests and responses of the Popularity Check and the Index Service Update operations have a negligible size with respect to the file size. Finally, the size of the Upload request varies depending on the block status. If a block is popular, the request only consists of the block ID and one key (32 bytes). If a block is not popular, the request contains the encrypted data, two keys (32 bytes each) and a few fields: the file ID (32 bytes), the user ID and the block status (1 byte). As shown in Table 1, the overhead introduced by the Upload is minimal and mainly depends on the encoding method used to transfer the encrypted binary data. For simplicity, we used JSON objects to pack encrypted blocks and keys and Base64 to encode binary data, which increases the size of the data by 1/3. To summarize, the preliminary operations performed in our scheme before the Upload introduce a negligible communication overhead. In addition, the scheme does not affect the gains in terms of storage space and bandwidth achieved thanks to deduplication.

8 Related Work

Secure deduplication for cloud storage has been widely investigated both in the literature and in the industry. Convergent encryption, has been proposed as a

Table 1. Communication overhead (in MB) introduced by each operation

	SCENARIO 1	SCENARIO 2	SCENARIO 3
PHF DOWNLOAD IN	0.67	0.67	0.67
POPULARITY CHECK REQUEST	0.004	0.004	0.004
POPULARITY CHECK RESPONSE	0.02	0.02	0.02
INDEX SERVICE UPDATE REQUEST	0.1	0.1	-
INDEX SERVICE UPDATE RESPONSE	0.009	0.04	-
UPLOAD REQUEST	13.51	13.47	0.09

simple but effective solution to achieve both confidentiality and deduplication [1,3,4]. However, it is vulnerable to well-known attacks which put data confidentiality at risk [3,4]. A relevant work on this topic is DupLESS [8], which is based on a privacy-preserving protocol running between the user and a trusted key server. If an attacker learns the secret stored at the key server, confidentiality can no longer be guaranteed. Recently, a system called ClouDedup [9] has been proposed, which achieves secure and efficient block-level deduplication while providing transparency for end users. However, the system relies on a complex architecture in which users have to trust an encryption gateway which takes care of encrypting/decrypting data. Similarly to DupLESS, the leakage of the secret key compromises confidentiality. Another relevant work is iMLE [2], which proposes an elegant scheme for secure data deduplication. However, the scheme is purely theoretical, hence cannot be adopted in real scenarios. In fact, it makes an extensive use of fully homomorphic encryption [23]. To the best of our knowledge, one of the most recent and relevant works in the field of secure data deduplication is [10], which is based on the idea of differentiating data protection depending on its popularity and makes use of a mixed cryptosystem combining convergent encryption and a threshold encryption scheme. However, this work suffers from a few drawbacks which we aim to solve. First, the system suffers from a significant storage and bandwidth overhead. Indeed, for each unpopular file the user uploads two encrypted copies, one encrypted with a random symmetric key and one encrypted with the mixed encryption scheme. In scenarios with a high percentage of unpopular files, the storage overhead will be significant and nullify the savings achieved thanks to deduplication. We propose to eliminate the storage overhead by storing one single copy for each data segment at a time, encrypted with either a random symmetric key or a convergent key. Second, the system proposed in [10] relies on a trusted component which provides an indexing service for all data, both popular and unpopular. We propose to limit the usage of this trusted component to unpopular data. In our scheme, popular data can be detected thanks to the secure lookup protocol, whereby [10] relies on the trusted component. Third, the effectiveness of the system proposed in [10] is limited to file-level deduplication, which is known to achieve lower space savings than block-level deduplication. Fourth, both the client and the CSP have to perform complex cryptographic operations based on threshold cryptography on potentially very large data. As opposed to this, our proposed scheme has been designed to perform only simple and lightweight cryptographic operations, which significantly lowers the cost for the client. Fifth, our scheme does not require any coordination or initialization among users as opposed to [10]'s requirement to setup and distribute key shares among users.

9 Conclusion and Future Work

We designed a system which guarantees full confidentiality for confidential files while enabling source-based block-level deduplication for popular files. The main building block of our system is our novel secure lookup protocol built on top of an enhanced version of Perfect Hashing. To the best of our knowledge, this is the first work that uses Perfect Hashing for a different purpose other than database indexing. Our system is not based on any key-management protocol, hence it does not require users to agree on a shared secret or trust a third party for storing encryption keys. A semi-trusted component is employed for the purpose of storing metadata concerning unpopular data and providing a support for detecting popularity transitions, meaning that a data block has just reached the popularity threshold. We also implemented a prototype of the proposed solution. Our measurements show that the storage, network and computational overhead is affordable and does not affect the advantage of deduplication. Also, we showed that the computational overhead is moved to the CSP, while the client has to perform very lightweight operations. As part of future work, PerfectDedup may be optimized in order to reduce the overhead due to the PHF generation and transmission.

References

1. Xu, J., Chang, E.-C., Zhou, J.: Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 195–206. ACM (2013)
2. Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication (2015)
3. Adya, A., Bolosky, W.J., Castro, M., Cermak, G., Chaiken, R., Douceur, J.R., Howell, J., Lorch, J.R., Theimer, M., Wattenhofer, R.P.: FARSITE: federated, available, and reliable storage for an incompletely trusted environment. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 1–14 (2002)
4. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, P., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 617–624. IEEE (2002)
5. Perttula. Attacks on convergent encryption. http://bit.ly/yQxyvl
6. Liu, C., Liu, X., Wan, L.: Policy-based de-duplication in secure cloud storage. In: Yuan, Y., Wu, X., Lu, Y. (eds.) ISCTCS 2012. CCIS, vol. 320, pp. 250–262. Springer, Heidelberg (2013)
7. Meyer, D.T., Bolosky, W.J.: A study of practical deduplication. ACM Trans. Storage (TOS) **7**(4), 14 (2012)

8. Bellare, M., Keelveedhi, S., Ristenpart, T.: DupLESS: server-aided encryption for deduplicated storage. In: Proceedings of the 22nd USENIX Conference on Security, pp. 179–194. USENIX Association (2013)
9. Puzio, P., Molva, R., Önen, M., Loureiro, S.: ClouDedup: secure deduplication with encrypted data for cloud storage. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), vol. 1, pp. 363–370. IEEE (2013)
10. Stanek, J., Sorniotti, A., Androulaki, E., Kencl, L.: A secure data deduplication scheme for cloud storage. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 99–118. Springer, Heidelberg (2014)
11. Belazzougui, D., Botelho, F.C., Dietzfelbinger, M.: Hash, displace, and compress. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 682–693. Springer, Heidelberg (2009)
12. Cox, L.P., Murray, C.D., Noble, B.D.: Pastiche: making backup cheap and easy. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 285–298 (2002)
13. Rabin, M.O.: Fingerprinting by random polynomials. Center for Research in Computing Techn., Aiken Computation Laboratory Univ. (1981)
14. Wilcox-O'Hearn, Z., Warner, B.: Tahoe: the least-authority filesystem. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 21–26. ACM (2008)
15. Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services, the case of deduplication in cloud storage. IEEE Secur. Priv. **8**(6), 40–47 (2010)
16. Is Convergent Encryption really secure? http://bit.ly/Uf63yH
17. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Nguyen, P.Q., Johansson, T. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 296–312. Springer, Heidelberg (2013)
18. Storer, M.W., Greenan, K., De Long, D., Miller, E.L.: Secure data deduplication. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 1–10. ACM (2008)
19. Olumofin, F., Goldberg, I.: Privacy-preserving queries over relational databases. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 75–92. Springer, Heidelberg (2010)
20. Description of SHA256. http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf
21. Jenkins hash function. http://www.burtleburtle.net/bob/c/lookup3.c
22. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
23. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. diss., Stanford University (2009)
24. SHA-3. http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf
25. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM (JACM) **45**(6), 965–981 (1998)
26. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)

Biometrics and Privacy Preservation

Privacy-Preserving Biometric Authentication and Matching via Lattice-Based Encryption

Constantinos Patsakis[1(✉)], Jeroen van Rest[2], Michał Choraś[3,4], and Mélanie Bouroche[5]

[1] Department of Informatics, University of Piraeus, Piraeus, Greece
kpatsak@unipi.gr
[2] TNO, The Hague, The Netherlands
jeroen.vanrest@tno.nl
[3] University of Science and Technology, UTP, Bydgoszcz, Poland
chorasm@utp.edu.pl
[4] ITTI Sp. z o.o., Poznan, Poland
mchoras@itti.com.pl
[5] Distributed Systems Group, School of Computer Science and Statistics, Trinity College, Dublin, Ireland
melanie.bouroche@scss.tcd.ie

Abstract. The continuous dependence on electronic media has radically changed our interactions, many of which are now performed online. In many occasions users need to authenticate to remote machines, but the hostile environment of the Internet may severely expose users and service providers. To counter these shortcomings, strong authentication is pushed forward. As a means to authenticate individuals, biometric authentication is gradually gaining more and more ground. While the use of biometric data enables many useful applications, these data are very sensitive. For this reason, it is essential to handle them with the least user exposure. In this work we propose a very efficient protocol for privacy-preserving biometric authentication using lattice-based encryption. More precisely, we exploit the homomorphic properties of NTRU to provide a robust and secure solution and provide experimental results which illustrate the efficacy of our proposal.

Keywords: Biometric authentication · Privacy-preserving authentication · Lattice-based encryption

1 Introduction

While we have transferred a wide variety of our social, economic and working interactions in the cyber world, one of the major challenges is to guarantee that all the entities involved are the ones they claim to be. To provide entity authentication most services depend on the secure exchange of credentials of the entities, which are assumed to be properly registered. In general, users are given a user name and they create a password which they use to access the services.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 169–182, 2016.
DOI: 10.1007/978-3-319-29883-2_11

While theoretically this model works and current protocols can provide high security standards, the truth is that most users do not pick good passwords, enabling an adversary to easily gain access.

The general concept of passwords is to authenticate users by something that they *know* (password). Another paradigm is to authenticate users by something that they *are*, something that cannot be forgotten or forged. The past decade, the use of biometric authentication is gradually becoming more widespread since the cost of the devices has been drastically reduced. While there is a variety of biometric authentication methods, ranging from fingerprints and vein recognition, to retina and iris scanning, all these methods have two inherent drawbacks:

– They are not exact: Regardless of the underlying data, every measurement is not exactly the same as the one registered. For instance, a user scans her iris to register as a user. The system extracts the pattern and stores the feature vector in the system. However, the next time that she will scan her iris, it is highly improbable that the system will extract the exact same feature vector. This differentiation is subject to many factors. For instance, the alterations might be introduced due to angle, motion, imaging noise, reflection etc. Therefore, all biometric authentication methods have a threshold τ which denotes how many differences in two measurements can be tolerated in order to authenticate a user.
– They are permanent: While one could easily pick another password if a service has been compromised, she could not change her eyes or fingertips. If an adversary could acquire the biometric measurements of a user, then she could masquerade as her forever. Notably, depending on the method this data can be easily acquired and replicated[1].

Due to their nature and how they can be used, biometric data are very sensitive and should be dealt with much caution. Their fuzziness; the fact that two measurements of the same subject may differ, creates further problems. Implicit authentication is fairly easy when using passwords, a user may prove the knowledge of the password without actually revealing it. However, the fuzziness of biometric measurements renders such protocols useless.

The problem where two entities want to check whether the values that they hold are the same without presenting them to each other or to any other entity is widely known as private equality testing, and there are many solutions in the literature. However, if the underlying data are not equal, the case of biometric data, then most of these protocols cannot work as well, or they will be inefficient. For instance, if the two values may differ in τ bits, then one of the parties may need to present 2^τ candidate values for checking. Other approaches such as the scheme of Feigenbaum et al. [19] are far more efficient, but not efficient enough for such applications.

1.1 Contribution of This Work

In our work, we use the well-known NTRU [24] public encryption algorithm and exploit its efficiency and additive homomorphic property to enable

[1] http://www.ccc.de/en/updates/2014/ursel.

privacy-preserving biometric authentication and matching. An overview of the proposed protocol is the following. Assume that Alice and Bob hold a biometric measurement and Alice wants to know whether Bob's measurement differs from her measurements by less than a threshold value. First, both of them split the biometric measurement into blocks and Alice encrypts them with her NTRU public key, blinding them from every other entity. However, Bob is still able to perform some operations on the encrypted data, which in our case is to subtract the according block value from his biometric measurements. To obfuscate the results, Bob randomly permutes the results and returns them to Alice. While Alice can decrypt each block, she cannot recover the order of the blocks to find Bob's measurements. Thus, she can compute whether their measurements are below the required threshold without further information leakage. For the sake of simplicity and performance, we will present the protocol for the standard NTRU algorithm, nevertheless, adapting it to the more secure variant of Stehlé and Steinfeld [34] is straightforward and does not imply further changes than the obvious ones. In this paper the considered biometric modality is the iris. However, our privacy-preserving methodology can be applied to any modality which can be represented as sequence of bits such as faces, DNA etc.

1.2 Organization of This Work

The rest of this work is organized as follows. In Sect. 2 we provide a small overview of the NTRU algorithm and then present the state of the art in privacy-preserving biometric authentication. Section 3 introduces our protocol and discusses its security, mostly focusing on the semi-honest model. In Sect. 4 we provide some experimental results and compare its performance with current state of the art. Then in Sect. 6 we present some application scenarios were our protocol could be applied. Finally the article concludes with some remarks and ideas for future work.

2 Related Work

2.1 NTRU and Its Variants

Lattices are being studied for decades and several problems in their theory, such as the shortest and closest lattice vector have been proven to be extremely hard to solve, leading to the development of several public key encryption schemes. However, in the past few years the interest in these schemes has been greatly increased as these schemes provide many interesting features in terms of security and applications. For instance, while the widely used public key algorithms such as RSA and ElGamal could be broken with quantum algorithms, lattice-based encryption algorithms seem to be immune to such attacks making them a good candidate for the post-quantum era of cryptography [6]. Moreover, lattices have very interesting algebraic features that can be exploited to develop fully homomorphic encryption.

Table 1. NTRU parameters for different security levels

Level(bits)	p	q	n	D_1	D_2	D_3	D_g	D_m
128	3	2048	439	9	8	5	146	112
192	3	2048	593	10	10	8	197	158
256	3	2048	743	11	11	15	247	204

One of the most well known lattice based algorithms is NTRU [24]. The algorithm was developed in the mid 90 s and it is an extremely fast public key encryption algorithm. In fact it so efficient that its performance can be compared to symmetric ciphers [22]. Currently there are many variants, however in this work we will work with the original algorithm of Hoffstein, Pipher and Silverman. To generate the public/private key pair, we firstly, select some parameters N, p and q which are publicly known and determine the security of the NTRU instance. N is a prime number, denoting the degree of the polynomials that are going to be used. In what follows, every polynomial is reduced modulo the polynomial $x^N - 1$. The other two parameters, p and q are the two moduli numbers, the "large" (q); current standards set q equal to 2048, and one "small" (p) typically equal to 3. All NTRU operations are either performed in $\mathbb{Z}_q[x]/(x^N - 1)$ or in $\mathbb{Z}_p[x]/(x^N - 1)$. We then select two random polynomials f and g with small coefficients, that is -1, 0 and 1. We also require f to be invertible in $\mathbb{Z}_q[x]/(x^N - 1)$ and $\mathbb{Z}_p[x]/(x^N - 1)$, and we denote these inverses f_q and f_p respectively. The public key h is defined as $h = pgf_q$, while f and f_p are the private key. The most common parameters for NTRU are shown in Table 2.

To encrypt a message we map it to a polynomial m with small coefficients and pick a random "small" polynomial r, and send the message $c = hr + m \in \mathbb{Z}_q[x]/(x^N - 1)$. To decrypt c, the recipient multiplies it with f and rearranges the coefficients to reside within $[-q/2, q/2]$ and reduces it modulo p. Finally, she multiplies the result with f_p.

The amount of 1s, 0s and -1s in f, g, m and r are very important for NTRU. More precisely, a message can be decrypted only if the following inequality holds:

$$\|f * m + p * r * g\|_\infty \leq q$$

Otherwise the result will be a random polynomial. The randomness of r may introduce some problems in the decryption of the ciphertext, that is some ciphertext might not be decrypted. However proper parameter selection can bound this probability so that this event can be considered improbable.

NTRU has been extensively studied and after many attacks, the original parameters have been updated [23]. Currently, the algorithm is considered highly secure and has been standardized in both IEEE 1363.1 and X9.98. Moreover, NTRU has triggered the introduction of many variants such as [3,15,29], however of specific interest are the recent variants of Stehlé and Steinfeld [34] and the variant of Lopez et al. [27]. The first variant is CPA-secure in the standard model under the assumed quantum hardness of standard worst-case problems over ideal lattices, using Regev's learning with error approach [32]. The latter exploits the homomorphic properties of NTRU to create a fully homomorphic encryption scheme.

Generally, most lattice-based encryption schemes have homomorphic properties, however, there are specific constraints. In (partial) homomorphic encryption, the cryptographic primitives can transfer only one operation from the plaintext to one operation of the ciphertext, while recently introduced fully homomorphic encryption can transfer two operations. Nevertheless, in both cases the operations can be applied arbitrary amount of times. However, somewhat homomorphic or leveled encryption cannot support arbitrary homomorphic operations. For instance, in the case of NTRU, with each operation the amount of "noise" that is added is increased. Therefore, at one point the added noise is so high that the message cannot be recovered. Therefore, NTRU can support only a limited amount of additions and multiplications. Note that the homomorphic properties of NTRU hold over $\mathbb{Z}_p[x]/(x^N - 1)$, so for instance the additive property is applied over polynomials which is very important in our protocol.

2.2 Privacy-Preserving Biometric Authentication

Nowadays, biometric human identification is widely used in many large-scale security applications such as border crossings, visa/passports etc. Also law enforcement agencies use biometrics in order to search for criminals and terrorists. Several modalities, including iris, face or fingerprint, are very mature and the discussion now is not about the performance rates (FAR/FRR), but rather about the scalability and throughput of the system as well as on assuring privacy and fundamental human rights.

Iris is the part of the eye bounded by the pupil and sclera and it consists of muscle tissue [17]. Nowadays, iris acquisition devices are gaining momentum and can acquire high-quality images even of the walking subjects in operational environment (e.g. airport) [31]. Typically, the iris recognition system consists of the following steps: image acquisition using iris acquisition device(s), iris segmentation, extraction of iris features (such as eg. iris codes, Gabor filters or wavelets), and iris pattern matching. Hereby, in order to assure privacy and template security, especially in realistic systems used by law enforcement agencies, we also propose to add privacy-preserving methodology. The latter is considered a basic ingredient in building cyber-physical systems which are compliant with the "privacy-by-design" concept [11].

The first privacy-preserving identification protocol for iris was introduced by Blanton et al. [7] which exploit the homomorphic properties of the encryption method of Damgard et al. [16]. Based on the Paillier homomorphic scheme, Shahandashti et al. [33] propose a method for private fingerprint matching. Other approaches include the use of oblivious RAM from Bringer et al. [12] or garbled circuits from Luo et al. [28] and Bringer et al. [14]. Kulkarni and Namboodiri [26] use the somewhat homomorphic scheme of Boneh et al. [9] to privately compute the hamming distance of two sequences. Another approach, more focused on faces, would be to divide the biometric into smaller pieces, store them in independent compartments and use methods such as the one of Forczmański and Łabędź to identify them [20].

Similar methods have also been used in private DNA sequence matchmaking, as genetic information is also very sensitive [2], however, the size of the data render most of these methods inefficient. Therefore, many researchers have resulted to the use of a semi-trusted third party which can significantly improve computational and bandwidth requirements [25].

Recently, Blundo et al. [8] proposed a probabilistic protocol for the privacy-preserving evaluation of sample set similarity. Based on the MinHash approach, they sample each set, and perform the protocol of De Cristofaro et al. [18] to determine the cardinality of the common elements of both sets. More precisely, we assume that we have Alice and Bob, holding sets A and B respectively and that each one selects k values $(r_1, r_2, \dots, r_k)$ for the sample of their set, that is $a_{r_1}, a_{r_2}, \dots, a_{r_k}$ and $b_{r_1}, b_{r_2}, \dots, b_{r_k}$. Furthermore, we assume that Bob has published a prime p. Alice picks a random $\alpha, gcd(\alpha, p-1) = 1$ and sends Bob the message:

$$m_A = \{h(a_{r_i})^{\alpha} \mod p\}, i \in [1, k]$$

On receiving this message, Bob picks a random β and computes:

$$m'_A = \{m_{A_i}^{\beta} \mod p\}, i \in [1, k]$$

Then, Bob computes:

$$m_B = \{h(h(b_{r_i})^{\beta} \mod p)\}, i \in [1, k]$$

and sends Alice the message: $A' = \pi(m'_A), B' = m_B$ where π is a random permutation. Finally, Alice computes:

$$C = \{h(c^{\alpha^{-1} \mod p-1} \mod p)\}, \forall c \in A'$$

and checks how many elements in common does C have with B'. If there are ν, then Alice assumes that the Jaccard similarity of the two sets is approximately ν/k, subject to $\mathcal{O}(1/\sqrt{k})$ error.

Yasuda et al. [35,36] exploit the properties of the somewhat homomorphic scheme of Brakerski and Vaikuntanathan [10] by packing the feature vectors of the biometrics, however, their method was proven to be insecure [1].

An overview of these methods can also be found in [5,13].

3 The Proposed Protocol

3.1 Main Actors and Desiderata

Let us assume two entities, Alice, the initiator of the protocol and Bob, the responder. Both Alice and Bob hold a sequence of bits $\mathcal{A} = a_1, a_2, \dots, a_k$ and $\mathcal{B} = b_1, b_2, \dots, b_k$ respectively. The goal of Alice is to determine whether

$d_H(\mathcal{A}, \mathcal{B}) < \tau, \tau \in \mathbb{N}$; and d_H denotes the Hamming distance, without disclosing any information to Bob or anyone else. On the other hand, Bob is willing to allow this computation, nevertheless, he does not want to leak any information regarding $\mathcal{B}$ to Alice or another entity.

In what follows, we work in the *honest-but-curious/semi-honest model.* Therefore, while each party is assumed to follow each step of the protocol correctly (honest), they may try to analyze any received information or messages to extract information about their peers (curious). Therefore, if the protocol dictates that a participant should send a message of a specific form, we assume that the participant will conform, and will not send a tampered version.

3.2 The Protocol

We assume that Alice has created an NTRU key pair, so h is her public key and f, f_p her private. Both parties split their sequences in blocks of length λ, creating k blocks. Moreover, we assume that both of them know a function $\chi : \{0,1\}^\lambda \to \mathbb{D}$, where $\mathbb{D}$ contains the polynomials of $\mathbb{Z}_q[x]/(x^N - 1)$ with coefficients -1, 0 and 1. For the sake of simplicity instead of $\chi(m)$ we will write m. Additionally, we denote α_i and $\beta_i, i \in [1,k]$ the blocks of Alice and Bob respectively.

The steps of the protocol are as follows. Initially, Alice sends Bob the message

$$M_A = \{hs_i + \alpha_i\}, \forall i \in [1,k]$$

where s_i are random polynomials in $\mathbb{D}$. On receiving the vector m_A, Bob computes the vector

$$M_B = \{M_{A_i} - (hs'_i + \beta_i)\}, \forall i \in [1,k]$$

where s'_i are random polynomials in $\mathbb{D}$. That is the encryption of her blocks with NTRU. Then, Bob picks a random permutation π and sends Alice $M'_B = \pi(M_B)$. So Bob encrypts his blocks with NTRU, subtracts them from Alice's; he exploits the additive homomorphic property of NTRU, and rearranges them.

On receiving this message, Alice can decrypt each $M_{B'_i}$ and compute the weight w_i of each recovered message. If $\sum_{i=1}^{k} w_i < \tau$ then Alice deduces that $d_H(\mathcal{A}, \mathcal{B}) < \tau$. Figure 1 illustrates the proposed protocol.

Initially, Alice and Bob extract the templates of their biometrics and encrypt them in blocks using the NTRU encryption algorithm using Alice's public key. Alice sends her encrypted data to Bob who subtracts them in the according order and then permutes the results. Alice decrypts the messages to recover the Hamming weight and compare it against the threshold τ.

3.3 Protocol Correctness

In the first step, the protocol splits $\mathcal{A}$ into blocks and encrypts them to hide them from Bob. In the second step, Bob subtracts his values from the encrypted ones. If two values are the same, then they will cancel each other out, otherwise,

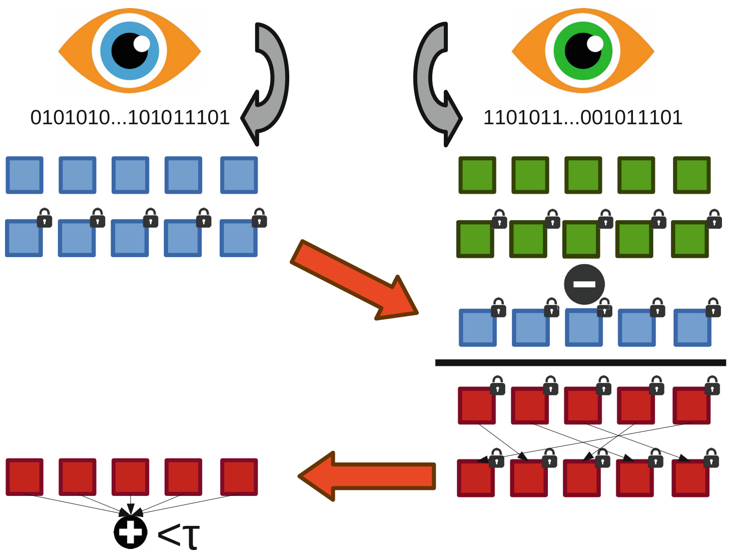


Fig. 1. The proposed protocol.

one coefficient (-1 or 1) is left in the encrypted block. Clearly, Bob's permutation does not alter the weight of the encrypted messages, but hides their order from Alice, who cannot recover Bob's sequence. Nevertheless, each block can be decrypted and the non-zero coefficients denote where each block differs with the others. Thus, Alice can easily find $d_H(\mathcal{A}, \mathcal{B})$.

3.4 Security of the Protocol

We do not consider active attacks; we assume that the messages exchanged in a protocol run are authenticated and integrity protected, thus the adversary is not able to modify or inject fake messages pretending to originate from another legitimate user.

Alice's input remains secret from Bob and any other active of passive adversary. Throughout the protocol, Alice sends a single message to Bob which contains her encrypted blocks. Therefore, anyone who wishes to recover Alice's input must break NTRU encryption which is considered infeasible. Note that NTRU is considered secure even from quantum algorithms.

While Alice can decrypt the encrypted blocks to compute their differences, in order to recover Bob's private input she has to find the proper order of κ blocks. This means $\kappa!$ arrangements, so finding the right order is infeasible. Clearly, an external adversary will not be able to recover any information about Bob's input, since it is encrypted with NTRU. Note that Bob does not simply subtract his input but he subtracts his encrypted input further confusing his output.

However, if Alice were malicious she could try to trick Bob and recover his input. For instance, instead of sending her input, she could mark each block and then put them in the right order. Since the information in each block is not going to fill it up to capacity, e.g. for 128 bits of security, NTRU can accept messages up to 439 bits but it will take only a fraction such as 32 or 64 bits, Alice could hide additional information in the unused bits. To counter such attack Bob could simply use a random padding for each message and alert Alice about its existence so that Alice would correctly calculate the weight of each block.

4 Experimental Results

We chose to compare our algorithm against the algorithm of Blundo et al. as it is the most efficient one in current state of the art, even though it samples the retinas and does not return exact results. The computer where the experiments were made has an Intel Core i3-2100 CPU at 3.1 GHz with 6 GB of RAM, running on Ubuntu 15.04 64 bit. The implementation in both cases is made in Sage 6.5[2]. For NTRU we have used the parameters proposed by SecurityInnovation[3], illustrated in Table 1. According to their recommendations, to generate f, we compute a polynomial $P(x)$ which is of the form $A_1(x)A_2(x) + A_3(x)$, where polynomial $A_i, i \in \{1, 2, 3\}$ have D_i coefficients set to 1 and D_i coefficients set to -1. Similarly, to construct polynomial g, we select a polynomial having D_g coefficients set to 1 and $D_g - 1$ coefficients set to -1. Finally, each message, when converted to polynomial must have at most D_m coefficients set to 1 and $D_m - 1$ coefficients set to -1. The set of parameters used for RSA and NTRU is shown in Table 2. The role of D_1, D_2 and D_3 is going to be discussed in Sect. 4.

Table 2. Parameters for the most popular security levels (in bits). For RSA the numbers denote the length (in bits) of the underlying modulo field according to NIST [4]. For NTRU, the numbers are precise and recommended by SecurityInnovation (https://github.com/NTRUOpenSourceProject/ntru-crypto/blob/master/doc/NewParameters.pdf).

Security level	RSA	NTRU			
		p	q	n	Public key (bits)
128	3072	3	2048	439	4829
192	7680	3	2048	593	6523
256	15360	3	2048	743	8173

The experimental results in Table 3 clearly indicate the performance gains of our protocol. It should be highlighted that Alice in the Blundo et al. protocol has to perform light calculations as the exponentiations are "soft", the exponent is 2^{16}, however the RSA decryptions of Bob are very intensive. Note

[2] sagemath.org.

[3] https://github.com/NTRUOpenSourceProject/ntru-crypto/blob/master/doc/NewParameters.pdf.

Table 3. Comparison of the Blundo et al. protocol with the proposed. Time in seconds and security in bits.

Security	Blundo et al.			Proposed		
	Alice	Bob	Total	Alice	Bob	Total
128	0.024	2.227	2.251	0.187	0.115	0.302
192	0.066	12.352	12.418	0.250	0.153	0.403
256	0.183	59.421	59.605	0.299	0.220	0.519

Table 4. Approximate communication cost in KB. Security in bits.

Security	Blundo et al.	Proposed
128	78.125	75.453
192	190.625	101.922
256	378.125	127.703

that the reported times account for a single thread in both cases, therefore, by multithreading these timings will be significantly reduced.

In our tests, we used random feature vectors of 2048 bits, such as iris. The sample for the Blundo et al. protocol was 100 bits, which accounts for an error of 10 %. Practically, this means that the "hard" computations for Alice and Bob are 100 RSA encryptions and 200 decryptions respectively.

In our protocol, we split retinas in blocks of 32 bits; that is 64 blocks, so Alice had to perform 64 encryptions and decryptions, while Bob had to perform 64 encryptions. The comparison of the communication cost for different security levels is shown in Table 4. Again, our proposed algorithm introduces lower communication costs compared to the protocol of Blundo et al. In fact, the higher the security level, the better our protocol performs. Note that the increase in the key length of NTRU is lower than RSA when the security level increases.

5 Discussion

Our proposed protocol has many benefits compared to its peers. The one that is most obvious is its performance, however, the there are other important aspects as well. For instance, the protocol manages to pack far more information than other protocols without reducing its security. Therefore, not only the bandwidth is reduced, but the protocol is secure in the post-quantum era. Undoubtly, one could use the Paillier [30] or the Goldwasser-Micali [21] cryptosystems to perform the XOR of the bits of the templates. However, to achieve the same level of security the bandwith overhead is considerably higher as only one bit would be processed at a time. Moreover, NTRU is far more efficient in terms of performance than any of these algorithms. One could argue that Alice could potentially find patterns regarding Bob's biometrics, with the risk being subject to the block size, the bigger the block, the higher the probability. While this is true, in the next paragraphs we provide a countermeasure for such attacks.

One generic attack of all these privacy preserving schemes is the following. Alice performs one execution of the protocol with Bob using firstly the sequence $00\dots000$ and then $10\dots000$. Clearly, comparing the output values Alice can determine whether the value of the first bit is 0 or 1. Having found the value of the first bit, Alice can proceed to the second bit etc. The main problem is that Bob uses the same template for each comparison and Alice can manipulate her own to find a better match at each execution. To counter this problem we propose the following method.

Let $\mathcal{F}(k, x)$ denote a Pseudo Random Function (PRF), where k is the PRF key and x is the point at which the function is evaluated. Bob proposes a random seed s so Alice and Bob compute the following for their sequences: $\mathcal{F}(s, m_i||i) \bmod 2, i \in \{1, 2, \dots, k\}$. Clearly, for each position where the bits of Alice and Bob are the same, the result is also going to be the same. However, when they differ, the result is going to be equal 50 % of the times. By processing their sequences like this, Bob's input is always randomized so Alice cannot perform this attack or find patterns in our scheme. Nevertheless, one should note that the threshold should now be close to half.

6 Application Scenarios

The presented methodology can be applied in several scenarios and it is valid for various biometric modalities. Herein, we are concerned with security scenarios, especially those interesting for law enforcement agencies, where preserving the privacy of citizens is challenging. On the one hand, the methodology can be applied for access control, where a person (the subject) wants to get access to a certain asset (e.g. terrain, building, room, laptop, service etc.), including critical infrastructures and high-risk assets with high accuracy biometrics such as iris. Such scenario can be realized in a verification mode (1:1 matching) or in the identification mode (1:many matching). In the latter case, so called white-listing is used, since the data (biometric feature vector) of the subject is matched versus those who can enter/gain access to the asset.

The second scenario where the proposed methodology is useful, is the matching of the subject biometric pattern versus templates from the law enforcement, or vice versa from private organisations. It can be realized as the typical 1:many identification or as the blacklisting. In such a case, e.g. the template of the subject (we can even imagine a wanted terrorist) is compared to the database of the people that agencies search for or those who are not allowed cross borders etc. The proposed methodology is useful because the law enforcement agency can query the database without disclosing who is the terrorist, and without learning anything about the other templates. Vice versa, private organisations can query law enforcement databases without disclosing any information about their customers.

7 Conclusions

The continuous use of biometrics might strengthen user authentication, however, it implies serious privacy risks. It should be understood that unlike passwords which can be easily generated, a user cannot generate a new body part, such as an iris or face. Addressing this challenge, privacy-preserving biometric authentication methods were recently introduced. These methods provide the needed functionality: biometric authentication, while simultaneously minimizing user's privacy exposure using state of the art cryptographic primitives. Clearly, this introduces a computational and communication overhead which might not be considered important in one-to-one scenarios - a user wants to authenticate to his device, but in one-to-many scenarios - a user authenticates to a server, the overhead might be substantial and decrease the quality of the provided service.

Based on the above, we introduced a novel protocol that takes advantage of the additive homomorphic property of NTRU to enable secure and exact privacy-preserving biometric authentication. Even if our implementation is not optimized, it is rather efficient, enabling it to be faster even than the "sampling" method of Blundo et al. In the future, we plan to explore the possibility of packing more data in each package with other algorithms and/or encodings to further decrease the computational and communication cost.

Acknowledgments. The research leading to these results has received funding by the European Commission under the Horizon 2020 Programme (H2020), as part of the *OPERANDO* project (Grant Agreement no. 653704) and the FP7 *TACTICS* project (Grant Agreement no. 285533) and is based upon work from COST Action *CRYPTACUS*, supported by COST (European Cooperation in Science and Technology).

The publication of this paper has been partly supported by the University of Piraeus Research Center.

References

1. Abidin, A., Mitrokotsa, A.: Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-lwe. In: IEEE International Workshop on Information Forensics and Security (WIFS), pp. 60–65. IEEE (2014)
2. Ayday, E., De Cristofaro, E., Hubaux, J.-P., Tsudik, G.: Whole genome sequencing: revolutionary medicine or privacy nightmare? Computer **2**, 58–66 (2015)
3. Banks, W.D., Shparlinski, I.E.: A variant of NTRU with non-invertible polynomials. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 62–70. Springer, Heidelberg (2002)
4. Barker, E., Dang, Q.: NIST special publication 800–57 part 3: Application-specific key management guidance. NIST Special Publication **800**(57) (2015)
5. Belguechi, R., Alimi, V., Cherrier, E., Lacharme, P., Rosenberger, C.: An overview on privacy preserving biometrics. In: Recent Application in Biometric, pp. 65–84. INTECH (2011). https://halv3-preprod.archives-ouvertes.fr/hal-00992461
6. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post-Quantum Cryptography. Springer Science & Business Media, Berlin (2009)

7. Blanton, M., Gasti, P.: Secure and efficient protocols for iris and fingerprint identification. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 190–209. Springer, Heidelberg (2011)
8. Blundo, C., De Cristofaro, E., Gasti, P.: EsPRESSo: efficient privacy-preserving evaluation of sample set similarity. In: Di Pietro, R., Herranz, J., Damiani, E., State, R. (eds.) DPM 2012 and SETOP 2012. LNCS, vol. 7731, pp. 89–103. Springer, Heidelberg (2013)
9. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM J. Comput. **43**(2), 831–871 (2014)
11. Bringer, J., Chabanne, H., Le Métayer, D., Lescuyer, R.: Privacy by design in practice: reasoning about privacy properties of biometric system architectures. In: Bjørner, N., de Boer, F. (eds.) FM : Formal Methods. LNCS, vol. 9109, pp. 90–107. Springer, Switzerland (2015)
12. Bringer, J., Chabanne, H., Patey, A.: Practical identification with encrypted biometric data using oblivious ram. In: International Conference on Biometrics (ICB), pp. 1–8. IEEE (2013)
13. Bringer, J., Chabanne, H., Patey, A.: Privacy-preserving biometric identification using secure multiparty computation: an overview and recent trends. IEEE Signal Process. Mag. **30**(2), 42–52 (2013)
14. Bringer, J., Favre, M., Chabanne, H., Patey, A.: Faster secure computation for biometric identification using filtering. In: 5th IAPR International Conference on Biometrics (ICB), pp. 257–264. IEEE (2012)
15. Coglianese, M., Goi, B.-M.: MaTRU: a new NTRU-based cryptosystem. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 232–243. Springer, Heidelberg (2005)
16. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. Int. J. Appl. Crypt. **1**(1), 22–31 (2008)
17. Daugman, J.: How iris recognition works. IEEE Trans. Circuits Syst. Video Technol. **14**(1), 21–30 (2004)
18. De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set intersection and union. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 218–231. Springer, Heidelberg (2012)
19. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. ACM Trans. Algorithms **2**(3), 435–472 (2006)
20. Forczmański, P., Łabędź, P.: Recognition of occluded faces based on multi-subspace classification. In: Saeed, K., Chaki, R., Cortesi, A., Wierzchoń, S. (eds.) CISIM 2013. LNCS, vol. 8104, pp. 148–157. Springer, Heidelberg (2013)
21. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 365–377. ACM (1982)
22. Hermans, J., Vercauteren, F., Preneel, B.: Speed records for NTRU. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 73–88. Springer, Heidelberg (2010)
23. Hirschhorn, P.S., Hoffstein, J., Howgrave-Graham, N., Whyte, W.: Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 437–455. Springer, Heidelberg (2009)

24. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
25. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.: Scaling private set intersection to billion-element sets. In: Christin, N., Safavi-Naini, R. (eds.) Financial Cryptography and Data Security. LNCS, vol. 8437, pp. 195–215. Springer, Heidelberg (2014)
26. Kulkarni, R., Namboodiri, A.: Secure hamming distance based biometric authentication. In: International Conference on Biometrics (ICB), pp. 1–6. IEEE (2013)
27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, pp. 1219–1234. ACM (2012)
28. Ying Luo, S., Cheung, T.P., Lazzeretti, R., Barni, M.: An efficient protocol for private iris-code matching by means of garbled circuits. In: 19th IEEE International Conference on Image Processing (ICIP), pp. 2653–2656. IEEE (2012)
29. Nevins, M., Karimianpour, C., Miri, A.: NTRU over rings beyond $\mathbb{Z}$. Des. Codes Crypt. **56**(1), 65–78 (2010)
30. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
31. Rakvic, R.N., Broussard, R.P., Kennell, L.R., Ives, R.W., Bell, R.: Iris acquisition device. In: Li, S.Z., Jain, A.K. (eds.) Encyclopedia of Biometrics, pp. 761–769. Springer, US (2009)
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 34 (2009)
33. Shahandashti, S.F., Safavi-Naini, R., Ogunbona, P.: Private fingerprint matching. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 426–433. Springer, Heidelberg (2012)
34. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) Advances in Cryptology - EUROCRYPT. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)
35. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Packed homomorphic encryption based on ideal lattices and its application to biometrics. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES Workshops 2013. LNCS, vol. 8128, pp. 55–74. Springer, Heidelberg (2013)
36. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Practical packing method in somewhat homomorphic encryption. In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W.M. (eds.) DPM 2013 and SETOP 2013. LNCS, vol. 8247, pp. 34–50. Springer, Heidelberg (2014)

Comprehensive and Improved Secure Biometric System Using Homomorphic Encryption

Avradip Mandal[1](✉), Arnab Roy[1], and Masaya Yasuda[2]

[1] Fujitsu Laboratories of America, Sunnyvale, CA, USA
{amandal,aroy}@us.fujitsu.com
[2] Kyushu University, Fukuoka, Japan
yasuda@imi.kyushu-u.ac.jp

Abstract. With the widespread development of biometric systems, concerns about security and privacy are increasing. An active area of research is template protection technology, which aims to protect registered biometric data. We focus on a homomorphic encryption approach, which enables building a "cryptographically-secure" system. In DPM 2013, Yasuda et al. proposed an efficient template protection system, using the homomorphic encryption scheme proposed by Brakerski and Vaikuntanathan. In this work, we improve and fortify their system to withstand impersonation attacks such as replay and spoofing attacks. We introduce a challenge-response authentication mechanism in their system and design a practical distributed architecture where computation and authentication are segregated. Our comprehensive system would be useful to build a large-scale and secure biometric system such as secure remote authentication over public networks.

Keywords: Biometric authentication · Homomorphic encryption · Template privacy

1 Introduction

Biometrics refers to human characteristics and traits of an individual such as fingerprint, iris, vein and signature. In biometric authentication, such characteristics are used to establish the identity of a person. Compared to the commonly used ID/password authentication schemes, biometrics do not require users to remember long and complex passwords. As a result, the usage of biometrics is now expanding in various applications ranging from international border crossings to securing information in databases (e.g., see US-VISIT [24]). On the other hand, concerns about security and privacy are expanding. In particular, it is important to protect *templates* which are stored biometric data, since once leaked, templates can neither be revoked nor replaced.

1.1 Template Protection and Its Approaches

The most straightforward way to make a biometric system secure is to put all the modules and the interfaces into a smart card (i.e., a secure processor). This

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 183–198, 2016.
DOI: 10.1007/978-3-319-29883-2_12

solution can never reveal any information outside the card. However, it is not useful for *large-scale* applications such as remote authentication since cards are relatively expensive and each client must always carry his or her own card for every authentication. There is also the vulnerability that raw templates can be extracted from stolen cards. To resolve the problem, template protection technology has been researched, and the following three main approaches (e.g., see [1,8,21]) exist:

Feature Transformation: Biometric data is transformed to random data by a client-specific key. Cancelable biometrics and biohashing are typical examples. This is practical in both performance and storage, but would no longer be secure if the client's key were leaked to adversaries.

Biometric Cryptosystem: Currently, these approaches are based on error correcting codes, and examples include fuzzy vault and fuzzy commitment. Since current schemes require strong restriction of authentication accuracy, both practical and security issues are controversial.

Homomorphic Encryption: This approach enables operating on encrypted data, without access to a decryption key. With the encryption, the confidentiality of biometric data can be protected, and the similarity of biometric data can be measured on encrypted data. As long as the secret key is managed by a trusted party, it makes a system cryptographically-secure. However, performance and ciphertext size are issues for practical usage.

1.2 Previous Work using Homomorphic Encryption

There is a line of work on enabling biometric matching while preserving the privacy of queried data and templates. In 2010, Osadchy et al. [19] introduced a privacy-preserving face identification system, called *SCiFI*, using the Paillier scheme [20] and oblivious transfer protocols. Similar to the SCiFI system, Blanton and Gasti [2] in 2011 developed privacy-preserving protocols for iris and fingerprints. Their extension is to combine additively homomorphic encryption such as the Paillier and the DGK schemes [9] with garbled circuit evaluation. In 2011, Evans et al. [12] also proposed an efficient privacy-preserving closest-matching system for fingerprint using the Paillier scheme and garbled circuits. Another line of work aims to build secure and large-scalable biometric systems, providing template protection security. In order to enhance the security of secure sketches of [11], Bringer and Chabanne [6] in 2008 modified the protocols of [7] using a simple homomorphic encryption scheme by Goldwasser and Micali [14] and private information retrieval protocols. In 2010, Upmanyu et al. [23] proposed a provably secure and blind authentication protocol by using the Paillier scheme. Based on [23], Hirano et al. [15,16] proposed cancelable protocols, which are cryptographically-secure against passive adversaries in a semi-honest model using pairing-based homomorphic encryption such as the BGN scheme [4]. In 2013, Kulkarni and Namboodiri [18] gave a one-to-one authentication protocol using the BGN scheme. While [15,16] involve three parties, the protocol of [18] is a client-server architecture for iris and palmprint matching. Their protocol is secure against a semi-honest adversary, and the privacy guarantee is that the server can learn only the matching score while the client learns the authentication result.

1.3 Our Contributions

In 2013, Yasuda et al. [26] proposed a template protection system applicable to various biometric feature codes (see Sect. 2.1), using the homomorphic encryption scheme proposed in [5]. In [26], the authors proposed a novel method to pack a feature code into a single ciphertext, and gave an efficient matching algorithm which can run over packed ciphertexts. Although the biometric system proposed in [26] can achieve template protection, it is not comprehensively secure in terms of biometric system security. In this paper, we propose several improvements for the security and privacy of the biometric authentication system proposed in [26]. Specifically, we develop challenge-response authentication mechanisms to mitigate replay attacks. The primary novelty of our work are the challenge-response schemes, which works in combination with the packing method of [26], while maintaining efficiency of secure matching. We also provide a detection technique against the universal spoofing attack introduced in [17], which is specific for homomorphic encryption systems and can output fake templates with high matching scores. The attacks from [17] can facilitate unauthorized access against the scheme from [26] by sending an encryption of spoof feature data into the biometric system, instead of ciphertext corresponding to genuine feature data. Our universal spoof detection technique can easily thwart such array of attacks. In order to demonstrate the efficiency of our improved system, we choose suitable parameters for the encryption scheme of [5], and report implementation results to show that our system is practical.

Notations. The symbols $\mathbb{Z}$ and $\mathbb{Q}$ denote the ring of integers and the field of rational numbers, respectively. For a prime p, the finite field with p elements is denoted by $\mathbb{F}_p$. For integers z and d, let $[z]_d$ denote the reduction of z modulo d included in $[-d/2, d/2)$ (the reduction of z modulo d included in $[0, d)$ is denoted by $z \bmod d$). For $\boldsymbol{A} = (a_1, \ldots, a_n)$, let $\|\boldsymbol{A}\|_\infty$ denote the ∞-norm $\max_i |a_i|$. For $\boldsymbol{A}$ and $\boldsymbol{B}$, we let $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$ and $d_H(\boldsymbol{A}, \boldsymbol{B})$ denote the inner product and the Hamming distance, respectively.

2 Preliminaries

2.1 Feature Codes and Homomorphic Encryption

A number of algorithms exist which extract a feature code from a biometric image. For an iris image, the algorithm of [10] can generate a code of 2048-bit. In 2013, Fujitsu Laboratories [13] announced that it has developed a new extraction algorithm for matching feature codes of 2048-bit from palm vein images. In[10,13], the Hamming distance gives a metric to measure the similarity of extracted feature codes; If the Hamming distance is smaller than a pre-defined threshold θ, the authentication is successful. The threshold θ depends on each extraction algorithm, and it is determined to take the balance of FAR (False Acceptance Rate) and FRR (False Rejection Rate). Here we assume that biometric data are always represented as binary codes of 2048-bit and the Hamming distance is used for matching (we do not discuss how to generate feature codes). Throughout this paper, we let $\boldsymbol{T}$, $\boldsymbol{Q}$ denote template and queried feature codes, respectively, and θ a fixed threshold.

Once feature codes are leaked, an adversary can illegally authenticate with the stolen codes. The homomorphic encryption approach can encrypt feature codes and compute the Hamming distance on encrypted data. Selection of an encryption scheme is the first task, and there are three types of schemes; *Additive schemes* can only support additions on encrypted data (e.g., the Paillier scheme [20]). In contrast, *somewhat homomorphic encryption (SHE)* can support both additions and multiplications, but the number of permissible operations is limited. The first SHE construction was the BGN scheme [4], which can only support depth-one multiplications, but it is sufficient to evaluate the Hamming distance (see [18]). *Fully homomorphic encryption (FHE)* can support "arbitrary" operations. After Gentry's breakthrough in 2009, a number of new FHE and associated SHE schemes have been proposed. Although current FHE is impractical, the associated SHE schemes can be used practically in biometrics (e.g., see [25,26]).

2.2 SHE Scheme Construction

Given an integer m, let ζ_m be a primitive m-th root of unity, and $\Phi_m(x) \in \mathbb{Z}[x]$ the m-th cyclotomic polynomial. We focus on SHE schemes with *homomorphic correctness* over the ring $R = \mathbb{Z}[x]/(\Phi_m(x))$ of integers of the cyclotomic field $\mathbb{Q}(\zeta_m)$. Informally, homomorphic correctness over R means that a scheme can support the ring operations over R. Here we present the SHE scheme proposed in [5], which has (somewhat) homomorphic correctness over R. The following four parameters are required:

$$\begin{array}{rl} n: & \text{2-power integer defining the ring } R = \mathbb{Z}[x]/(x^n+1) \\ & \text{(note that } \Phi_{2n}(x) = x^n + 1 \text{ in this case)} \\ q: & \text{prime number with } q \equiv 1 \bmod 2n\text{, which defines the} \\ & \text{base ring } R_q = \mathbb{F}_q[x]/(x^n+1) \text{ of ciphertext space} \\ t: & \text{integer with } t < q \text{ to determine plaintext space} \\ & R_t = (\mathbb{Z}/t\mathbb{Z})[x]/(x^n+1) \text{ (} t \text{ is not necessarily prime)} \\ \sigma: & \text{parameter defining a discrete Gaussian error} \\ & \text{distribution } \chi = D_{\mathbb{Z}^n,\sigma} \text{ with the standard deviation } \sigma \end{array} \tag{1}$$

We can regard samples from $\chi = D_{\mathbb{Z}^n,\sigma}$ as elements of R by the map:
$a = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}^n \mapsto \sum_{i=0}^{n-1} a_i x^i \in R.$

Key Generation. We choose an element $s \leftarrow \chi$ and then $s \in R$. We then sample a uniformly random element $p_1 \in R_q$ and an error $e \leftarrow \chi$. Set $\mathsf{pk} = (p_0, p_1)$ with $p_0 = -(p_1 s + te)$ as the public key and $\mathsf{sk} = s$ as the secret key.

Encryption. Given a plaintext $m \in R_t$ and the public key $\mathsf{pk} = (p_0, p_1)$, the encryption samples $u, f, g \leftarrow \chi$ and compute a *fresh* (i.e., non-operated) ciphertext given by

$$\mathsf{Enc}(m, \mathsf{pk}) = (c_0, c_1) = (p_0 u + tg + m, p_1 u + tf), \tag{2}$$

where the plaintext $m \in R_t$ is considered as an element of R_q in the natural way due to the condition $t < q$.

Homomorphic Operations. While the above encryption generates a ciphertext with only two elements of R_q, the homomorphic multiplication defined below makes the ciphertext length longer. Hence, we need to define homomorphic operations for ciphertexts of any length; Let $\mathsf{ct}' = (c'_0, \ldots, c'_\xi)$, $\mathsf{ct}'' = (c''_0, \ldots, c''_\eta)$ be two ciphertexts. The homomorphic addition "$\dot{+}$" is computed by component-wise addition $\mathsf{ct}' \dot{+} \mathsf{ct}'' = (c'_0 + c''_0, \ldots, c'_{\max(\xi,\eta)} + c''_{\max(\xi,\eta)})$ by padding with zeros if necessary. Similarly, the homomorphic subtraction can be defined. In contrast, the homomorphic multiplication "$*$" is computed by $\mathsf{ct}' * \mathsf{ct}'' = (\widehat{c}_0, \ldots, \widehat{c}_{\xi+\eta})$ with $\displaystyle\sum_{i=0}^{\xi+\eta} \widehat{c}_i z^i = \left(\sum_{i=0}^{\xi} c'_i z^i\right) \cdot \left(\sum_{j=0}^{\eta} c''_j z^j\right)$, where z denotes a symbolic variable.

Decryption. For any ciphertext $\mathsf{ct}' = (c'_0, \ldots, c'_\xi)$, decryption with the secret key $\mathsf{sk} = s$ is computed by

$$\mathsf{Dec}(\mathsf{ct}', \mathsf{sk}) = [\tilde{m}]_q \bmod t \in R_t, \tag{3}$$

where $\tilde{m} = \sum_{i=0}^{\xi} c'_i s^i \in R_q$. For $\boldsymbol{s} = (1, s, s^2, \ldots)$, we can rewrite $\mathsf{Dec}(\mathsf{ct}', \mathsf{sk}) = [\langle \mathsf{ct}', \boldsymbol{s} \rangle]_q \bmod t$.

2.3 Packing Method for Efficient Secure Matching

Component-wise encryption would cause performance slowdown in secure matching. In contrast, Yasuda et al. [26] proposed a new packing method for efficient computation of secure Hamming distance. Specifically, their method can pack a feature code into a single ciphertext, and furthermore, a certain linear combination of homomorphic operations can compute secure Hamming distance over packed ciphertexts. Below, we introduce their packing method for 2048-bit feature codes:

Definition 1. *Set $n \geq 2048$. For $\boldsymbol{T} = (t_0, \ldots, t_{2047})$ and $\boldsymbol{Q} = (q_0, \ldots, q_{2047})$, we define two types of polynomials in the base ring $R = \mathbb{Z}[x]/(x^n + 1)$ of the SHE scheme;*

$$\mathsf{pm}_1(\boldsymbol{T}) = \sum_{i=0}^{2047} t_i x^i \ \textit{and}\ \mathsf{pm}_2(\boldsymbol{Q}) = -\sum_{j=0}^{2047} q_j x^{n-j}.$$

Then two types of packed ciphertexts are defined as

$$\begin{cases} \mathsf{ct}^{(1)}_{\text{pack}}(\boldsymbol{T}) = \mathsf{Enc}(\mathsf{pm}_1(\boldsymbol{T}), \mathsf{pk}), \\ \mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}) = \mathsf{Enc}(\mathsf{pm}_2(\boldsymbol{Q}), \mathsf{pk}), \end{cases} \tag{4}$$

where we regard $\mathsf{pm}_1(\boldsymbol{T})$ *and* $\mathsf{pm}_2(\boldsymbol{Q}) \in R$ *as elements in the plaintext space* R_t *by taking sufficiently large* t.

The key point of [26] is that the constant term of one multiplication between $\mathsf{pm}_1(\boldsymbol{T})$ and $\mathsf{pm}_2(\boldsymbol{Q})$ gives the inner product $\langle \boldsymbol{T}, \boldsymbol{Q} \rangle$; Specifically, since $x^n = -1$ in R, we have

$$\mathsf{pm}_1(\boldsymbol{T}) \times \mathsf{pm}_2(\boldsymbol{Q}) = \langle \boldsymbol{T}, \boldsymbol{Q} \rangle + (\text{non-constant terms}).$$

Then the homomorphic correctness of the SHE scheme shows that only one homomorphic multiplication between $\mathsf{ct}^{(1)}_{\text{pack}}(\boldsymbol{T})$ and $\mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q})$ computes the inner product on encrypted data. This method can be applied to secure Hamming distance computation as follows [26, Sect. 3.2]:

Theorem 1. *Let* $C_1 = -\sum_{i=0}^{n-1} x^{n-i}$ *and* $C_2 = 2 - C_1 = \sum_{i=0}^{n-1} x^i$. *Let* ct_H *be a ciphertext given by*

$$\mathsf{ct}^{(1)}_{\text{pack}}(\boldsymbol{T}) * C_1 \dotplus \mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}) * C_2 \dotplus (-2\mathsf{ct}^{(1)}_{\text{pack}}(\boldsymbol{T})) * \mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}), \tag{5}$$

where homomorphic operations are defined in the same way as in Sect. 2.2. Then, the constant term of $\mathsf{Dec}(\mathsf{ct}_H, \mathsf{sk})$ *gives our desired Hamming distance* $d_H(\boldsymbol{T}, \boldsymbol{Q})$.

3 Improved Security and Privacy

Here we introduce three techniques to improve both security and privacy in the template protection system of [26].

3.1 Countermeasure Against Replay Attack

Insecure communication allows an adversary to mount hill-climbing and replay attacks. However, as mentioned in [1, Sect. 2.6], simple-minded use of a public key infrastructure does not always give a complete solution against replay attacks. Consider a case where all communicated data are encrypted by utilizing a public key infrastructure. If an adversary intercepts the encrypted data of a legitimate client, he cannot recover the raw data but he can stage a replay attack by sending the captured data, if no freshness is introduced. Furthermore, an insider adversary can steal raw biometric data since they are decrypted in the system (i.e., secure communication is not sufficient for adversaries having power to observe internal memory of the system). A challenge-response authentication mechanism is effective against the external replay attack, and it is a family of two-party protocols where one party presents a question, called a "challenge", and another party provides a valid answer, called a “response”, to be authenticated successfully. To protect against insider adversaries, we employ homomorphic encryption. As in [3,16], we try to introduce a challenge-response mechanism in the system of [26]. The difficulty is to combine the mechanism with the packing method of Sect. 2.3 so as not to reduce efficiency of the secure matching (5).

3.2 Countermeasure Against Spoofing Attack

In general, given a template $\boldsymbol{T}$, it is hard to find a binary vector $\boldsymbol{Q}'$ such that $d_H(\boldsymbol{T}, \boldsymbol{Q}') < \theta$ without knowing any information about $\boldsymbol{T}$ (the hardness mainly depends on θ). However, we can easily generate a *non-binary* vector $\boldsymbol{Q}'$ satisfying $d_H(\boldsymbol{T}, \boldsymbol{Q}') < \theta$. For example, take $\boldsymbol{Q}' = (q'_0, q'_1, \ldots, q'_{2047})$, where only one entry q_j has a large value $e > 0$ and the other entries are equal to either 0 or 1. Then, for any $\boldsymbol{T} = (t_0, t_1, \ldots, t_{2047})$, the Hamming distance $d_H(\boldsymbol{T}, \boldsymbol{Q}')$ equals to

$$\sum_{i=0}^{2047}(t_i + q'_i - 2t_iq'_i) \approx \begin{cases} 1024 + e \ (\text{if } t_j = 0) \\ 1024 - e \ (\text{if } t_j = 1) \end{cases} \tag{6}$$

where we assume $\sum_{i \neq j}(t_i + q'_i - 2t_iq'_i) \approx 1024$. Hence, if $e > 1024 - \theta$, the Hamming distance $d_H(\boldsymbol{T}, \boldsymbol{Q}')$ becomes smaller than θ with about 50 % success probability (such attacks are introduced in [17]). Here we call such vectors $\boldsymbol{Q}'$ *universal spoofing vectors*, which result in a low Hamming distance for any binary code $\boldsymbol{T}$ (with high probability).

With a universal spoofing vector, a malicious client can attack a biometric system. Note that the attack cannot be detected in a homomorphic encryption system since biometric codes are always in encrypted format. Our countermeasure is simple and it is to check the Hamming distance for a *dummy* template $\boldsymbol{T}'$, instead of a genuine one $\boldsymbol{T}$. Consider a case where a malicious client sends a ciphertext of $\boldsymbol{Q}'$ to a computation server. Then the server randomly generates a dummy template $\boldsymbol{T}'$, and computes the encrypted Hamming distance ct between $\boldsymbol{T}'$ and $\boldsymbol{Q}'$ as in Theorem 1. Then the server sends the ciphertext to a decryptor. Finally, the decryptor decrypts it to obtain $d_H(\boldsymbol{T}', \boldsymbol{Q}')$, and checks its size to detect the spoofing attack. As seen from Fig. 1, there are two distributions of Hamming distance values; one distribution is obtained from genuine clients, and another one is from intruders. For a genuine code $\boldsymbol{Q}$, the Hamming distance $d_H(\boldsymbol{T}', \boldsymbol{Q})$ is included in the distribution of genuine clients. However, the universal spoofing vector $\boldsymbol{Q}'$ has Hamming distance smaller than θ, which is included in the left distribution in Fig. 1. More specifically, by (6), the spoofing vector $\boldsymbol{Q}'$ has $d_H(\boldsymbol{T}', \boldsymbol{Q}')$ satisfying either

$$d_H(\boldsymbol{T}', \boldsymbol{Q}') < (\theta + \theta') \text{ or } d_H(\boldsymbol{T}', \boldsymbol{Q}') > 2048 - (\theta + \theta'), \tag{7}$$

where θ' denotes the order of standard deviation of binomial distribution (e.g., $\theta' = 25$). To detect the spoofing attack, it only needs to test that the above condition is satisfied.

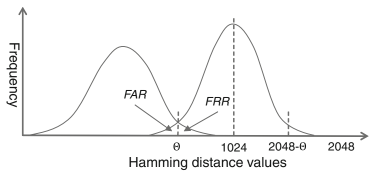


Fig. 1. Distributions in biometric authentication (the distribution on the left is of Hamming distance values from genuine clients, while the one on the right is from intruders, see also [3, Fig. 3])

3.3 Privacy Enhancing Technique

The secure matching computation (5) gives a decryptor additional information on top of the Hamming distance. In order to suppress the extra information, we consider the following additional procedure: Let ct_H denote a ciphertext given by (5). Our idea is to add random data in the ciphertext ct_H. Specifically, given $\mathsf{ct}_H = (c_0, c_1, c_2) \in (R_q)^3$ (note that it has three ring elements in R_q by the homomorphic multiplications in (5)), it generates a random polynomial $r = r_1 x + \cdots + r_{n-1} x^{n-1} \in R$ without constant term for $0 < r_i < t$, and computes a ciphertext given by one homomorphic addition $\mathsf{ct}_H \dot{+} r = (c_0 + r, c_1, c_2)$. Then the decryption of $\mathsf{ct} \dot{+} r$ includes our desired Hamming distance in the constant term, but the other terms are masked by random information r_i's.

4 Secure Protocol

Here we introduce a secure protocol based on improvements described in the previous section. Our protocols involves seven parties (see Fig. 2): Certificate Authority ***CA***, Registration Device ***RD***, Client Device ***CD***, Application Service ***ApS***, Computation Server ***CS***, Storage Server ***SS***, and finally Authentication Server ***AS***.

4.1 The Protocol

Below, (KeyGenHom, Enc, Dec, $HEval$) denotes Key Generation, Encryption, Decryption and Homomorphic Evaluation algorithms corresponding to the SHE scheme described in Sect. 2.2. (KeyGenSym, EncSym, DecSym) denotes Key Generation, Encryption and Decryption algorithms corresponding to any secure symmetric key encryption scheme. Below we present schematics of our protocol. In Figs. 2, 3 and 4 all SSL communications are authenticated by trusting root certificate authority ***CA*** in certificate chain.

Remark 1. The security of the protocol against replay attacks relies on the security of SSL channel (However, only SSL is not sufficient as we are considering insider attackers as explained in Sect. 4.2); The challenge $p \leftarrow \mathbb{Z}/t\mathbb{Z}$ is protected over communication between ***CD*** and ***AS***, and ***CS*** cannot know p. Hence ***CS***

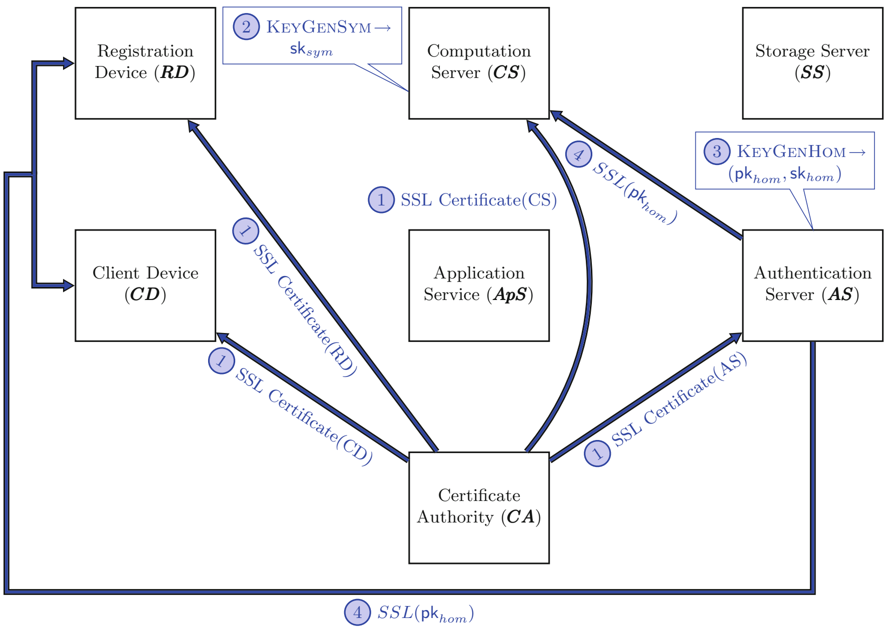

Registration Device (*RD*)
2 KeyGenSym → sk_{sym}
Computation Server (*CS*)
Storage Server (*SS*)
3 KeyGenHom → $(\mathsf{pk}_{hom}, \mathsf{sk}_{hom})$
4 SSL(pk_{hom})
1 SSL Certificate(CS)
1 SSL Certificate(RD)
Client Device (*CD*)
Application Service (*ApS*)
Authentication Server (*AS*)
1 SSL Certificate(CD)
1 SSL Certificate(AS)
Certificate Authority (*CA*)
4 SSL(pk_{hom})

Fig. 2. Setup phase. Certificate provisioning (Step 1) is trusted.

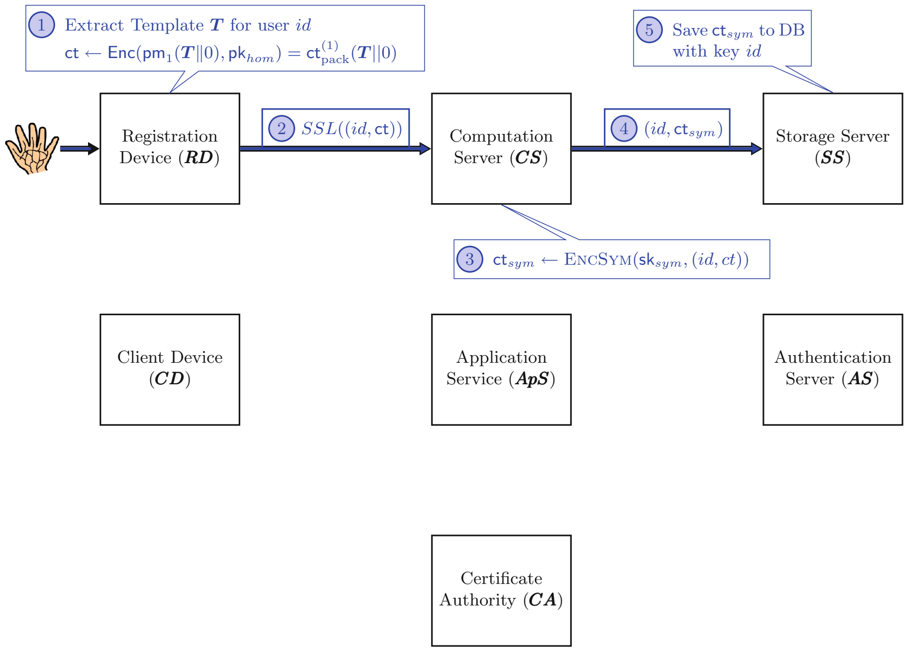

1 Extract Template $\boldsymbol{T}$ for user id
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pm}_1(\boldsymbol{T}\|0), \mathsf{pk}_{hom}) = \mathsf{ct}^{(1)}_{\text{pack}}(\boldsymbol{T}\|0)$
5 Save ct_{sym} to DB with key id
Registration Device (*RD*)
2 SSL((id, ct))
Computation Server (*CS*)
4 (id, ct_{sym})
Storage Server (*SS*)
3 $\mathsf{ct}_{sym} \leftarrow$ EncSym(sk_{sym}, (id, ct))
Client Device (*CD*)
Application Service (*ApS*)
Authentication Server (*AS*)
Certificate Authority (*CA*)

Fig. 3. Registration.

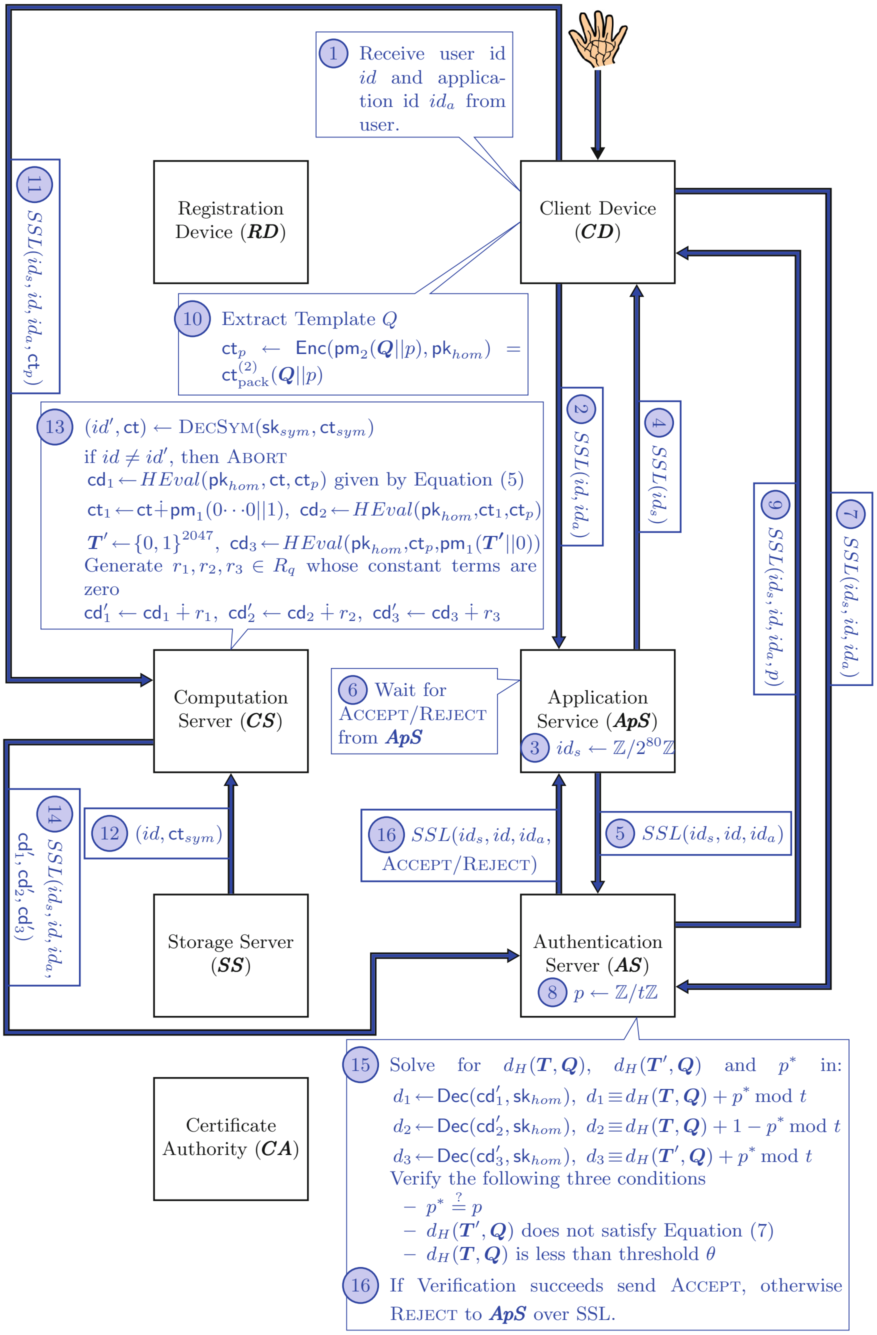

1 Receive user id id and application id id_a from user.
Registration Device (*RD*)
Client Device (*CD*)
11 $SSL(id_s, id, id_a, \mathsf{ct}_p)$
10 Extract Template Q $\mathsf{ct}_p \leftarrow \mathsf{Enc}(\mathsf{pm}_2(\boldsymbol{Q}||p), \mathsf{pk}_{hom}) = \mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}||p)$
13 $(id', \mathsf{ct}) \leftarrow \textsc{DecSym}(\mathsf{sk}_{sym}, \mathsf{ct}_{sym})$
if $id \neq id'$, then Abort
$\mathsf{cd}_1 \leftarrow HEval(\mathsf{pk}_{hom}, \mathsf{ct}, \mathsf{ct}_p)$ given by Equation (5)
$\mathsf{ct}_1 \leftarrow \mathsf{ct} \dot{+} \mathsf{pm}_1(0 \cdots 0||1)$, $\mathsf{cd}_2 \leftarrow HEval(\mathsf{pk}_{hom}, \mathsf{ct}_1, \mathsf{ct}_p)$
$\boldsymbol{T}' \leftarrow \{0,1\}^{2047}$, $\mathsf{cd}_3 \leftarrow HEval(\mathsf{pk}_{hom}, \mathsf{ct}_p, \mathsf{pm}_1(\boldsymbol{T}'||0))$
Generate $r_1, r_2, r_3 \in R_q$ whose constant terms are zero
$\mathsf{cd}'_1 \leftarrow \mathsf{cd}_1 \dot{+} r_1$, $\mathsf{cd}'_2 \leftarrow \mathsf{cd}_2 \dot{+} r_2$, $\mathsf{cd}'_3 \leftarrow \mathsf{cd}_3 \dot{+} r_3$
2 $SSL(id, id_a)$
4 $SSL(id_s)$
9 $SSL(id_s, id, id_a, p)$
7 $SSL(id_s, id, id_a)$
Computation Server (*CS*)
6 Wait for Accept/Reject from *ApS*
Application Service (*ApS*)
3 $id_s \leftarrow \mathbb{Z}/2^{80}\mathbb{Z}$
14 $SSL(id_s, id, id_a, \mathsf{cd}'_1, \mathsf{cd}'_2, \mathsf{cd}'_3)$
12 (id, ct_{sym})
16 $SSL(id_s, id, id_a,$ Accept/Reject$)$
5 $SSL(id_s, id, id_a)$
Storage Server (*SS*)
Authentication Server (*AS*)
8 $p \leftarrow \mathbb{Z}/t\mathbb{Z}$
Certificate Authority (*CA*)
15 Solve for $d_H(\boldsymbol{T}, \boldsymbol{Q})$, $d_H(\boldsymbol{T}', \boldsymbol{Q})$ and p^* in:
$d_1 \leftarrow \mathsf{Dec}(\mathsf{cd}'_1, \mathsf{sk}_{hom})$, $d_1 \equiv d_H(\boldsymbol{T}, \boldsymbol{Q}) + p^* \bmod t$
$d_2 \leftarrow \mathsf{Dec}(\mathsf{cd}'_2, \mathsf{sk}_{hom})$, $d_2 \equiv d_H(\boldsymbol{T}, \boldsymbol{Q}) + 1 - p^* \bmod t$
$d_3 \leftarrow \mathsf{Dec}(\mathsf{cd}'_3, \mathsf{sk}_{hom})$, $d_3 \equiv d_H(\boldsymbol{T}', \boldsymbol{Q}) + p^* \bmod t$
Verify the following three conditions
– $p^* \overset{?}{=} p$
– $d_H(\boldsymbol{T}', \boldsymbol{Q})$ does not satisfy Equation (7)
– $d_H(\boldsymbol{T}, \boldsymbol{Q})$ is less than threshold θ
16 If Verification succeeds send Accept, otherwise Reject to *ApS* over SSL.

Fig. 4. Authentication.

can know only the response $\mathsf{ct}_p = \mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}||p)$, and even an adversary having power to observe internal memory of $\boldsymbol{CS}$ cannot mount replay attacks since he cannot obtain $\mathsf{ct}^{(2)}_{\text{pack}}(\boldsymbol{Q}||0)$ without p. In this scheme success probability of a replay attacker is at most $1/t$. For higher security the challenger can choose $p = p_1\| \cdots \|p_k \leftarrow (\mathbb{Z}/t\mathbb{Z})^k$ to achieve $1/t^k$ security. In that case, in Step 13, $\boldsymbol{CS}$ needs to send $k + 2$ ciphertexts such that during Step 15, $\boldsymbol{AS}$ can solve $k + 2$ linear equations to recover $p_1^*, \ldots, p_k^*$, $d_H(\boldsymbol{T}, \boldsymbol{Q})$ and $d_H(\boldsymbol{T}', \boldsymbol{Q})$.

4.2 Security Model

Overview. If a stored template is leaked, an adversary cannot recover the raw template $\boldsymbol{T}$ since it is encrypted. Due to our improvements described in Sects. 3.1 and 3.2, our biometric system is secure against impersonation attacks at $\boldsymbol{CD}$, such as the replay and the universal spoofing attacks. Furthermore, as in [26], our system is secure against the hill-climbing attack to $\boldsymbol{SS}$ since the matching score is still encrypted.

Since all data are encrypted, $\boldsymbol{SS}$ cannot learn any information about client's biometric data. Furthermore, due to the improvement of Sect. 3.3, $\boldsymbol{AS}$ cannot know any extra information other than the Hamming distance between two feature codes as in [18].

Adversary Models. We will assume that $\boldsymbol{CA}, \boldsymbol{RD}$ and $\boldsymbol{ApS}$ are all trusted. The other parties $\boldsymbol{CS}, \boldsymbol{CD}$ and $\boldsymbol{SS}$ can be attributed varying levels of trust - they can be completely adversarial or can be 'semi-honest' in the sense that they perform all computations faithfully, but their memory is visible to an adversary. However, we observe that the semi-honest setting is only as secure as the completely malicious setting. This is because, any secret possessed by a semi-honest entity can be made visible to the adversary and then the adversary can completely bypass that entity. Thus any attack in the malicious setting can also be simulated in the semi-honest setting. Thus any party being adversarial versus being semi-honest actually imply the same level of security. The semi-honest setting becomes meaningfully distinct from the malicious setting, when we ensure that communication integrity with the party can be maintained in some way, such as protecting SSL private keys from adversary visibility by employing secure hardware.

Following Simoens et al. [22], we enumerate several attack scenarios and describe the security and privacy guarantees provided by our construction in the subsequent section. In Table 1, "security" denotes integrity of authentication, that is, authentication can only be achieved by providing the correct biometric to $\boldsymbol{CD}$. On the other hand, "privacy" denotes the weaker guarantee that the biometric template itself remains confidential.

In realistic scenarios, it may be desirable to host the $\boldsymbol{CS}$ in a public cloud. In that case assuming that the $\boldsymbol{CS}$ is completely honest or equivalently semi-honest (as above) may be too strong. A reasonable compromise may be to assume that the $\boldsymbol{CS}$ is semi-honest, except that it has an SSL module which is completely honest. This ensures that communications with the $\boldsymbol{CS}$ are authenticated, thus

Table 1. Security guarantees under various attack models

Adversary	Security	Justification	Privacy	Justification
AS	No	Trivially insecure	Yes (*)	Data encrypted with sk_{sym}. Hill climbing mitigated by authenticating $\boldsymbol{CD}$. (*) Hill climbing possible if any $\boldsymbol{CD}$ is completely compromised
SS	Yes	Data encrypted with sk_{sym}	Yes	Data encrypted with sk_{sym}
CS	No	Can deviate from correct homomorphic evaluation	Yes	Data encrypted with pk_{hom}
AS + SS	No	Inherited from $\boldsymbol{AS}$	Yes	Data encrypted with sk_{sym}
AS + CS	No	Inherited from $\boldsymbol{AS}$	No	$\boldsymbol{CS}$ and $\boldsymbol{AS}$ can decrypt data using sk_{sym} and sk_{hom} in sequence
CS + SS	No	Inherited from $\boldsymbol{CS}$	Yes	Data encrypted with pk_{hom}
AS + CS + SS	No	Inherited from $\boldsymbol{AS}$	No	Inherited from $\boldsymbol{AS}$+$\boldsymbol{CS}$
Γ	Yes	See text	Yes	See text

mitigating bypassing attacks which could have access to SSL private keys in the pure semi-honest model.

In general if a Client Device $\boldsymbol{CD}$ is compromised, then it can just read off the user biometric and reuse it afterward, and hence neither security nor privacy is protected. Thus $\boldsymbol{CD}$ being malicious is too strong to protect against. However, it may still be desirable to protect against adversaries having physical access to $\boldsymbol{CD}$, in particular, which can present fake biometrics to $\boldsymbol{CD}$. While online hill climbing attacks may be realistic to perform if such an opportunity exists, we can assume that hill climbing by producing appropriate biometrics physically to $\boldsymbol{CD}$ is still infeasible. To ensure that the adversary is indeed physically accessing $\boldsymbol{CD}$ and not performing online hill climbing attack, which is possible when $\boldsymbol{AS}$ is adversarial, we needed to provision $\boldsymbol{CD}$ with trusted SSL private keys. Any other biometric authentication system which exposes the hamming distance to $\boldsymbol{AS}$, is susceptible to similar hill climbing attacks unless all $\boldsymbol{CD}$'s are provisioned with trusted SSL private keys.

To balance between security, privacy and feasibility, we therefore also define the following adversary model, referred to as Γ: $\boldsymbol{CA}, \boldsymbol{RD}, \boldsymbol{ApS}$ and $\boldsymbol{AS}$ are all trusted. $\boldsymbol{SS}$ is adversarial. $\boldsymbol{CS}$ has two components; (1) an SSL communication module which is trusted, (2) a computation module which is semi-honest (memory of $\boldsymbol{CS}$ is visible to an adversary). Finally, $\boldsymbol{CD}$ is also trusted, but the adversary is allowed to physically access $\boldsymbol{CD}$, i.e., it can present fake biometrics to $\boldsymbol{CD}$.

Table 2. Performance of our secure protocol

Protocol phase	Time
(a) Challenge	≈0 ms
(b) Response	3.65 ms
(c) Secure matching	10.62 ms
(d) Decryption	10.41 ms
Total time of auth. phase	24.68 ms

The security and privacy properties of our protocol in various adversary models are summarized in Table 1. In particular, we claim that our protocol is secure in the adversary model Γ. An intuitive justification is as follows. We observe that $\boldsymbol{CS}$ cannot be bypassed as its SSL module is trusted. Although the adversary is able to observe the memory of $\boldsymbol{CS}$, Computations at $\boldsymbol{CS}$ take place homomorphically. Hill climbing by adversary physically accessing $\boldsymbol{CD}$ is infeasible as it only sees Yes/No answers. If we consider a stronger assumption that encryptions of $\mathsf{pm}_1(\boldsymbol{T})$ are infeasible to convert into encryptions of $\mathsf{pm}_2(\boldsymbol{Q})$ such that $d_H(\boldsymbol{T}, \boldsymbol{Q})$ is small, then we can claim a stronger security guarantee as well: even if several $\boldsymbol{CD}$'s are completely compromised by the attacker, but the user in question presents biometrics to an honest $\boldsymbol{CD}$, then also security and privacy are both preserved in model Γ.

Much of the complexity of our protocol arises in order to segregate trust among $\boldsymbol{AS}$ and $\boldsymbol{CS}$. From Table 1, we observe that when one of $\boldsymbol{AS}$ and $\boldsymbol{CS}$ is malicious, but the other one is trusted, security fails to hold, but privacy of the templates is protected, even against online hill climbing attacks. However, privacy no longer holds if both parties are malicious.

5 Implementation Evaluation

For our secure protocol described in Sect. 4.1, we implemented the SHE scheme and used standard SSL and AES algorithms. For the SHE scheme, we used $(n, q, t, \sigma) = (2048, 63\text{-bit}, 2048, 4)$, which can give at least 80-bit security. Our experiments ran on an Intel Xeon X3480 at 3.07 GHz with 16 GB memory, and we used our library written in C language for all computations of the SHE scheme. Our C code was compiled with `gcc 4.6.0` on Linux.

In Table 2, we summarize the performance of our secure protocol. We remark that in Table 2 we show only the computation cost of the SHE scheme since the computation cost of the standard SSL and AES algorithms is not dominant compared to the SHE scheme (we note that the performance of the SSL channels mainly depends on network throughput).

Performance Comparison. Here we compare the performance of our protocol with related work introduced in Sect. 1.2 (Unfortunately, no implementation result is reported in [6,15,16]). SCiFI [19] and the protocol [2] use additively homomorphic encryption for secure matching. In [19] (resp. [2]), it took 310 ms

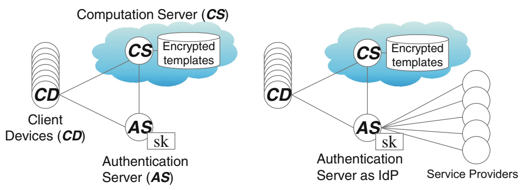


Fig. 5. A typical application of our secure biometric system

(resp. 150 ms) for secure Hamming distance of 900-bit (resp. 2048-bit) feature codes over an 8 core machine of 2.6 GHz AMD Opteron processors (resp. an Intel Core 2 Duo 2.13 GHz). Our protocol is at least 5 times faster than [2,19]. Furthermore, while [2,19] can only protect one feature code due to the usage of an additive scheme, ours can protect both $\boldsymbol{T}$ and $\boldsymbol{Q}$. On the other hand, the protocol of [18] uses the BGN scheme [4] for protection of two feature codes as in our work. According to [18], it took 58 s for secure Hamming distance of 2048-bit over an Intel 3.2 GHz PC. It is much slower than our protocol, and the main reasons are as follows; The homomorphic multiplication in the BGN scheme is slower than that in the SHE scheme, and we cannot use the packing method described in Sect. 2.3 in the BGN scheme. Therefore, in the BGN scheme, we have to use bit-wise encryption and it requires at least 2048 homomorphic multiplications for secure Hamming distance computation (cf. ours requires only a few homomorphic multiplications).

Our secure protocol gives an improvement over [26] for impersonation attacks such as replay and spoofing attacks. According to [26], it takes 3.65 ms to encrypt a biometric feature vector of 2048-bit, 5.31 ms for secure Hamming distance computation, and 3.47 ms for decryption. The total performance is about 12.43 ms for authentication, and it is about twice faster than ours. This is because, compared to the simple protocol of [26], our protocol needs to compute three each of the secure Hamming distance homomorphic evaluations and decryptions instead of just one each.

6 Conclusion and Typical Applications

In a general biometric system the authentication server must securely manage clients' templates and perform matching computation for every authentication request. Large-scale applications require large template storage and vast computation resources. In contrast, our system enables the authentication server to *securely outsource* the template storage and the matching computation resources, for example, to the cloud. In our system, since the biometric templates and matching computation are in encrypted format, the cloud cannot learn any information. Our system can prevent a malicious client from trying impersonation attacks such as replay and universal spoofing attacks. The authentication server only needs to store the secret key securely and perform decryption procedure for authentication decision (as for privacy, the authentication server can only know matching scores). Furthermore, our system has practical performance, and it is (much) faster than the state-of-the-art work using homomorphic encryption.

As Fig. 5 shows, our system is suitable for remote authentication over the Internet, and enables secure match-on-server using the cloud; The computation server ***CS*** can be deployed as a cloud service, while the authentication server ***AS*** with secret key sk is placed on a private platform (Fig. 5, left). To serve a large number of parties who demand results from biometric authentication, a trusted identity provider (IdP) can act as ***AS*** (Fig. 5, right). Match-on-server authentication is suitable for personal identification and usage of multiple clients. Specifically, identity federation protocols (e.g., *SAML* and *OpenID Connect*), which are used between the IdP and the Service Providers, benefit from identities authenticated with biometrics.

References

1. Jain, A.K., Nandakumar, K., Nagar, A., et al.: Biometric template security. EURASIP J. Adv. Signal Process. **113** (2008)
2. Blanton, M., Gasti, P.: Secure and efficient protocols for iris and fingerprint identification. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 190–209. Springer, Heidelberg (2011)
3. Bolle, R.M., Connell, J.H., Ratha, N.K.: Biometric perils and patches. Pattern Recognit. **35**(12), 2727–2738 (2002)
4. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
5. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
6. Bringer, J., Chabanne, H.: An authentication protocol with encrypted biometric data. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 109–124. Springer, Heidelberg (2008)
7. Bringer, J., Chabanne, H., Izabachène, M., Pointcheval, D., Tang, Q., Zimmer, S.: An application of the Goldwasser-Micali cryptosystem to biometric authentication. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 96–106. Springer, Heidelberg (2007)
8. Campisi, P.: Security and Privacy in Biometrics. Springer, New York (2013)
9. Damgård, I., Geisler, M., Krøigård, M.: Homomorphic encryption and secure comparison. Int. J. Appl. Cryptogr. **1**(1), 22–31 (2008)
10. Daugman, J.: The importance of being random: statistical principles of iris recognition. Pattern Recognit. **36**(2), 279–291 (2003)
11. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
12. Evans, D., Huang, Y., Katz, J., Malka, L.: Efficient privacy-preserving biometric identification. In: 17th Proceedings of the Network and Distributed System Security Symposium-NDSS, 2011 (2011)
13. Fujitsu Laboratories Ltd. Press release: Fujitsu develops world's first authentication technology to extract and match 2,048-bit feature codes from palm vein images (2013)

14. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing-STOC, pp. 365–377. ACM (1982)
15. Hattori, M., Matsuda, N., Ito, T., Shibata, Y., Takashima, K., Yoneda, T.: Provably-secure cancelable biometrics using 2-DNF evaluation. J. Inf. Process. **20**(2), 496–507 (2012)
16. Hirano, T., Hattori, M., Ito, T., Matsuda, N.: Cryptographically-secure and efficient remote cancelable biometrics based on public-key homomorphic encryption. In: Sakiyama, K., Terada, M. (eds.) IWSEC 2013. LNCS, vol. 8231, pp. 183–200. Springer, Heidelberg (2013)
17. Izu, T., Sakemi, Y., Takenaka, M., Torii, N.: A spoofing attack against a cancelable biometric authentication scheme. In: Advanced Information Networking and Applications-AINA. IEEE (2014)
18. Kulkarni, R., Namboodiri, A.: Secure hamming distance based biometric authentication. In: International Conference on Biometrics (ICB), pp. 1–6. IEEE (2013)
19. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: SCiFI-a system for secure face identification. In: IEEE Symposium on Security and Privacy (SP), pp. 239–254. IEEE (2010)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Rathgeb, C., Uhl, A.: A survey on biometric cryptosystems and cancelable biometrics. EURASIP J. Inf. Secur. **2011**(1), 1–25 (2011)
22. Simoens, K., Bringer, J., Chabanne, H., Seys, S.: A framework for analyzing template security and privacy in biometric authentication systems. IEEE Trans. Inf. Forensics Secur. **7**(2), 833–841 (2012)
23. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.: Blind authentication: a secure crypto-biometric verification protocol. IEEE Trans. Inf. Forensics Secur. **5**(2), 255–268 (2010)
24. U.S. Department of Homeland Security. Privacy impact assessment for the biometric storage system (2007)
25. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Packed homomorphic encryption based on ideal lattices and its application to biometrics. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) cd-ares workshops 2013. LNCS, vol. 8128, pp. 55–74. Springer, Heidelberg (2013)
26. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Practical packing method in somewhat homomorphic encryption. In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W.M. (eds.) DPM 2013 and SETOP 2013. LNCS, vol. 8247, pp. 34–50. Springer, Heidelberg (2014)

On the Privacy of Horizontally Partitioned Binary Data-Based Privacy-Preserving Collaborative Filtering

Murat Okkalioglu[1], Mehmet Koc[2], and Huseyin Polat[3](✉)

[1] Department of Computer Engineering, Yalova University, 77200 Yalova, Turkey
murat.okkalioglu@yalova.edu.tr

[2] Department of Electrical and Electronic Engineering, Bilecik Seyh Edebali University, 11210 Bilecik, Turkey
mehmet.koc@bilecik.edu.tr

[3] Department of Computer Engineering, Anadolu University, 26555 Eskisehir, Turkey
polath@anadolu.edu.tr

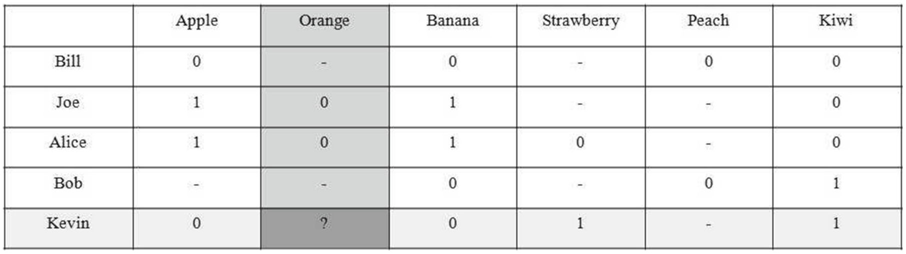
Abstract. Collaborative filtering systems provide recommendations for their users. Privacy is not a primary concern in these systems; however, it is an important element for the true user participation. Privacy-preserving collaborative filtering techniques aim to offer privacy measures without neglecting the recommendation accuracy. In general, these systems rely on the data residing on a central server. Studies show that privacy is not protected as much as believed. On the other hand, many e-companies emerge with the advent of the Internet, and these companies might collaborate to offer better recommendations by sharing their data. Thus, partitioned data-based privacy-persevering collaborative filtering schemes have been proposed. In this study, we explore possible attacks on two-party binary privacy-preserving collaborative filtering schemes and evaluate them with respect to privacy performance.

Keywords: Privacy · Collaborative filtering · Binary data · Attack scenarios

1 Introduction

Information technologies have been developing at a great pace and providing comfort for everyone to perform their everyday tasks. The Internet is a great medium to offer countless services due to its ease of access by people from every different taste and class. This trend of moving people's routines to the Internet has made e-commerce very attractive. However, attracting new customers or recommending the right products for the existing ones is a competitive task for an online company to survive in the market. Collaborative filtering (CF) is a technique for providing referrals. It was proposed with the Tapestry project [7]. CF systems aim to produce recommendations for an active user (a) who asks for a prediction by collecting user opinions (ratings) and considering these opinions

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 199–214, 2016.
DOI: 10.1007/978-3-319-29883-2_13



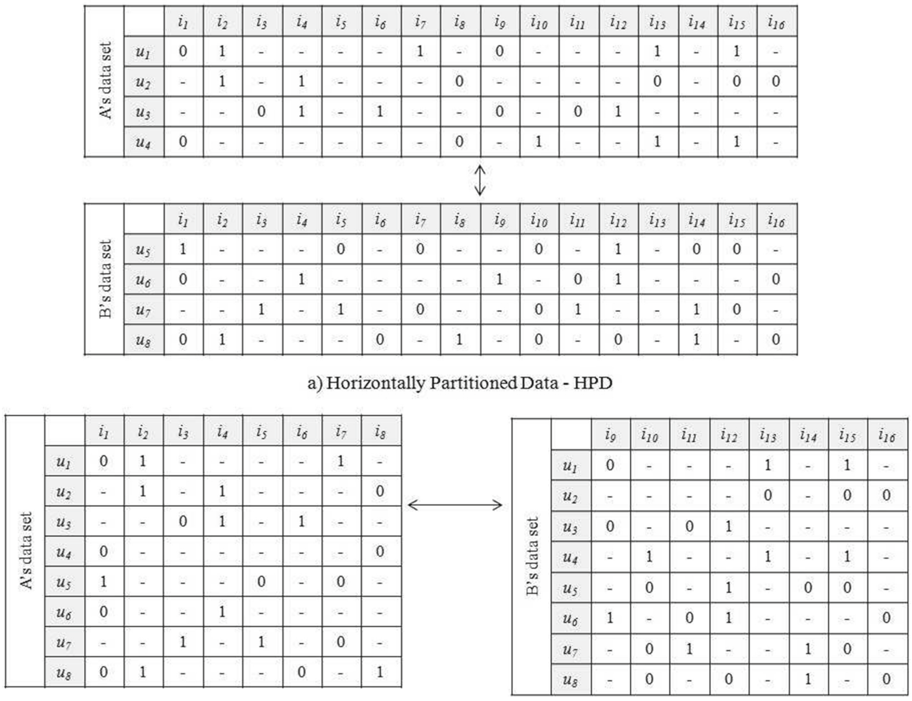
	Apple	Orange	Banana	Strawberry	Peach	Kiwi
Bill	0	-	0	-	0	0
Joe	1	0	1	-	-	0
Alice	1	0	1	0	-	0
Bob	-	-	0	-	0	1
Kevin	0	?	0	1	-	1

Fig. 1. A typical CF system

with similar interest for referrals for a [11,25]. A typical CF scheme is composed of an nxm user-item matrix, where n different users express their opinions on m different items as seen in Fig. 1. User opinions can be expressed using binary ratings showing if the item is *liked* (1) or *disliked* (0).

True participation is critical for CF schemes to produce accurate referrals. However, users might be unwilling to participate in them due to privacy concerns. User data is a valuable asset and might be subjected to different threats such as unsolicited marketing like unwanted e-mails or phone calls, government surveillance, price discrimination, or even subpoena [5,6]. Privacy-preserving collaborative filtering (PPCF) addresses privacy. If a user believes that individual privacy is promised by a CF scheme, she will be eager to be part of it and to provide true ratings. On the other hand, privacy cannot be considered the sole purpose of PPCF. The schemes must also produce accurate referrals. Thus, the objective of PPCF schemes can be considered as balancing equilibrium between privacy and accuracy without neglecting performance [3]. Recently, some scholars argue that PPCF schemes do not protect individual privacy as promised [4,29]. Zhang et al. [29] propose reconstruction methods to recover numerically rated data, which is perturbed by randomization [21]. Calandrino et al. [4] study inference attacks on online CF systems.

Data sparsity is one of the main challenges for CF systems [12,26]. In general, CF systems operate on a large user-item matrix and this matrix is usually extremely sparse because users vote on a relatively small number of items based on their interest. Therefore, companies might lack accurate recommendations if they have insufficient data. The missing ratings of a user-item matrix can be compensated if data is partitioned between parties. When data is partitioned between any two parties, the parties can fill some missing ratings and better recommendation can be achieved. Data can be partitioned horizontally, vertically, or arbitrarily. In horizontally partitioned data (HPD) schemes, two parties having ratings for the same set of items by different users share their data while ratings for different items from the same set of users are shared in vertically partitioned data (VPD). HPD- and VPD-based schemes are displayed in Fig. 2. The users are displayed by $<u_1, u_2, \ldots.., u_7, u_8>$ and the items are displayed as $<i_1, i_2, \ldots.., i_{15}, i_{16}>$. In arbitrarily partitioned data, both parties share arbitrarily ratings for items and users.



A's data set

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
u_1	0	1	-	-	-	-	1	-	0	-	-	-	1	-	1	-
u_2	-	1	-	1	-	-	-	0	-	-	-	-	0	-	0	0
u_3	-	-	0	1	-	1	-	-	0	-	0	1	-	-	-	-
u_4	0	-	-	-	-	-	-	0	-	1	-	-	1	-	1	-

B's data set

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
u_5	1	-	-	-	0	-	0	-	-	0	-	1	-	0	0	-
u_6	0	-	-	1	-	-	-	-	1	-	0	1	-	-	-	0
u_7	-	-	1	-	1	-	0	-	-	0	1	-	-	1	0	-
u_8	0	1	-	-	-	0	-	1	-	0	-	0	-	1	-	0

a) Horizontally Partitioned Data - HPD

A's data set

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	0	1	-	-	-	-	1	-
u_2	-	1	-	1	-	-	-	0
u_3	-	-	0	1	-	1	-	-
u_4	0	-	-	-	-	-	-	0
u_5	1	-	-	-	0	-	0	-
u_6	0	-	-	1	-	-	-	-
u_7	-	-	1	-	1	-	0	-
u_8	0	1	-	-	-	0	-	1

B's data set

	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
u_1	0	-	-	-	1	-	1	-
u_2	-	-	-	-	0	-	0	0
u_3	0	-	0	1	-	-	-	-
u_4	-	1	-	-	1	-	1	-
u_5	-	0	-	1	-	0	0	-
u_6	1	-	0	1	-	-	-	0
u_7	-	0	1	-	-	1	0	-
u_8	-	0	-	0	-	1	-	0

b) Vertically Partitioned Data - VPD

Fig. 2. Partitioned data schemes in CF

This study focuses on the HPD-based PPCF scheme with binary ratings proposed by Polat and Du [22,24]. The authors propose PPCF schemes with *threshold* or *best-N* neighbors based methods. Kaleli and Polat [14] propose a two-party binary PPCF scheme based on naïve Bayes classifier. In these studies [14, 22,24], two-party binary PPCF schemes are only examined in terms of accuracy. However, attacks that can be applied to these schemes are not studied. In this study, we *discuss attack scenarios applicable to HPD schemes in* [22,24] *and perform attacks against HPD-based two-party binary PPCF schemes to examine how much privacy is offered by these schemes.*

The rest of the paper is organized as follows. Section 2 presents related work in the literature. Section 3 discusses the two-party PPCF schemes to inform the reader. In Sects. 4 and 5, we present possible attack scenarios and perform our experiments, respectively. In Sect. 6, final comments and future works are presented.

2 Related Work

Privacy is an important issue for customers. Since customers worry about the privacy, they may not want to share their ratings or they provide fake ratings to hide their true profiles. Although privacy-preserving methods offer privacy measures to protect individual privacy, researchers argue that they are not successful as promised [16,17]. The authors in [16,17] show that the original data can be closely estimated from the perturbed data using *spectral filtering* (SF). They propose random matrix based SF technique to reveal the data from the perturbed one. Huang et al. [13] propose a data reconstruction technique based on *principal component analysis* (PCA) to derive the correlation structure of the original data. If noise is added to the original data, the method tries to remove the noise using the correlation structure of the original data. This procedure ends with the values which are close to the original ones. Guo et al. [10] give a lower bound and an upper bound for the reconstruction error when the SF techniques are applied to derive the original data. Using these bounds the attackers can decide the quality of their attacks [10]. Agrawal and Aggarwal [1] propose a data reconstruction algorithm based on *expectation maximization* that converges the maximum likelihood estimate of the original distribution, which is an extension of the study presented in [2].

Guo et al. [9] propose an attack based on *interquantile-range*, which is used to measure the amount of the variability of the random variable. Attackers can use this approach to estimate the range of each individual data with a confidence interval. In [29], the authors propose a reconstruction technique targeting a numerically rated PPCF scheme proposed in [21]. Their method use *k-means* clustering and *singular value decomposition* (SVD) to derive original ratings from perturbed data. Calandrino et al. [4] devise different attacks on CF systems. They utilize auxiliary data and make inferences about target live CF systems. They propose attacks including *knn-based* attack, which is also exploited in this study, based on temporal changes of public outputs of the commercial web sites. In [19], the authors propose a reconstruction technique to recover binary rated data masked by using randomized response techniques [27] offered by Polat and Du [23]. The same data disguising method also introduce some fake ratings to enhance privacy and the scholars in [20] analyze this scheme in order to discover fake binary rated items.

In all of the above studies, the attacks proposed by the authors to recover the original ratings are for single party. Unlike these methods, we examine three different attack scenarios to the HDP-based two-party binary PPCF schemes proposed by Polat and Du [22,24] to discover the ratings.

3 Horizontally Partitioned Binary Data-Based Privacy-Preserving Collaborative Filtering

Dense ratings are important factors for PPCF systems in terms of accurate recommendations since the denser the ratings are, the likelier it is to discover the relationships among items or users. Small companies or start up might have sparse ratings due to the limited number of users [3,15,24,28]. Even some companies might be interested in migrating to another business segment and they might be in need of the related item ratings for the new market. Such companies could find a partner to exchange their ratings without neglecting privacy promises. Essentially, companies can overcome the problem of data sparsity if they share their data with a partner. One party could have ratings for a set of items/users that the collaborating party does not have. Thus, data sharing will be mutually beneficial for both parties to produce better recommendations on the joint data. There are various partitioned data-based PPCF schemes with binary ratings like HPD- and VPD-based ones [14,22,24]. In this study, various attack scenarios have been performed to test the privacy offered by the HPD-based PPCF schemes [22,24]. Before giving the details of these scenarios we would like to introduce these binary data-based PPCF schemes. Assume that one of the parties is A and the other party is B.

3.1 Preliminaries

In general, two-party binary ratings-based PPCF schemes proposed by Polat and Du [22,24] provide *top-N* (TN) results for users. TN schemes return the first N items matching with earlier user preferences as a referral. Their schemes (both HPD- and VPD-based) find a set of neighbors for a based on similarities between a and other users. Remember that a is an active user who is looking for a prediction. Additionally, Polat and Du [22,24] use a protocol to compute similarities without jeopardizing the privacy of the users. In this subsection, the similarity metric and the protocol used to privately compute the similarities are given as a preliminary to the PPCF schemes.

Similarity Metric. This metric is used to calculate how much a user is similar or dissimilar to a. The metric used by Polat and Du [22,24] is based on the difference of similarly rated items from dissimilarly rated items over the commonly rated items. They use this similarity metric to identify the neighbors. The formulation of this metric is given below:

$$W_{au} = \frac{t\left(R_s\right) - t\left(R_d\right)}{t\left(R\right)} \quad . \tag{1}$$

in which u is the related user whose similarity between a is calculated, W_{au} is the similarity between u and a, $t(R_s)$ is the number of similarly rated items, $t(R_d)$ is the number of dissimilar rated items, and *t(R)* is the number of the commonly rated items ($t(R) = t(R_s)+t(R_d)$). W_{au} can range between 1 and -1. If $W_a u > 0$, then u and a are similar, otherwise they are dissimilar. Upon determining W_{au} values, the neighbors are selected based on the criterion (*best-N* or *threshold*) in the relevant PPCF scheme given in the following subsections. If a dissimilar user is selected as one of the neighbors, her ratings are reversed discussing that dissimilar users have a negative correlation and would vote opposite for the same item. After selecting the neighbors, Polat and Du [22,24] find the number of *likes* (l_j) and *dislikes* (d_j), where j is the item number, among the selected neighbors. Then ld_j = l_j - d_j is calculated. If $ld_j > 0$, then the item will be liked by a. Otherwise, it will be disliked by a.

Private Similarity Computation Protocol PSCP. PSCP is used to compute the similarities without compromising a's privacy [22,24]. Assume that M is the number of rated items by a. This protocol has two cases, where the first case fills some unrated entries while the second case removes some rated entries. The first case is applied if M is initially less than $m/2$; remember that m is the number of items. In this case, unrated entries of a is filled with R_{Aa} number of items, where R_{Aa} is a uniform random number picked from the range (1, m-M). The chosen unrated entries are filled with default votes, where default votes are estimated for each cell based on ld_j values. In the second case, if M initially is greater than $m/2$, then a uniform random number, R_{Ar} is picked from the range (1, M), and R_{Ar} number of ratings are canceled (removed) from a's rating vector.

3.2 HPD-based Privacy-Preserving Schemes

Threshold-Based. This method is proposed by Polat and Du [22,24]. Once a sends a query, which includes the set of unrated items (N_a) that a is looking for TN, both parties start to collaborate to produce TN predictions. Collaborating parties calculate the similarities between its users and a. They select their neighbors on a predefined threshold (τ_n). The scheme is explained as follows:

1. a sends a query to both parties.
2. A calculates similarities between its users and a. Then, the neighbors are selected based on τ_n. A adds a uniform random number to τ_n to prevent B from learning τ_n.
3. A calculates $ld_{Aj} = l_{Aj} - dAj$ values for items $j = 1.N_a$ where N_a is the set of queried items from selected neighbors. Recall that a's query includes N_a unrated items among which she is looking for a prediction. A sends ld_{Aj} values to B through a.
4. B calculates similarities of its users and a; and selects neighbors based on τ_n. Then, B finds $ld_{Bj} = l_{bj} - d_bj$. It computes final ld_j values by adding corresponding ld_{Aj} and ld_{Bj} values. Then, B sorts them. TN of the sorted items are sent to a.

Best N_n-based. This scheme selects the best N_n number of neighbors after the similarities have been calculated [22,24]. PSCP is applied to perform the similarity computations privately. Details of the scheme are as follows:

1. a sends a query to both parties.
2. A finds the similarities (W_{Aua}) between its users, u, and a using PSCP. The similarities (W_{Aua}) are permuted by a function (π_A) only known to A and they are converted to their absolute values ($|W_{Aua}|$) and sent to B through a.
3. B calculates the similarity values W_{Bau} and finds the best N_n neighbors among all neighbors including A's.
4. B calculates ld_{Bj} values for N_a items and sends them to A with the selected neighbors of A.
5. A finds ld_{Aj} values for N_a items and calculates final ld_j values. It then finds TN referrals and sends them to a.

4 Attack Scenarios

Some attack scenarios can be utilized to derive information from two-party binary PPCF schemes; such schemes apply privacy measures arguing that they annihilate possible attacks [22,24]. By privacy measures, we mean the effort being made to prevent the other party from learning unintended information. Privacy has two aspects [23]; (1) to preserve the rating value of users, (2) to disguise if an item is rated or unrated. We will call them as the first and the second aspect of privacy, respectively. In this section, possible attack scenarios applicable to such schemes are discussed. We also explain the reasons of the applicability of each individual attack if no measures are taken in terms of privacy. Moreover, which aspect of privacy can be exploited by each attack will be given. In Sect. 6, we discuss how privacy measures are effective by comparing the outcomes to both cases.

4.1 Acting as an Active User in Multiple Scenarios

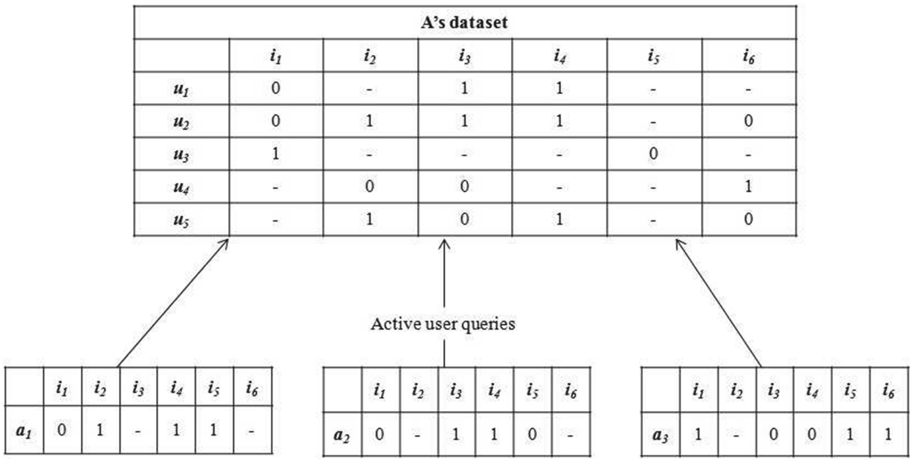
Any party who wants to derive data from the other party constructs a rating vector and acts as an active user in multiple scenarios. The active party employs the same rating vector for multiple times by only manipulating one entry at a time. Tracking changes in the similarity values, or interim results between each query, the active party can figure out the rating for the manipulated entry. Note that the change in interim result occurs due to the manipulated entry. Repeating this process for each rating will help the active party learn true ratings for all items in the rating vector. This attack discloses actual ratings of the target users and if an item is rated; therefore, it exploits the first and the second aspect of privacy.

Threshold-based scheme has only one interaction between parties on aggregate values of ld_{Aj} and it does not disclose any individual information about users. However, this attack tracks temporal changes of individual variables on responses for altered queries. Thus, it is not applicable for the *threshold-based* scheme. The *best-N_n* approach includes privacy measures to prevent B from discovering information. Similarities calculated by A are sent to B. If there are no privacy measures, this transaction would be subjected to this attack by tracking changes in the similarities each time a rating is manipulated. First, we eliminate the measures to see the success of this attack without privacy measures. Then, privacy measures are introduced to see how much effective they are. Note that B acts as an active user and sends multiple queries to A to derive data.

4.2 Perfect Match Attack

Polat and Du [22,24] discuss only *acting as an active user* attack. However, their schemes make frequent interactions between parties by exchanging the similarities or the partial similarity values required for the similarity calculation. Exchanging such values might cause information disclosure. In a typical scenario, where B is the master site and there is no privacy concern, A calculates similarities and sends them to B. If the similarity between any user of A and a is 1 or -1, this means that these users are perfectly matched and such similarities will be called *perfect match*. Every commonly rated entry of a *perfect match* either exactly matches if similarity is 1 or oppositely matches if similarity is -1. By 'opposite', bitwise NOT operation is implied. However, there might be some unrated entries from both sides; those entries are not taken into consideration while calculating similarities by Polat and Du [22,24]. Remember that the similarity metric is based on the similarly and dissimilarly ratings and having a *perfect match* means that the commonly rated entries are rated identically. Thus, B concludes that the corresponding user of a *perfect match* in A's data set has either similarly voted or not voted of a rating in a's vector if the similarity is 1. If the similarity is -1 in a *perfect match*, B concludes that the corresponding user voted opposite or not voted.

Figure 3 displays three different queries from B to A. Pay special attention to u_1, a_1 and a_2. The similarity between u_1 and these two active users are 1, although they have different ratings. Once B finds out that u_1 and a_1 is a *perfect match*, it concludes that rated entries of a_1 are either rated identically or not rated by u_1. u_1 and a_2 is another *perfect match* and the same assumption holds. Remember that a_1 and a_2 have different rating vectors. For example, i_5 is rated 1 and 0 by a_1 and a_2, respectively. Thus, B can figure out that i_5 is unrated by u_1 because each commonly rated item of a *perfect match* needs to be identical. In this case, rating vectors of a_1 and a_2 have different values for i_5. Different queries with a *perfect match* reveal unrated entries of the related user if there are opposite overlapping ratings in the queries.



A's dataset						
	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	-	1	1	-	-
u_2	0	1	1	1	-	0
u_3	1	-	-	-	0	-
u_4	-	0	0	-	-	1
u_5	-	1	0	1	-	0

	i_1	i_2	i_3	i_4	i_5	i_6
a_1	0	1	-	1	1	-

	i_1	i_2	i_3	i_4	i_5	i_6
a_2	0	-	1	1	0	-

	i_1	i_2	i_3	i_4	i_5	i_6
a_3	1	-	0	0	1	1

Fig. 3. An example of different queries from B to A

This attack discloses two privacy breaches: (1) even a single *perfect match* reveals that the actual rated value of the target item or it is unrated; (2) unrated entries can be disclosed if multiple *perfect matches* occur. To be more precise for the first vulnerability with our example, when a *perfect match* is captured between u_1 and a_1, B finds that i_1, i_2, i_4, and i_5 are either rated 0, 1, 1, and 1 by u_1, respectively, or they are unrated. When a_2 is queried, the second vulnerability occurs and i_5 is discovered to be unrated. *Perfect match* threatens the first and the second aspect of privacy because it might both disclose the possible rating value and if an item is rated or not.

This attack is applicable for the $best{-}N_n$ scheme. Note that the similarities of A are sent to B and B can exploit this information. However, the *threshold-based* scheme does not exchange such information; so, this attack is not applicable.

4.3 *knn*-based Scenario

knn-based attacks target the CF schemes that are based on neighboring approach if a history of a user is known [4]. It is claimed that if the history of ratings of a user is known, then the attacker can insert k fake users into the CF system. When a recommendation is queried for any of the fake users, it is highly possible that the neighbors will be selected among the *k-1* fake users and the target user. Thus, any item in the prediction, which is not in the known history of the target user, will be probably coming from the rating of the target user. *knn*-based attack only discloses the information of whether an item belongs to the target user or not, so it poses a threat to the second aspect of privacy.

This attack holds if the related scheme is based on selecting k neighbors. *Threshold-based* and $best - N_n$ schemes utilize the best neighboring approach. But, in the *threshold-based* scheme, we cannot determine how many neighbors will be selected, because neighborhood is determined on τ_n value. There is not a single k value to determine the number of neighbors. Hence, this attack might be applicable for the *threshold-based* scheme if some numbers of fake users are inserted. On the other hand, it is known that how many neighbors will be picked in the $best - N_n$ scheme. k fake users can be inserted into the data set. Being able to insert such fake users makes this attack possible.

5 Experiments

Experiments are performed using MovieLens Million (MLM) and Jester data sets. MLM was collected by GroupLens research group[1]. It contains 1,000,209 ratings from 6,040 users and 3,952 items. The density of the data set is about 4.2 %. Ratings are on a 5-star scale. Jester is a web-based joke recommendation system[2] [8]. It is a fairly dense data set (72.47 %) and ratings are continuous scale between -10 and 10.

[1] www.cs.umn.edu/research/GroupLens.
[2] http://eigentaste.berkeley.edu/dataset/.

Precision and *recall* have been used as evaluation criteria in this study. *Precision* is the percentage of classified items that are relevant while *recall* is the percentage of relevant items. *Precision* and *recall* value are calculated cumulatively.

This study focuses on binary rated PPCF schemes. Ratings are converted to binary scale based on a threshold [18]. MLM ratings greater than 3 are marked as *like* (1) and *dislike* (0) for the rest. Jester is labeled 1 if ratings is greater than 2.0 and labeled 0 if ratings are less than or equal to 2.0. Experiments are repeated 100 times. 500 and 2,000 users are picked randomly from MLM and Jester, respectively. Jester users are picked among the users with at least 60 % rated cells for a denser data set. During our experiments, we permute users and items in HPD, to eliminate density advantages of any party.

5.1 Experiments

Experiment I. This experiment is based on acting as an active user in multiple scenarios. Our aim in this experiment is to show how privacy measures help protect individual privacy when compared to the total disclosure.

An active query is created with the random density between 10 % and 50 % and it is sent to A. As privacy measures, PSCP and the absolute values of similarities are applied. Remember that PSCP fills or cancels (removes) some ratings based on the density of incoming active query. Although the subsequent active queries are different from the first (original) active query with only one cell, some ratings are either inserted into or removed from the active query by PSCP and it invalidates this assumption in practice. In addition, PSCP makes it very difficult to observe empty ratings in A because altering the active query makes every subsequent query almost independent from each other although they are originally different from each other with only one cell. Moreover, taking absolute values for similarities misleads this attack for negative valued similarities. For example, a possible decrease in the similarity for negative values will be reflected as an increase to the absolute values of the similarity and vice versa. We expect that the result will be worse compared to scheme when privacy is not applied due to the reasons mentioned. Table 1 demonstrates the results associated with the experiment. Note that *NP* means no privacy measure is applied while *WP* means scheme is applied with privacy measures and d is the density.

As seen from Table 1, recall values are worse than precision values in NP cases. This is because recall indicates the fraction of correctly guessed items by

Table 1. Acting as an active user attack

	Best-N$_n$ NP		Best-N$_n$ WP	
	MLM d = 4.3 %	Jester d = 85.0 %	MLM d = 4.4 %	Jester d = 4.4 %
Precision	1.000	1.000	0.025	0.448
Recall	0.248	0.341	0.162	0.124

the attack. In the attack scenario, the active query is created with the random density between 10 % and 50 %. This attack does not try to guess the actual ratings for all entries; therefore, we cannot expect full recall results. As one can see in Table 1, precision and recall values are poorer in WP compared to NP. As discussed in the previous paragraph, PSCP and absolute value while calculating similarities help users protect their privacy. Recall results are slightly better for MLM data set compared to Jester in WP case although precision results for MLM is considerably worse. This means that this attack predicts the similar fraction of relevant items from both data sets either dense or sparse. On the other hand, precision value is worse for the sparse data set (MLM) because this attack tries to guess actual values of the items in the active query and those entries are possibly empty in sparse data set. As detailed in the previous paragraph, observing empty ratings are difficult in WP case. Thus, it makes precision to deteriorate when sparsity increases.

In the *best-N_n* approach, the similarities are permuted to prevent B from learning whose similarities it is dealing with. This case has not been considered for this experiment. If this case is applied, we need to consider the probability of guessing correct similarities based on users, which is 1 out of $n_A!$, where n_A is the number of users A holds.

Experiment II. In this experiment, *perfect match* attack has been performed. The density of active query is an important factor in this attack. If the active query is dense, capturing a *perfect match* becomes difficult in comparison with a sparse active query because every common rating between the active query and users' vector must match. Therefore, we first examine the best density rate among 5 %, 10 %, 20 %, and 50 % without privacy measures. We find out that 5 % density rate achieves the best results. The density rate for active queries in this experiment is 5 % for the following tests.

First, we evaluate the performance of this attack without privacy measures. Then, the change in accuracy is compared to schemes when privacy is applied. We randomly fill 5 % of the query vector with 0 s and 1s. This attack has been executed for 100 different subsequent queries for each experiment and averages are listed in the relevant tables.

Remember that *perfect match* attack reveals two kinds of privacy breaches: (1) either the actual value of the relevant item or it is unrated; (2) the unrated entries. Then, the first breach means that if the relevant item is rated, the correct rated value is discovered. We perform a coin toss to determine the actual value of the relevant item or to mark it unrated in the case of the first privacy breach. The second breach reveals that an entry is not rated. Thus, two kinds of precision and recall have been calculated for this experiment. The first is precision and recall values for the actual value of items, $prec_1$ and rec_1 (the first privacy breach); the second is the precision and recall value to determine unrated items, $prec_2$ and rec_2 (the second privacy breach). Table 2 displays the results.

Unlike the previous experiment, precision results are highly correlated with the density in NP case. $Prec_1$ results for MLM, which is very sparse, is about 0.02 while

Table 2. Perfect match attack

	$Best - N_nNP$		$Best - N_nWP$	
	MLM d=4.1 %	*Jester d=84.8 %*	*MLM d=4.2 %*	*Jester d=85.0 %*
$prec_1$	0.023	0.868	0.005	0.371
rec_1	0.106	0.297	0.003	0.003
$prec_2$	0.985	0.238	0.991	0.393
rec_2	0.298	0.521	0.027	0.019

it goes up around 0.87 when data set is fairly dense, Jester. This attack reveals limited ratings with sparse data set on $prec_1$ metric. In a sparse data set, a *perfect match* can be captured with even a single commonly rated item and the remaining items are determined as 1 or 0 based on the active query ratings. However, the most of these ratings are probably unrated due to the sparsity of MLM. On other hand, the dense data set, Jester, performs well and it is more probable that items marked as 0 or 1 by coin toss process is indeed rated.

In terms of discovering unrated items (the second privacy breach); *perfect match* attack achieves better $prec_2$ results with MLM while Jester has better rec_2 results. In this attack, if a cell is marked as unrated, it is guaranteed to be unrated unless privacy measures are applied. However, we perform a coin toss to decide the actual rating of an item and mark some items as unrated based on this coin toss process. This process of determining actual ratings deteriorates the $prec_2$ results because we mark possibly some rated items as unrated. $Prec_2$ is still very high for MLM because it is fairly sparse and marking a rated item as unrated does not affect the metric seriously. On the other hand, marking a rated item as unrated is more expensive for a dense data set for $prec_2$ metric because number of unrated cell is limited. Rec_2 is worse for MLM; this is because items classified as unrated do not form a large number when compared to the number of unrated cells in MLM due to the sparsity. Remember that *d=4.1 %* is for MLM, so MLM is filled with unrated items for about 96 % in this experiment. As mentioned earlier, recall result are not very informative.

As a privacy measure, PSCP and the absolute values of similarities are applied. Only 5 % of the active queries are filled for this experiment, the active queries will always be filled randomly with fake entries from the range (1, *m-m*.05*). In our initial tests, to pick the density of the active query, we see decrease in the result for denser active queries; therefore, decrease in $prec_1$ and rec_1 are expected in WP case. Also, taking absolute values preserves the sign information of *perfect matches* which allows discovering only positive *perfect matches*. Notice WP part in Table 2 that the privacy is almost preserved for MLM and there is a significant decrease for Jester for the first privacy breach ($prec_1$, rec_1). The second privacy breach is mostly affected by coin toss process and taking absolute values. If PSCP is applied alone without these two factors, the second aspect of the privacy would produce full $prec_2$ results. $Prec_2$ results seem slightly better than NP case in both data sets. Since less perfect matches occur with privacy measures, the coin toss process nominates less unrated items. Practically, 50 % of unrated items determined by coin toss could be rated; thus, $prec_2$ results are better compared to NP case. Conversely, there will be fewer items marked unrated causes poor rec_2. Better recall results could be achieved by more true predictions in number. Note that permutation is not applied in this experiment as in the previous one; the probability of guessing correct similarities would be 1 out of $n_A!$ if permutation is applied.

Experiment III. This experiment performs *knn*-based attack on both NP and WP cases. This attack requires the history of a user. However, HPD schemes do not have such history inherently. To overcome this issue, we assume that the attacker has the half of the history of a targeted user. This attack can be utilized for the $best - N_n$ and *threshold-based* scheme because these schemes use neighborhood for determining the similarities; however, the number of neighbors is not determined for the *threshold-based* scheme beforehand. They are picked based on τ_n. k fake users are required to be inserted in *knn*-based attack, so we insert 200 fake users to accomplish *knn* attack for the *threshold-based* schemes. In this experiment, we are looking for predictions for randomly selected 100 users and 30 items. Final precision and recall values are calculated by taking average of precision and recall values for each user.

In this experiment, we expect the denser data set, Jester, will outperform the sparse data set, MLM. When a data set is sparse, an incoming history query might match some unrelated similarities. The similarity computation only considers the commonly rated entries and even a single match will produce a *perfect match* just like inserted fake users. This means that there will be t users who have similarity of 1 (perfect match), where t $\gg$ k. Notice that our implementation selects the first k users as neighbors. Therefore, inserted fake users could fail to be picked as neighbors. However, a denser data set is more specific while determining the similarities because a denser user vector will probably have more common items with an incoming query so that the resulting similarity is more reliable. More commonly rated items means that it is more probable to have some dissimilar items. As a result, the best k users have more chances to be matched among the inserted k fake users. It is less likely to select unrelated users with denser data sets in comparison with sparse data sets.

Table 3 displays results of this attack. Notice that the results confirm our assumption that denser data sets outperform sparse data sets for all schemes in precision. On one hand, our assumption for denser and sparse data sets holds; on the other hand, it is clear in Table 3 that privacy measures do not have a direct negative effect on the results. Remember that PSCP and absolute values are applied for HPD schemes. It is, prima facie, an interesting result. Although PSCP could affect similarity between a and genuine users, the similarities between k fake users and a are not affected by PSCP because fake users have the exact same vector with the original active query. Items inserted into or removed from the active query does not alter commonly rated items between k fake users and a. Similarities between a and k fake users are still 1. PSCP does not have an

Table 3. *knn*-based attack

	NP				*WP*			
	Best $-$ N_n		*Threshold-based*		*Best* $-$ N_n		*Threshold-based*	
	MLM	*Jester*	*MLM*	*Jester*	*MLM*	*Jester*	*MLM*	*Jester*
Precision	0.212	0.749	0.278	0.717	0.251	0.731	0.277	0.750
Recall	0.081	0.515	0.134	0.504	0.114	0.506	0.139	0.526

important effect in terms of privacy for this attack, because it does not seriously affect the similarity results. Thus, the results follow similar trends in NP with WP. Consequently, we believe precision and recall results are arbitrarily better for some cases and worse for some others. As a result, PSCP does not prevent privacy breaches for *knn*-based attacks.

6 Conclusions and Future Work

Partitioned data-based collaborative filtering schemes are important because e-commerce companies might prefer to collaborate to offer richer prediction to their customers. To ensure privacy, privacy-preserving collaborative filtering schemes are offered and these schemes have some substantial measure to prevent privacy breaches that might occur during mutual interaction. In this study, we focus on horizontally partitioned data-based privacy-preserving collaborative filtering schemes on binary ratings. We experimentally test their resilience to the known attack types in the literature and propose an attack type, *perfect match*. Acting as an active user attack can be considered serious for dense data set even if privacy measures are introduced. Our proposed attack, *perfect match*, can be considered successful when privacy measures are not applied; however, private similarity computation protocol is an important factor to mitigate its damage in terms of the first aspect of privacy. On the other hand, *perfect match* attack is a serious threat for the second aspect of privacy. *knn*-based attack is not seriously affected by private similarity computation protocol because the protocol does not affect the similarity computation. The similarity metric is calculated by only considering commonly rated items and our initial instinct is that all rated entries of the active user and the related user whose similarity is calculated should be taken into account while considering the similarities. Another inference is that dense data sets are more prone to the attacks.

As a future plan, we plan to offer some measures to enhance privacy levels against the known attack types and try to enlarge our vision in terms of possible privacy risks that may occur beyond the attacks discussed in this study. Furthermore, we wish to apply multi-party version of two-party binary privacy-preserving collaborative filtering schemes attacked in this study; analyze and test it to discover how much privacy they offer against the privacy risks.

Acknowledgements. This work is supported by the Grant 113E262 from TUBITAK.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 247–255, Santa Barbara, CA, USA (2001)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 19th ACM SIGMOD International Conference on Management of Data, pp. 439–450, Dallas, TX, USA. (2000)
3. Bilge, A., Kaleli, C., Yakut, I., Gunes, I., Polat, H.: A survey of privacy-preserving collaborative filtering schemes. Int. J. Softw. Eng. Knowl. Eng. **23**(08), 1085–1108 (2013)
4. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: You might also like: privacy risks of collaborative filtering. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 231–246, Oakland, CA, USA (2011)
5. Canny, J.: Collaborative filtering with privacy. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 45–57, Oakland, CA, USA (2002)
6. Cranor, L.F.: I didn't buy it for myself privacy and ecommerce personalization. In: Proceedings of the ACM Workshop on Privacy in the Electronic Society, pp. 111–117, Washington, DC, USA (2003)
7. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992). http://doi.acm.org/10.1145/138859.138867
8. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: a constant time collaborative filtering algorithm. Inf. Retr. **4**(2), 133–151 (2001). http://dx.org/10.1023/A:1011419012209
9. Guo, S., Wu, X., Li, Y.: Deriving private information from perturbed data using iqr based approach. In: Proceedings 22nd International Conference on Data Engineering Workshops, pp. 92–92 (2006)
10. Guo, S., Wu, X., Li, Y.: Determining error bounds for spectral filtering based reconstruction methods in privacy preserving data mining. Knowl. Inf. Syst. **17**(2), 217–240 (2008)
11. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.T.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237, Berkeley, CA, USA (1999)
12. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, CSCW 2000, pp. 241–250. ACM, USA (2000). http://doi.acm.org/10.1145/358916.358995
13. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: Proceedings of the 24th ACM SIGMOD International Conference on Management of Data, pp. 37–48, Baltimore, MD, USA (2005)
14. Kaleli, C., Polat, H.: Providing Naïve bayesian classifier-based private recommendations on partitioned data. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 515–522. Springer, Heidelberg (2007)
15. Kaleli, C., Polat, H.: Privacy–preserving naïve bayesian classifier based recommendations on distributed data. Comput. Intell. **31**(1), 47–68 (2015). http://dx.org/10.1111/coin.12012

16. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: Proceedings of the 3rd IEEE International Conference on Data Mining, pp. 99–106, Melbourne, FL, USA (2003)
17. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: Random-data perturbation techniques and privacy-preserving data mining. Knowl. Inf. Syst. **7**(4), 387–414 (2005)
18. Miyahara, K., Pazzani, M.J.: Collaborative filtering with the simple bayesian classifier. In: Mizoguchi, R., Slaney, J. (eds.) PRICAI 2000. LNCS, vol. 1886, pp. 679–689. Springer, Heidelberg (2000)
19. Okkalioglu, M., Koc, M., Polat, H.: Deriving binary ratings from masked data. Submitted for review
20. Okkalioglu, M., Koc, M., Polat, H.: On the discovery of fake binary ratings. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC 2015, pp. 901–907. ACM, USA (2015). http://doi.acm.org/10.1145/2695664.2695866
21. Polat, H., Du, W.: Privacy-preserving collaborative filtering using randomized perturbation techniques. In: Proceedings of the 3rd IEEE International Conference on Data Mining, pp. 625–628, Melbourne, FL, USA (2003)
22. Polat, H., Du, W.: Privacy-preserving top-n recommendation on horizontally partitioned data. In: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2005, pp. 725–731. IEEE Computer Society, Washington, DC (2005). http://dx.org/10.1109/WI.2005.117
23. Polat, H., Du, W.: Achieving private recommendations using randomized response techniques. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 637–646. Springer, Heidelberg (2006)
24. Polat, H., Du, W.: Privacy-preserving top-n recommendation on distributed data. J. Am. Soc. Inf. Sci. Technol. **59**(7), 1093–1108 (2008). http://dx.org/10.1002/asi.20831
25. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–58 (1997). http://doi.acm.org/10.1145/245108.245121
26. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artif. Intell, p. 4 (2009). http://dx.org/10.1155/2009/421425
27. Warner, S.L.: Randomized response: a survey technique for eliminating evasive answer bias. J. Am. Stat. Assoc. **60**(309), 63–69 (1965)
28. Yakut, I., Polat, H.: Estimating NBC-based recommendations on arbitrarily partitioned data with privacy. Knowl. Based Syst. **36**(0), 353–362 (2012). http://www.sciencedirect.com/science/article/pii/S0950705112002031
29. Zhang, S., Ford, J., Makedon, F.: Deriving private information from randomly perturbed ratings. In: Proceedings of the 6th SIAM International Conference on Data Mining, pp. 59–69, Bethesda, MD, USA (2006)

Position Papers

Privacy Threats in E-Shopping (Position Paper)

Jesus Diaz[1](✉), Seung Geol Choi[2], David Arroyo[1], Angelos D. Keromytis[3], Francisco B. Rodriguez[1], and Moti Yung[3,4]

[1] Universidad Autónoma de Madrid, Madrid, Spain
{j.diaz,david.arroyo,f.rodriguez}@uam.es
[2] United States Naval Academy, Annapolis, USA
choi@usna.edu
[3] Columbia University, New York, USA
[4] Google Inc., New York, USA
{angelos,moti}@cs.columbia.edu

Abstract. E-shopping has grown considerably in the last years, providing customers with convenience, merchants with increased sales, and financial entities with an additional source of income. However, it may also be the source of serious threats to privacy. In this paper, we review the e-shopping process, discussing attacks or threats that have been analyzed in the literature for each of its stages. By showing that there exist threats to privacy in each of them, we argue our following position: "It is not enough to protect a single independent stage, as is usually done in privacy respectful proposals in this context. Rather, a complete solution is necessary spanning the overall process, dealing also with the required interconnections between stages." Our overview also reflects the diverse types of information that e-shopping manages, and the benefits (e.g., such as loyalty programs and fraud prevention) that system providers extract from them. This also endorses the need for solutions that, while privacy preserving, do not limit or remove these benefits, if we want prevent all the participating entities from rejecting it.

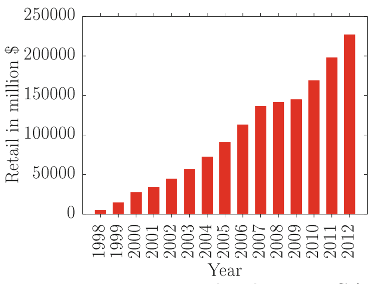
Keywords: Privacy · Online shopping · Payment systems · Purchase systems

1 Introduction

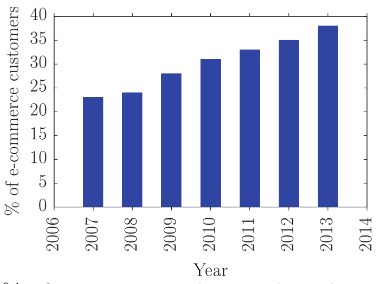
E-shopping[1] has been growing continuously (Fig. 1), providing customers with convenience, merchants with increased sales, and financial entities with an additional source of income. Concurrently, e-shopping has become more complex, the complete process being divided in several stages dealing with specific subprocesses (i.e., purchase, payment, delivery and completion). As the e-shopping process has grown more complex, protecting privacy of consumers has become

[1] In this work, we restrict ourselves to the context of B2C (business-to-consumer). B2B (business-to-business) may require additional considerations.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 217–225, 2016.
DOI: 10.1007/978-3-319-29883-2_14



E-commerce retail sales in USA between 1998 and 2012 [19].



% of EU-28 users having bought via e-commerce in last 3 months [10].

Fig. 1. Indicators of e-commerce growth in USA and EU-28.

more difficult. There are multiple parties involved, each managing various pieces of information; if any single party mistreats or misuses consumer data, consumer privacy will be at risk. Moreover, the complexity of the information each party manages makes the mistreatment or misuse only more likely.

In this paper, we review the e-shopping process, discussing attacks or threats analyzed in the literature for each of its stages. By showing privacy threats in each phase, we argue that it is not enough to protect a single independent stage, as is usually done in privacy respectful proposals in this context. Rather, a complete solution is needed spanning the overall process, dealing also with the required interconnections between stages. Our overview also reflects the diverse types of information that e-shopping manages, and the benefits (like loyalty programs and fraud prevention) that system providers extract from them. This endorses the need for solutions that, while privacy preserving, do not hinder these benefits, if we want prevent all the participating entities from rejecting it.

2 The Process of E-Shopping Transactions

The participants in e-shopping are basically the same as in the conventional shopping setting. Customers (`C` hereafter) acquiring goods, merchants (`M`) offering their products, and banks, credit card companies, etc., responsible for managing the financial backend (hereafter referred to as financial network, or `FN`). Finally, when selling physical goods, a delivery company (`DC`) is also necessary.

The entire process of an e-shopping transaction may be divided in three phases [24], plus an optional final phase (see Fig. 2):

1. Purchase. Customer `C` selects the products from merchant's `M` website.
2. Checkout. Having specified the shipping and payment information, `C` confirms the purchase and pays (through `FN`) for the selected products.
3. Delivery. `M` (probably, through `DC`) delivers the products to `C`.
4. [Optional] Completion (evaluation and dispute solving). `C` evaluates her experience, maybe including product returns or refunds.

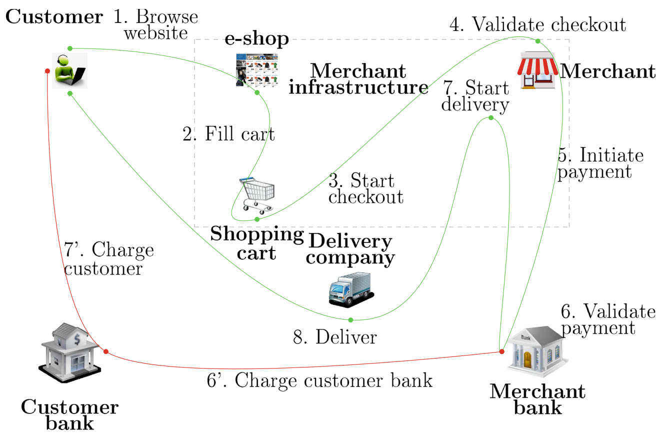


Fig. 2. Overview of the online shopping process.

3 Threats to Privacy in E-Shopping

We review existing threats to privacy through the lens of e-shopping as described in Sect. 2. Table 1 summarizes the threats that have been exposed to some extent in the literature for each phase (in the references under the third column).

We note that some of them occur quite naturally in current industry systems (like threat 2.2, since `M` usually learns `C`'s payment information, or threat 3.2, since `DC` learns both `C` and `M` addresses, being able to link `C` and `M`). However, as we will see in some of the reviewed attacks, sometimes it is enough with few additional information in order for an attacker to seriously undermine privacy. Therefore, it is advisable to keep the principle of *least information*. We also note that we mostly focus on threats which allow third (or unauthorized) parties to gain information they should not learn in a privacy-ideal scenario; many times, this leads to threats to anonymity, or leakage of product or payment information.

Table 1. Summary of privacy threats and related works.

Phase	Privacy threats	References
Purchase	1.1. Product info leaked to `FN` or 3rd parties	[5,22]
	1.2. Link `C` and `M` by 3rd parties	
Checkout	2.1. Product info leaked to `FN` or 3rd parties	[1,2,5,8,15,16,22,25]
	2.2. Payment info leaked to `M` or 3rd parties	
	2.3. Link `C` and `M` by 3rd parties	
Delivery	3.1. Shipping address leaked to `M` or 3rd parties	[4,5]
	3.2. Link `C` and `M` by `DC` or 3rd parties	
Completion	4.1. Private info leaks through feedback (All previous threats may affect completion)	[13,18,20,23]

Threats in the Purchase Stage. We first note that 9 out of 13 risks outlined in [5] (risks 4 to 13, mostly dealing with C's personal information and M's dishonesty) affect the purchase process and the communications between C and M.

Additional vulnerabilities may lie in loyalty programs; M can apply loyalty programs to lure more customers into buying products. The promotions offered by M will probably be based on C's profile, purchase history, and shopping cart. Since M will likely apply loyalty programs to increase their revenue, it is necessary to analyze how customers' personal information is treated. Amazon declares they automatically collect information such as purchase history, IP address, e-mail address and miscellaneous browser information, in order to improve users' experience.[2] eBay also claims they collect similar information.[3] Indeed, the practice of gathering users' data is common (e.g., the *Magento* includes several extensions for dealing with customer relationship management[4]).

In [22], threat 1.1 in Table 1 is exposed for e-shops using PayPal. In this study, it was observed that 52 % of the analyzed e-shops where sending product names, number of items and descriptions to PayPal. In addition, [22] also showed that PayPal leaked tracking information to Adobe's Omniture, including the referrer URL, which directly allows to link C and M (realizing threat 1.2 in Table 1). Moreover, note that in conventional e-shopping, risk 1.2 is always present, since FN and DC usually learn both C and M identities [5].

Threats in the Checkout Stage. In this stage, C specifies the payment information and shipping address. After applying fraud prevention techniques (e.g., reject purchases of more than a predefined price), M checks the promotions presented by C, if any, and forwards the payment information to FN. After validating the payment information (along with additional fraud prevention mechanisms), FN executes the payment. When checkout is completed, M updates C's profile.

This stage handles most pieces of information; risks 1 to 6 and risk 13 of [5] (dealing with misuse, by M or an attacker, of the payment information) directly affect this stage. Namely, either M or FN may misuse C's personal or payment information. Even if it is not misused by a dishonest entity, a *honest but curious* party may still pose a serious threat.

Concerning threat 2.1 in Table 1, as pointed out in [16], the current widely deployed 3-D Secure protocol, e.g., "Verified by Visa", "MasterCard SecuriCode", or "American Express SafeKey", requires a description of the transaction to be sent to FN (more exactly, the card issuer) in order for the cardholder to see and check it later. In particular, we know that some merchants leak product information to FN [22]. As to threat 2.2, in the protocol "Verified by Visa", M receives C's PAN (Primary Account Number, i.e., the credit card number) [28].

[2] See "Amazon.com Privacy Policy" at https://www.amazon.com/gp/help/customer/display.html?nodeId=468496. Last access on January 13th, 2015.

[3] See eBay's "User Privacy Notice" at http://pages.ebay.com/help/policies/privacy-policy.html. Last access on January 13th, 2015.

[4] http://www.magentocommerce.com. Last access on June 27th 2015.

A relevant example of threat 2.3 in Table 1, which may also imply threats 2.1 and 2.2, appears in [8]. From a large set of simply anonymized financial data (without names, addresses or obvious identifiers), [8] shows that it is possible to de-anonymize 90 % of the individuals, if the data contain three items: price, when, and where. Note that it is very likely that the financial data collected by FN contain the three pieces of information. Price and time are directly known to FN. As for the location, for online purchases it may be deduced from IP addresses, shipping addresses, M's information, etc. Worse yet, mobile-based payment would directly provide spatial coordinates through cellsite location [1,25].

The concepts of *receiver privacy* and *value privacy* [15] formally capture the importance of the empirical analysis in [8]. As stated in [15], receiver privacy is maintained if "*the adversary cannot determine the receiver of a transaction, as long as this is issued by a non-compromised sender*", and value privacy is maintained if "*the adversary cannot determine the value of a transaction between two non-compromised users*". Ignoring value privacy implies leaking the price of a transaction, which significantly eases de-anonymization [8]. Ignoring receiver privacy leads to linking C and M, which may represent a privacy violation by itself (i.e., when buying sensitive products or services), or may lead to a privacy violation enabling re-identification of C from aggregated metadata.

Moreover, the payment information processed by financial entities includes card and account numbers, identifiers that persist across online and offline platforms and systems (unlike, e.g., cookies). This further implies that financial entities possess very sensitive information that paves the way to link purchases with payment transactions and perform behavioral analysis over customers' data [22]. Finally, fraud prevention is very relevant in the payment phase. It is the main mechanism that merchants and financial entities employ to prevent losses, which are far from negligible [1,2]. However, as pointed out in [1] new trends in these fraud prevention may pose a serious threat to privacy, like incorporating geolocation from mobile phones or information from social networks.

Threats in the Delivery Stage. Once M receives the payment, it delivers the purchased goods to C. For digital goods, the files are sent via Internet, and using anonymizing networks [9] is a robust way to protect privacy. For physical goods, these will be shipped through some delivery company DC to the shipping address specified by C to M during checkout (thus, realizing threat 3.1 in Table 1). Also, as pointed out in [5], depending on the information available to DC, it may pose additional privacy threats. In the real world, the delivery company DC at least learns both C's and M's addresses (threat 3.2 in Table 1), which allows it to link them, and may also learn other data, such as product related information.

However, preventing M (or other entities) from learning C's physical address and DC to learn both C's and M's addresses is costly. Probably, physical mix networks are the most privacy respectful option [4]. Alternatively, Post Office boxes or equivalent delivery methods offer an intermediate solution between complexity and privacy, as it reveals a nearby location instead of C's address.

Threats in the Completion Stage. After receiving the purchased items, C verifies that everything is correct, checking the debited amount, the received items, etc. If C is satisfied, the purchase is completed. If some error is detected, C may initiate a complaint. The situation is more complicated for purchases through e-shopping platforms (e.g., Amazon) rather than directly with the merchant; in this case, although it is usually recommended to first contact the merchant[5], it may be necessary for the e-shopping platform to mediate. In these situations, the privacy risks described for the previous stages will also be present, since C may need to provide product or payment information, or her contact information.

Additionally, whichever the final result is, C may provide online feedback about M, for other customers to decide whether or not to buy from him; in some platforms, such as eBay, M may also evaluate C. Concerning the possibility of leaving feedback, [13] shows how insufficient privacy controls may lead to serious privacy threats. Indeed, it is possible to infer the purchase history of a specific user by correlating the feedback she has received with the feedback received by the sellers with whom she has interacted. Also, it is possible to perform a *category attack* to obtain a list of the people that has bought an item of a specific type (e.g., guns). Other attacks explained in [13] include a *broad profiling attack* and a *side-information attack*, which also pose a serious threat to buyers (even enabling third parties to compromise their privacy in the case of the side-information attack). In a related context, [18] explains how to identify users in the Netflix database from little external information. All these attacks are realizations of threat 4.1 in Table 1. A more general model of the privacy threats related to recommendation systems is described in [20,23].

4 Proposals for Privacy-Preserving E-Shopping

There exists a reasonable amount of work addressing privacy flaws in different e-shopping subprocesses. Probably, most of the effort has been dedicated to payment systems. See, e.g., [3,12,14] for credit-card based systems protecting customers' privacy in different manners, or Bitcoin [17] and other cryptocurrencies for e-cash based systems. Private purchase systems are presented in [24], and iPrivacy [26] uses a proxy for sanitizing sensitive data from purchase orders. Fraud prevention has also been analyzed in the case of micropayments [6] and for the Bitcoin system as a means to prevent double-spending [11]. Also, privacy preserving marketing systems have received some attention [7,21]. Finally, [4] describes a physical mix network for physical objects delivery, and [26] proposes to use intermediate depots for protecting the customers' address.

However, as this summary of related privacy preserving systems shows, current proposals focus on specific stages of e-shopping. Nevertheless, if only one stage is protects privacy, attackers will just move to another one for breaking it. Moreover, solutions providing a subset of the functionality in current industry

[5] See https://payments.amazon.com/help/5968. Last access on June 29th, 2015.

systems will probably be rejected by the main entities in the ecosystem, since all the mentioned features provide important usability and economic benefits.

5 Discussion

The process of an e-shopping transaction is complex, involving many entities, stages (Sect. 2) and features. In Sect. 3, we have outlined the privacy issues present at each stage. As a whole, the risks derived from those threats increase with the integration of all the mentioned processes in a complete infrastructure, as the *attack surface* increases as result of the greater complexity, amount and variety of the information required to complete an e-shopping operation. However, only solutions for specific parts of the system have been proposed, putting aside the remaining components and leaving the overall design vulnerable.

The realization of any of the previous risks would imply serious threats to the privacy of online shopping customers. Not protecting against these threats may allow third parties to obtain customers' sensitive information. But privacy is not just a theoretical concern. As it has been empirically observed, when correctly informed about the subject, customers prefer privacy preserving online shops [27]. Moreover, customers are even willing to pay additional fees or higher prices for privacy preserving systems. For companies, this preference for privacy preserving alternatives is yet another incentive to address these issues.

To prevent rejection from the industry due to important features being ignored, we argue that first, global infrastructures promoting privacy and spanning the overall process should be devised. Second, proposals for each subsystem should be integrated within it, preventing information leaks. Finally, once every subsystem is implemented within the global architecture, a practical, comprehensive and privacy preserving e-shopping solution would have been achieved.

Thus, a central question is whether such a privacy supporting and feature-comprehensive solution is possible, being a central challenge in this scenario to find a good balance to satisfy both customers and service providers. An approach to achieve this is to employ the cryptographic primitives put forward in state-of-the-art Privacy Enhancing Technologies. A successful combination of these techniques would enable a robust solution, compatible with the complexity of the trust sharing features required by the e-shopping infrastructure.

Acknowledgements. This work was supported by project S2013/ICE-3095-CM (CIBERDINE) of the Comunidad de Madrid and MINECO TIN2010-19607, TIN2012-30883, TIN2014-54580-R. The work of Seung Geol Choi was supported in part by the Office of Naval Research under Grant Number N0001415WX01232. The work of Moti Yung was done in part while visiting the Simons Institute for Theory of Computing, UC Berkeley. The work of Jesus Diaz was done in part while visiting the Network Security Lab at Columbia University.

References

1. Anderson, R.J.: Risk and privacy implications of consumer payment innovation (2012). http://www.cl.cam.ac.uk/rja14/Papers/anderson-frb-kansas-mar27.pdf
2. Anderson, R.J., Barton, C., Böhme, R., Clayton, R., van Eeten, M., Levi, M., Moore, T., Savage, S.: Measuring the cost of cybercrime. In: WEIS 2012, Germany, pp. 25–26, June 2012
3. Androulaki, E., Bellovin, S.: An anonymous credit card system. In: Fischer-Hübner, S., Lambrinoudakis, C., Pernul, G. (eds.) TrustBus 2009. LNCS, vol. 5695, pp. 42–51. Springer, Heidelberg (2009)
4. Androulaki, E., Bellovin, S.: APOD: anonymous physical object delivery. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 202–215. Springer, Heidelberg (2009)
5. Antoniou, G., Batten, L.M.: E-commerce: protecting purchaser privacy to enforce trust. Electron. Commer. Res. **11**(4), 421–456 (2011)
6. Blaze, M., Ioannidis, J., Keromytis, A.D.: Offline micropayments without trusted hardware. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, p. 21. Springer, Heidelberg (2002)
7. Chen, L., Escalante B., A.N., Löhr, H., Manulis, M., Sadeghi, A.-R.: A privacy-protecting multi-coupon scheme with stronger protection against splitting. In: Dietrich, S., Dhamija, R. (eds.) FC/USEC 2007. LNCS, vol. 4886, pp. 29–44. Springer, Heidelberg (2007)
8. de Montjoye, Y.-A., Radaelli, L., Singh, V.K., Pentland, A.: Unique in the shopping mall: on the reidentifiability of credit card metadata. Science **347**(6221), 536–539 (2015)
9. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: the second-generation onion router. In: USENIX Security Symposium (2004)
10. The Eurostat. E-commerce by individuals and enterprises (December 2014). http://epp.eurostat.ec.europa.eu
11. Karame, G.O., Androulaki, E., Roeschlin, M., Gervais, A., Capkun, S.: Misbehavior in bitcoin: a study of double-spending and accountability. ACM Trans. Inf. Syst. Secur. **18**(1), 2 (2015)
12. Low, S.H., Maxemchuk, N.F., Paul, S.: Anonymous credit cards and their collusion analysis. IEEE/ACM Trans. Netw. **4**(6), 809–816 (1996)
13. Minkus, T., Ross, K.W.: I know what you're buying: privacy breaches on eBay. In: De Cristofaro, E., Murdoch, S.J. (eds.) PETS 2014. LNCS, vol. 8555, pp. 164–183. Springer, Heidelberg (2014)
14. Molloy, I., Li, J., Li, N.: Dynamic virtual credit card numbers. In: Dietrich, S., Dhamija, R. (eds.) FC/USEC 2007. LNCS, vol. 4886, pp. 208–223. Springer, Heidelberg (2007)
15. Moreno-Sanchez, P., Kate, A., Maffei, M., Pecina, K.: Privacy preserving payments in credit networks: enabling trust with privacy in online marketplaces. In: NDSS 2015, San Diego (2015)
16. Murdoch, S.J., Anderson, R.: Verified by visa and mastercard securecode: or, how not to design authentication. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 336–342. Springer, Heidelberg (2010)
17. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009)
18. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy (S&P 2008), 18–21 May 2008. Oakland (2008)

19. U.S. Department of Commerce. The 2nd quarter retail e-commerce sales report (2013)
20. Parra-Arnau, J., Rebollo-Monedero, D., Forné, J.: Optimal forgery and suppression of ratings for privacy enhancement in recommendation systems. Entropy **16**(3), 1586–1631 (2014)
21. Partridge, K., Pathak, M.A., Uzun, E., Wang, C.: PiCoDa: privacy-preserving smart coupon delivery architecture (2012)
22. Preibusch, S., Peetz, T., Acar, G., Berendt, B.: Purchase details leaked to PayPal (short paper). In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 217–226. Springer, Heidelberg (2015)
23. Ramakrishnan, N., Keller, B.J., Mirza, B.J., Grama, A., Karypis, G.: Privacy risks in recommender systems. IEEE Internet Comput. **5**(6), 54–62 (2001)
24. Rial, A.: Privacy-preserving e-commerce protocols. Ph.D. thesis, Arenberg Doctoral School, KU Leuven (2013)
25. Sadeh, N.M.: M-Commerce: Technologies, Services, and Business Models. John Wiley & Sons Inc., New York (2002)
26. Stolfo, S., Yemini, Y., Shaykin, L.: Electronic purchase of goods over a communications network including physical delivery while securing private and personal information of the purchasing party. US Patent App. 11/476,304, 2 November 2006
27. Tsai, J.Y., Egelman, S., Cranor, L.F., Acquisti, A.: The effect of online privacy information on purchasing behavior: an experimental study. Inf. Syst. Res. **22**(2), 254–268 (2011)
28. Visa. Verified by Visa - acquirer and merchant implementation guide (2011)

Comparison-Based Privacy: Nudging Privacy in Social Media (Position Paper)

Jan Henrik Ziegeldorf(✉), Martin Henze, René Hummen, and Klaus Wehrle

Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany
{ziegeldorf,henze,hummen,wehrle}@comsys.rwth-aachen.de

Abstract. Social media continues to lead imprudent users into *over-sharing*, exposing them to various privacy threats. Recent research thus focusses on *nudging* the user into the 'right' direction. In this paper, we propose *Comparison-based Privacy (CbP)*, a design paradigm for privacy nudges that overcomes the limitations and challenges of existing approaches. *CbP* is based on the observation that comparison is a natural human behavior. With *CbP*, we transfer this observation to decision-making processes in the digital world by enabling the user to compare herself along privacy-relevant metrics to user-selected comparison groups. In doing so, our approach provides a framework for the integration of existing nudges under a self-adaptive, user-centric norm of privacy. Thus, we expect *CbP* not only to provide technical improvements, but to also increase user acceptance of privacy nudges. We also show how *CbP* can be implemented and present preliminary results.

Keywords: Behavioral nudge · Privacy · Social media

1 Introduction

Over-sharing of personal information on social media has led to several privacy incidents: (i) People have missed career opportunities [6], (ii) embarrassed themselves [17], or (iii) become victims of crimes [5]. In response, websites have implemented access and privacy controls. However, these protection mechanisms usually come with very lenient defaults and users often fail or simply neglect to set up individual settings [12,17]. Recent research explores the use of behavioral nudges to raise awareness about privacy risks and lead users to informed decisions about their social media privacy. These *privacy nudges* try to detect privacy sensitive contexts and warn users, e.g., a-priori to critical posts on Facebook [18].

The proposed nudges face two challenges inherent to their design. First, they require ground truth to detect sensitive content which is not available per se. Consequently, proposed systems focus only on very specific privacy threats, e.g., a nudge that warns about the disclosure of vacation plans based on laboriously hand-labelled data [13]. Second, the proposed systems convey only the subjective privacy norm of the person designing and training the system, i.e., privacy is

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 226–234, 2016.
DOI: 10.1007/978-3-319-29883-2_15

defined in a *one-for-all* manner. Considering the importance of individual and social aspects in privacy, it is not surprising that many users disagree and reject advise from nudges that dictate a one-for-all definition of privacy [10].

In this position paper, we propose *Comparison-based Privacy (CbP)*, a new best-effort approach for nudging privacy. *CbP* is motivated by the observation that comparisons are widely used by humans in their every-day lives to assess their own status, behavior, and decisions, and that such comparisons are also effective in influencing a person's behavior [7]. Therefore, we propose to support a user's decision making in privacy contexts by comparing her sharing behavior along different metrics (e.g., amount of shared content or usage patterns) to different comparison groups, which she can intuitively relate to (e.g., family, friends and colleagues, users with the same profession or same age). Because of its inherently relative nature, *CbP* neither assumes nor requires any fixed privacy norm or ground truth. Instead, a user is nudged completely based on the behavior of her peer groups. This also allows *CbP* to harmonize individual and social factors of privacy. Individual aspects are covered by the user's choice of comparison metrics, while social aspects are captured in the aggregated behavior of a specific comparison group. With this, *CbP* overcomes the restrictions of other privacy nudges and promises increased user acceptance, easier deployment and maintenance, and a certain degree of adaptivity to changing notions of privacy.

2 Problem Analysis and Related Work

The problem of over-sharing fundamentally stems from users' inability to responsibly decide how often to share which content with whom. Recently proposed privacy nudges tackle this problem by raising awareness about specific consequences of over-sharing. `PleaseRobMe`[1] addresses geo-location information and `FireMe!`[2] abusive language related to work. These systems are based on manually configured filter rules that allow to detect only very specific privacy risks. As improvement, [13] employs supervised machine learning to detect sensitive tweets and [9] automatically annotates text-based social media content with privacy labels. However, due to the apparent lack of ground truth to train these systems, only a small set of less than 1 000 hand-labeled tweets [13] or synthetic data [9] is used. [11] proposes a privacy score based on the sensitivity of profile items, which was exemplarily determined through a user study. These approaches show that providing ground truth on the sensitivity of social media content currently requires substantial manual effort. *CbP* avoids these efforts by basing nudging decisions solely on comparisons between a user and her peer groups.

A second challenge common to related work [10,13,18] is that the norm of privacy is dictated during system development and is immutable from there on. However, privacy is both an individual and social concept that cannot be defined in a one-for-all manner. First, individual factors such as a user's demographics,

[1] http://pleaserobme.com/.
[2] http://fireme.l3s.uni-hannover.de/.

profession, or personal preferences play an important role. Second, privacy decisions are also shaped by the perception and appreciation of privacy in the user's social environment. Negligence of these factors leads to non-acceptance among users: While [13] does not investigate user acceptance, users tend to reject the nudged advise of [10,18] as they do not feel addressed individually. In contrast, *CbP* proposes nudging users in a self-adaptive, user-centric way.

Finally, a long line of research investigates on how to learn and configure users' access and sharing policies: [1] (semi-) automatically learns a user's group memberships and [4] automatically assigns privileges to a user's friends based on a limited amount of user input and settings of other users. Other approaches focus on predicting location sharing preferences [15,16]. [14] proposes to let users collaboratively manage access control to social media data. Our work is orthogonal as it engages the user one step earlier: We aim at nudging users towards treating their digital privacy more consciously, which could, e.g., lead the user to customize privacy preferences using one of the above approaches. However, our proposed *CbP* paradigms draws and extends on the idea of collaboratively managing privacy that is present in some of the discussed approaches.

3 Comparison-Based Privacy

To enable self-adaptive, user-centric privacy nudges, we make the following three observations. First, comparison is a natural human behavior. People compare themselves to their peer groups everyday based on a wide set of criteria ranging from salary to health. Second, comparison does not require ground truth or training data. Instead, self-reflection and decision making is rather guided by relative values. The aggregated behavior of the peer group dynamically provides individual 'ground truth' for people to evaluate their own decisions. Third, people usually compare not to random strangers. They compare to people from their social environment who they can individually relate to, e.g., people with the same profession, age, or other demographics. In doing so, they harmonize individual and social factors that influence their decision-making process.

Based on these observations, we argue that comparing privacy relevant aspects of a user's social media activity allows her to intuitively understand and assess her privacy risk. Specifically, we propose to compare a user's sharing behavior along a number of *comparison metrics* to user-specific *comparison groups*. We refer to this novel approach as *Comparison-based Privacy (CbP)*. Notably, our approach renounces any fixed norm of privacy and fully embraces privacy as both an individual and a social concept. We now discuss our comparison metrics and groups and their combination, while deferring technical details to Sect. 4.

Comparison Metrics: Comparison metrics capture privacy-critical aspects of a user's sharing behavior. They are motivated from an analysis of the consequences of over-sharing on social media. Related work already proposed a wide range of such metrics: It has been recognized that employers and credit scorers look at *linguistic features* of applicants [8], e.g., correctness of grammar and spelling or abusive language. Other threats, e.g., stalking and cybercasing, exploit certain *content types* such as geo-location or pictures [5]. Embarrassment or loss of career opportunities often emerge from talk about *sensitive topics* such as drug abuse or disease [6]. Finally, hints for mental diseases such as depression can be detected in users content [2]. It is important to note that our *CbP* approach neither obsoletes these related works nor is limited by it. Instead, *CbP* provides a unifying and extensible framework to integrate existing approaches as comparison metrics or devise new ones. We can, e.g., integrate as comparison metrics Kawase's job hater filter [10], Wang's nudge based on expressed sentiment [18], or Mao's disease and drunkenness classifiers [13]. The application of the *CbP* paradigm thereby transforms their fixed norm of privacy into a relative, comparison-based notion, thus increasing the acceptance among users.

Comparison Groups: Comparison groups allow a user to adapt *CbP*-based nudges to her specific norm of privacy. Hence, a user should select groups that she has an intuitive relation to. Social media sites already provide inherent structures and information, e.g., social graphs, profiles, lists of friends/followers, that provide such comparison groups and require no configuration at all. Besides these preexisting comparison groups, we can automatically build comparison groups based on user profile information, e.g., age, profession, interests and hobbies or even religion and political orientation, to provide an even more individualized nudging experience. Since not all users share this information publicly, comparisons for these groups would potentially be restricted to users of our system.

Nudging the User: The user chooses the desired comparison metrics and comparison groups individually, e.g., *"compare the amount of abusive language to people of the same profession"*. This allows the user to individualize the used norm of privacy. Our *CbP* approach then evaluates each chosen metric on the target user and builds an aggregate (e.g., average or median) over each chosen comparison group. The aggregates serve as empirical ground truth *relative* to how the social environment behaves. Social aspects of privacy are thus factored in to the nudging decision. A particular comparison between one user and the group aggregate can result in the three different cases as depicted in Fig. 1. In the first case, the user and the group behave in similar ways, i.e., the target user's result is close to the group's aggregate. This information confirms the user in her behavior with respect to this group. If a particular comparison exceeds a threshold in either direction (Cases 2 and 3 in Fig. 1), a *CbP*-based nudge would alert the user to this fact. The nudge would, e.g., alert her that the amount of abusive language in her posts exceeds the average in her peer groups. Thresholds can be set individually by users or according to general profiles representing typical privacy attitudes of an unconcerned, critical, or very anxious user. It is a desired feature of our system that a user's behavior is evaluated only in relation to her peer groups, even if results may vary or contradict each other across different groups. Such personalized appeals have proven to be more effective than judging behavior by a fixed norm of 'good' and 'bad' [7].

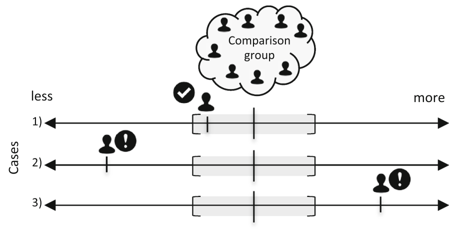


Fig. 1. Comparing a user against the aggregate group behavior.

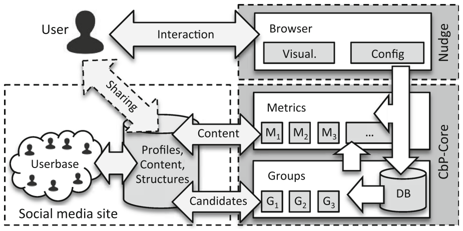


Fig. 2. Overview of the system components and their interaction.

4 Proposed System Design

We now describe a system architecture that leverages *CbP* to nudge social media users towards more privacy-conscious sharing decisions. As illustrated in Fig. 2, the system has two main components: (i) the *CbP*-core and (ii) the actual privacy nudge. The privacy nudge handles all interactions with the user. These interactions primarily include the initial discovery or configuration of the comparison groups and the comparison metrics as well as the actual nudging based on the comparison results obtained from the *CbP*-core. The *CbP*-core is a stand-alone application. It manages the comparison groups and implements comparison metrics. The user must grant it sufficient rights to query the social media site for the user's content to construct groups and evaluate the metrics. We now describe the details of the two *CbP* components and their interactions. Then, we discuss privacy implications of this design and possible alternatives.

Nudge: The nudge runs in the user's browser and represents the user interface. It asks the user to sign in and grant permission to access her social media accounts. To keep configuration efforts to a minimum, the user is presented with a preconfigured selection of comparison groups and metrics, but may refine this choice by filling in additional information. The nudge module triggers the *CbP*-core and then receives the results which it uses to nudge the user. Effective ways of actually presenting such nudging advise to a user is a question orthogonal to our approach and subject to ongoing research, e.g., [18] proposes to alter the control flow by delaying posts and [10] prods the user to delete certain content.

CbP-Core: The *CbP*-core contains the functionality to realize *CbP*, i.e., a groups module that builds and manages the comparison groups and a metrics module that implements the different comparisons (cf. Sect. 3). The *groups module* draws on standard structures of the social media site to build basic groups, e.g., the social graph or friend lists. If granted sufficient permissions, the *CbP*-core also accesses the user's protected profile information to build more specific groups. Further personal information that the user may supply during configuration, e.g., profession or age, is used to build more sophisticated groups. The *metrics module* takes a comparison group and evaluates the desired comparison metrics for each group member. Basic implementations of the metrics described in Sect. 3 can be realized using simple content filters based on word lists, e.g., for abusive language or sensitive topics, or by quantifying the amount of shared content, e.g., number of shared geo-locations. Quantifying how often similar content has been shared in the comparison groups additionally provides an indication of the sensitivity of the shared content. More comparison metrics can be built using publicly available APIs, e.g., for sentiment analysis, and through the integration of related work. Finally, the metrics module provides the aggregated results of the comparison group and the result for the particular user to the nudge module.

4.1 Discussion

Any entity, i.e., the operator of the nudge system or other users in the comparison groups, may try to spy on or actively attack the nudged user. We thus discuss how to establish trust in our system and prevent information leakage and coercion.

Trust: In our proposed design, the *CbP*-core runs as a stand-alone third-party application, as this is the easiest deployment option. However, this requires the user to trust the *CbP*-core and grant it access to her social media content. We identify two alternatives to this approach: First, the site operator itself could run the *CbP*-core or provide a suitable query interface that allows evaluation of the comparison metrics without explicitly accessing the user's contents. The second alternative is to run the *CbP*-core on the user side, e.g., as a browser plugin, and collect only the aggregate results of the comparison groups centrally. Both alternatives would not require the user to trust an additional entity.

Information Leakage: In all previously mentioned deployment scenarios, the user learns the results of the comparisons aggregated over the chosen comparison groups. However, this might be sensitive information, e.g., a malicious user may learn private information about outliers by choosing artificially small comparison groups. A trivial protection mechanism would be to only allow groups of a certain minimum size. To achieve rigorous privacy guarantees, we propose to apply Differential Privacy [3] to the aggregated outcome of the comparison.

Coercion: The aggregated behavior of a comparison group may unintentionally move into a harmful direction or a an attacker may try to manipulate it to steer a user's privacy decisions into a particular direction. We argue that a user can counter such attacks by choosing multiple, diverse, and sufficiently large comparison groups or even known reference groups, e.g., comprising the national data protectionists. As a second protection mechanism, extreme outliers, e.g., results contributed by an attacker who wants to manipulate the aggregate group behaviour, could be filtered out by the *CbP*-core component.

5 Preliminary Results

We demonstrate the feasibility of our approach by the example of Twitter. Information on Twitter is mostly public, which has lead to the many privacy violations [13], but also makes Twitter and its users a prime target for our proposed privacy nudge. Although other OSNs may enforce stricter access control on shared data and attract different categories of users, qualitatively similar privacy violations have been reported for them, e.g., for Facebook [6,17]. Thus, we expect that our obtained results also generalize to other OSNs. We collected half a million tweets of 1 839 active Twitter users in four comparison groups by profession: teachers (659), nurses (542), journalists (559), and U.S. senators (79). Groups were obtained through the Twitter Search API and scraped from public lists. We evaluate a choice of comparison metrics from Sect. 3 for each group.

The *location disclosure* metric measures the percentage of a user's tweets tagged with a geo-location. We find that all groups are very restrictive about location disclosure. Specifically, well above 90 % of the users disclose their location in less than 7.8 % of their tweets. Nearly all of them do not disclose their location at all. This result shows a wide consensus among Twitter users concerning location disclosure. This fact would immediately become apparent through the use of *CbP*. Unaware users (we observe outliers among the nurses and teachers) could therefore better assess their privacy risks with *CbP*. Results are less homogeneous for *abusive language*, defined as the percentage of tweets containing expressions regarded as offensive. Journalists and politicians use very little abusive language, while nurses and teachers show considerable use of it. Hence, it appears that some amount of abusive language is tolerable in particular groups. This confirms our relative norm of privacy and the need for user-specific comparison groups. *CbP* captures this fact and, e.g., would rather nudge the politician than the nurse. The *sensitive topics* metric measures the percentage of tweets containing references to work, diseases or drug abuse. We find that these comparisons are less useful as such topics are also referenced in many privacy irrelevant contexts. Using *sentiment analysis* on tweets as comparison metric, again shows the importance of nudging users individually with respect to their social environment. While nurses and teachers tweet with rather neutral sentiment, senators are clearly more upbeat. Surprisingly, journalists commonly display a negative mood, part of which relates to reports about crimes and disasters.

We additionally scraped the top 300 job haters from the `FireMe!` site and used it as a contrast group. Those users are endangered of job loss and our system should detect and warn against this privacy risk. Indeed, job haters spike for all our metrics, i.e., disclosing more locations than others, having significantly higher rates of abusive language, and tweeting with clearly more negative sentiment.

While our system could not directly point them to the risk of losing their job, it would still nudge them away from their harmful sharing behavior by pointing out their discrepancy with social norms established from the comparison groups.

6 Outlook and Conclusion

We are developing our proposed system for Twitter and Facebook to answer practical questions, e.g., how stable comparison results are. We also investigate further comparison metrics and groups as those briefly mentioned in Sect. 3. Finally, we intend to conduct a user study based on our developed system to answer non-technical questions: Our system may issue possibly contradicting advise, how do users respond to this? Usability is a major design goal; how much configuration is really necessary for inexperienced users?

To conclude, *CbP* presents a novel paradigm for nudging users in a best-effort manner towards more informed privacy decisions – an important challenge due to the increasing proliferation of social media among young and inexperienced users. Our *CbP* approach promises to overcome the restrictions of related work by employing a relative norm of privacy that considers both individual and social factors and does not require training data or preconfigured rules. The preliminary results show that our *CbP* paradigm indeed has the potential to effectively nudge social media users towards more privacy conscious sharing decisions.

Acknowledgements. This work has been funded by the Excellence Initiative of the German federal and state governments.

References

1. Amershi, S., Fogarty, J., Weld, D.: Regroup: interactive machine learning for on-demand group creation in social networks. In: CHI 2012. ACM (2012)
2. De Choudhury, M., Counts, S., Horvitz, E.: Social media as a measurement tool of depression in populations. In: Web-Sci 2013. ACM (2013)
3. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
4. Fang, L., LeFevre, K.: Privacy wizards for social networking sites. In: WWW 2010. ACM (2010)
5. Friedland, G., Sommer, R.: Cybercasing the joint: on the privacy implications of geo-tagging. In: HotSec 2010. USENIX (2010)
6. Garone, E.: Can social media get you fired? (2013). http://www.bbc.com/capital/story/20130626-can-social-media-get-you-fired
7. Goldstein, N.J., Cialdini, R.B.: A room with a viewpoint: using social norms to motivate environmental conservation in hotels. JCR **35**(3), 472–482 (2008)
8. Post, H.: 37 Percent of employers use Facebook to pre-screen applicants, new study says (2012). http://huff.to/1c5fvQg
9. Jakob, M., Moler, Z., Pěchouček, M., Vaculín, R.: Content-based privacy management on the social web. In: WI-IAT 2011. IEEE (2011)

10. Kawase, R., et al.: Who wants to get fired? In: WebSci 2013. ACM (2013)
11. Liu, K., Terzi, E.: A framework for computing the privacy scores of users in online social networks. TKDD **5**(1), 6 (2010)
12. Liu, Y., Gummadi, K.P., Krishnamurthy, B., Mislove, A.: Analyzing Facebook privacy settings: user expectations vs. reality. In: IMC 2011. ACM (2011)
13. Mao, H., Shuai, X., Kapadia, A.: Loose tweets: an analysis of privacy leaks on Twitter. In: WPES 2011. ACM (2011)
14. Squicciarini, A.C., Shehab, M., Paci, F.: Collective privacy management in social networks. In: WWW 2009. ACM (2009)
15. Toch, E.: Crowdsourcing privacy preferences in context-aware applications. Pers. Ubiquitous Comput. **18**(1), 129–141 (2014)
16. Toch, E., Cranshaw, J., Drielsma, P.H., Tsai, J.Y., et al.: Empirical models of privacy in location sharing. In: UbiComp 2010, pp. 129–138. ACM (2010)
17. Wang, Y., et al.: I regretted the minute I pressed share: a qualitative study of regrets on Facebook. In: SOUPS 2011. ACM (2011)
18. Wang, Y., et al.: Privacy nudges for social media: an exploratory Facebook study. In: WWW 2013. IW3C2 (2013)

Short Papers

Can you Really Anonymize the Donors of Genomic Data in Today's Digital World?

Mohammed Alser , Nour Almadhoun, Azita Nouri, Can Alkan, and Erman Ayday(✉)

Computer Engineering Department, Bilkent University, Bilkent, 06800 Ankara, Turkey
erman@cs.bilkent.edu.tr

Abstract. The rapid progress in genome sequencing technologies leads to availability of high amounts of genomic data. Accelerating the pace of biomedical breakthroughs and discoveries necessitates not only collecting millions of genetic samples but also granting open access to genetic databases. However, one growing concern is the ability to protect the privacy of sensitive information and its owner. In this work, we survey a wide spectrum of cross-layer privacy breaching strategies to human genomic data (using both public genomic databases and other public non-genomic data). We outline the principles and outcomes of each technique, and assess its technological complexity and maturation. We then review potential privacy-preserving countermeasure mechanisms for each threat.

Keywords: Genomics · Privacy · Bioinformatics

1 Introduction

Today, next-generation sequencing technologies (NGS) are capable of generating a tremendous amount of sequencing data. As a result, the production of genetic information for research, clinical care, and direct-to-consumer genomics at a rapid pace is no longer impossible from the technological point of view. The availability of human genetic biobanks provides an adequate basis for several important applications and studies. Genomic research typically includes collecting samples from thousands of individuals, but a large push is underway to sequence hundreds of thousands to millions of genomes aiming at discovering the functional impact of *de novo* (not inherited from either parent) genetic variations on diseases such as autism and cancer [9]. Accelerating the pace of biomedical breakthroughs and discoveries necessitates not only collecting millions of genetic samples, but also granting open access to the genetic biobanks and databases. This trend has caused the launch of more than one thousand publicly available online genetic databases, in which individuals publicly share their genomic data [5]. Several studies [11,16] show that the majority (i.e., 69–92 %) of the respondents have positive attitudes towards genomics research and donating their DNA samples. The most common intention behind it is to support the

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 237–244, 2016.
DOI: 10.1007/978-3-319-29883-2_16

personalized medicine studies. Second, to learn about their genetic predispositions to diseases and even their genetic compatibilities with potential partners. Last but not least, to identify their distant patrilineal relatives and the potential surnames of their biological fathers. However, the overwhelming majority of the respondents rank privacy of sensitive information as one of their top concerns. Thus, the biggest challenge of widely utilizing the human genomes and pushing the frontiers of the genetic research is both social and technical. In the literature, there exist reviews addressing genomic privacy (e.g., [4,12]). This paper focuses on the cross-layer attacks against genomic privacy of individuals (using both genomic and non-genomic data) and proposes potential countermeasure mechanisms in a systematic way. The rest of the paper is organized as follows. In Sect. 2, we survey a wide spectrum of known privacy threats to human genomic data. In Sect. 3, we overview the existing works and present our recommendations and guidelines for potential privacy-preserving countermeasure techniques for each threat. Finally, we conclude the paper in Sect. 4.

2 Genetic Privacy Breaching Strategies

In this section, we survey a wide spectrum of privacy threats to human genomic data, as reported by prior research. In general, we assume the existence of a passive attacker who has bounded computational power. In all below threats, the attacker only has access to publicly available genetic databases and other publicly available resources on the Internet.

2.1 Identity Tracing by Meta-Data and Side-Channel Leaks

In such an attack, as illustrated in Fig. 1, the hacker or curious party needs both human genomic data, which is already available online via a certain privacy-preserving mechanism (i.e., hiding the identity information of the owner), and additional metadata. Such an attack, once it succeeds, can cause serious implications, for instance genetic discrimination, financial loss, and blackmail. A real-life example of this threat was in 1997 when Sweeney [17] successfully identified the medical condition of William Weld, former governor of Massachusetts, using only his demographic data (i.e., date of birth, gender, and 5-digit ZIP code) appearing in the hospital records and voter registration forms that are available to everyone. In 2013, Sweeney [18] again showed that it is possible to utilize the demographic data to discover the real identities of the DNA donors even though their names are removed from the published genomic database. The approach was very similar to her previous attack, besides, in this work, she exploited the side-channel data in the downloaded genomic data files associated with anonymized PGP profiles. Even for some participants, once the downloaded file was uncompressed, the resulting file had a filename that included the actual name of participant.

2.2 Identity Tracing by Genealogical Triangulation

In most human societies, surnames are paternally inherited, resulting a correlation with specific Y-chromosome haplotypes. Thus, there are several online

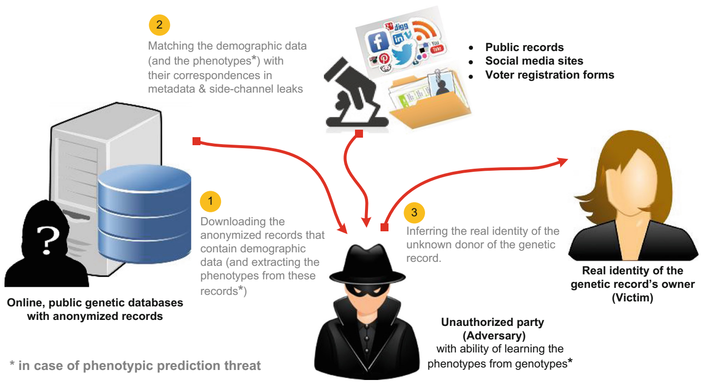

2
Matching the demographic data (and the phenotypes*) with their correspondences in metadata & side-channel leaks
Public records
Social media sites
Voter registration forms
1
Downloading the anonymized records that contain demographic data (and extracting the phenotypes from these records*)
3
Inferring the real identity of the unknown donor of the genetic record.
Real identity of the genetic record's owner (Victim)
Online, public genetic databases with anonymized records
Unauthorized party (Adversary) with ability of learning the phenotypes from genotypes*
* in case of phenotypic prediction threat

Fig. 1. A possible route for identity tracing using both metadata/side-channel leaks and phenotypic prediction.

public databases (e.g., Ysearch.org and SMGF.org) that collectively contain hundreds of thousands of surname-haplotype records, aiming at helping the public to identify their distant patrilineal relatives and the potential surnames of their biological fathers. However, these services can be exploited by an adversary towards learning the participant's identity, as illustrated in Fig. 2. With the help of surname inferences in addition to the birth year and Zip code, the search results can be narrowed down the identity to few matches that can be investigated individually [6].

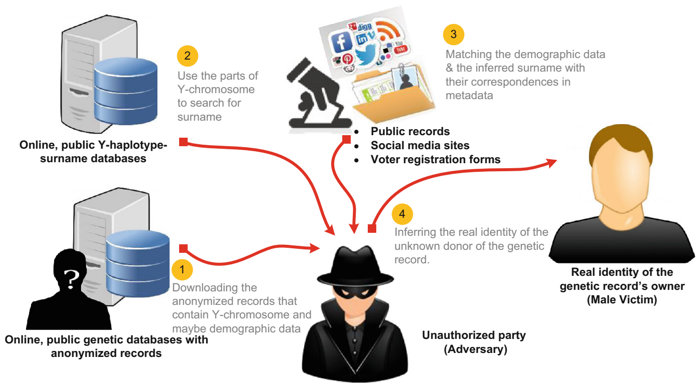

2
Use the parts of Y-chromosome to search for surname
3
Matching the demographic data & the inferred surname with their correspondences in metadata
Online, public Y-haplotype-surname databases
Public records
Social media sites
Voter registration forms
4
Inferring the real identity of the unknown donor of the genetic record.
1
Downloading the anonymized records that contain Y-chromosome and maybe demographic data
Real identity of the genetic record's owner (Male Victim)
Online, public genetic databases with anonymized records
Unauthorized party (Adversary)

Fig. 2. A possible route for identity tracing using genealogical triangulation.

2.3 Identity Tracing by Phenotypic Prediction

Visible phenotypes from genetic data could help in identity tracing. Such visible traits with high heritability that can be inferred from DNA include height, eye color, facial morphology, and age [10]. These traits can then be used as quasi-identifiers for decreasing the degree of uncertainty to infer the identity of an individual with the help of public records and social networks as explained in Fig. 1. However, using only these quasi-identifiers for re-identification does not provide high accuracy; as the population-wide registries of these visible traits are not publicly accessible and searchable.

2.4 Attribute Disclosure Attacks via DNA (ADAD)

The main concept of ADAD is when the adversary gains access to the DNA sample of the target. Using the identified DNA, the adversary can search genetic databases with sensitive attributes (e.g., drug abuse) as shown in Fig. 3. Finding the identified DNA in the database reveals the link between the person and the sensitive attribute. Based on [4], three scenarios are identified to illustrate the attribute disclosure attacks: the n=1 scenario, the summary statistic scenario, and the gene expression scenario. The n=1 scenario is the simplest scenario of ADAD. By acquiring a chosen set of 45 autosomal single nucleotide polymorphisms (SNPs)[1], the adversary can simply match the genotype data that is associated with the identity of the individual with the genotype data that is associated with the attribute [14]. Thus, Genome-Wide Association Studies (GWAS) stores individual genotypes and phenotypes in restricted access area, while the statistics of allele frequencies[2] are stored in the public access area. In spite of the separation, GWAS datasets with allele frequencies of the participants have been exploited by the ADAD's summary statistic scenario [7] as follows: The allele frequencies are positively biased towards the target genotypes in the case group compared to the allele frequencies of the general population. Moreover, the analyzed common variations can be exploited to conduct ADAD by integrating the biases in the allele frequencies over a large number of SNPs in GWAS. Therefore, the performance of ADAD is a function of the size of the study and the adversary's prior knowledge. Apart from GWAS, the NIH's Gene Expression Omnibus (GEO) databases are also vulnerable to the ADAD's gene expression scenario [15]. The GEO database holds hundreds of thousands of human gene expression profiles and their linked medical attributes. However, the NIH did not change their policies regarding sharing the gene expression data due to several complications of this threat.

2.5 Completion Attacks

In genomics, genotype imputation is a well-studied task in which genetic information can be reconstructed from partial data by completing the missing genotype values. A well-known example of a completion attack is the inference of Jim Watson's predisposition for Alzheimer's disease from his published genome, despite

[1] SNPs are the main cause for variations in the human genome. They are also responsible for the differences in our phenotypes/traits and genotypes.

[2] The allele frequency represents the incidence of a gene variant at a given gene location in a population gene pool.

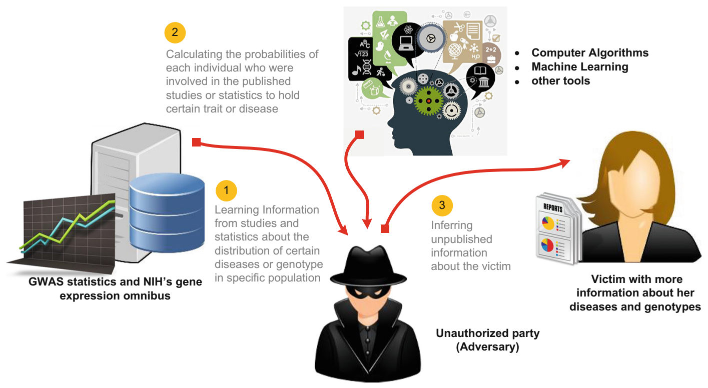

2
Calculating the probabilities of each individual who were involved in the published studies or statistics to hold certain trait or disease
Computer Algorithms
Machine Learning
other tools
1
Learning Information from studies and statistics about the distribution of certain diseases or genotype in specific population
3
Inferring unpublished information about the victim
REPORTS
GWAS statistics and NIH's gene expression omnibus
Victim with more information about her diseases and genotypes
Unauthorized party (Adversary)

Fig. 3. Attribute disclosure attacks via DNA.

removing the ApoE locus gene (which is the indicator for Alzheimer's predisposition) from the published data [13]. Completion techniques can be used to predict the genomic information when there is no access to the DNA of a known individual, as shown in Fig. 4.

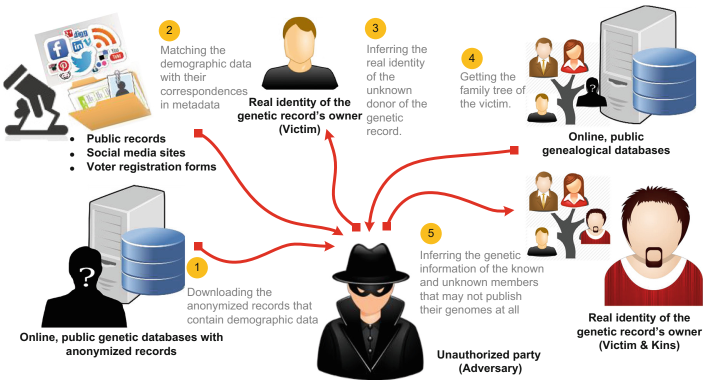

2
Matching the demographic data with their correspondences in metadata
3
Inferring the real identity of the unknown donor of the genetic record.
4
Getting the family tree of the victim.
Public records
Social media sites
Voter registration forms
Real identity of the genetic record's owner (Victim)
Online, public genealogical databases
1
Downloading the anonymized records that contain demographic data
Online, public genetic databases with anonymized records
5
Inferring the genetic information of the known and unknown members that may not publish their genomes at all
Unauthorized party (Adversary)
Real identity of the genetic record's owner (Victim & Kins)

Fig. 4. A possible route for identity de-anonymization using a completion attack.

3 Mitigation Techniques

In this section, we survey a wide spectrum of known privacy-preserving techniques against each aforementioned threat and make suggestions to prevent such threats. Here, we focus on the scenario, in which genomic data or the results of GWAS are made publicly available. There are also crypto-based mitigation techniques in which genomic data of individuals is stored in a database in encrypted form, and hence it is not publicly available on the Internet. Once other parties (e.g., medical centers) want to do operations on the data, they apply privacy-preserving techniques and they only obtain the result of the operation without having access to whole data. In this line of research, Ayday et al. proposed privacy-preserving techniques for medical tests and personalized medicine methods [2]. Baldi et al. make use of both medical and cryptographic tools for privacy-preserving paternity tests, personalized medicine, and genetic compatibility tests [3]. Also Ayday et. al developed a technique for privacy-compliant processing of raw genomic data [1]. We note that such scenarios, in which genomic data is not publicly shared, are out-of-the-scope of this paper.

3.1 Identity Tracing by Meta-Data and Side-Channel Leaks

As discussed in this threat model, metadata can be used for inferring the identities of involved individuals. Hence, any metadata that may decrease the level of privacy, should either be removed from datasets or strictly follow the 2002 Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. Data covered under HIPAA should follow certain strict formats; dates (e.g. birth, admittance, and discharge dates) would only contain the year, the ZIP code would only have the first 2 digits if the population in the ZIP code is less than 20,000 people, and no explicit identifiers (e.g. Social Security numbers) would be present.

3.2 Identity Tracing by Genealogical Triangulation

The first step towards protecting against this attack depends on the purpose of the genetic database. If the database provides services for descendants of anonymous sperm donors to identify the surnames of their potential biological father and distant patrilineal relatives, then it should be an access-controlled database. Otherwise, the surname should be removed or replaced with the given name in haplotype records in order to decrease the ability of connecting surname to unknown's genome [6]. Reconstruction attacks based on available online datasets should be performed to measure the connection of surname or other unique identifier with genomic data.

3.3 Identity Tracing by Phenotypic Prediction

To prevent this threat, data about visible traits of individuals in public genomic databases as well as other public sources should be restricted (only to qualified researchers or close connections) or removed whenever applicable in order to preserve privacy. Nonetheless, predicting a victim's phenotypes is not only based on the revealed information through genetic databases; online social networks can also be a rich source of public sensitive data, and hence privacy risk will be amplified.

3.4 Attribute Disclosure Attacks via DNA (ADAD)

To address this threat, data perturbation techniques (e.g., differential privacy) can be used for adding noise to the result of a query (on a genomic database) before releasing it publicly. In this way, the reported result will not be much different than original result, but an adversary will not understand if a given individual is in the database or not. Assuming the genomic database includes individuals with a given sensitive attribute, an adversary with prior knowledge can never be sure if that sensitive attribute belongs to a specific individual, as similar results will be given when the individual is included in the database or not. However, the added noise should be carefully considered as it will affect the accuracy and the utility of the data at the expense of privacy.

3.5 Completion Attacks

For this attack that relies on reconstructing genetic information based on partial data, one must consider all available data of each individual that is publicly shared (either by himself, his family members, or genomic researchers). If with existing completion techniques, one can predict the missing genomic information then specific parts of genomic data should be removed from datasets. Another solution is using dedicated cryptographic techniques, which enable researchers to access only some parts of the genome by requesting the decryption key from the owner. Such solutions can be merged with the reconstruction attack model from [8] to infer the amount of risk that occurs with releasing new portions of data.

4 Conclusion

The main concern when publishing anonymized genomic information is usually the privacy of its owner. As it is not trivial to predict the amount of information that will be available to the attacker in today's digital World, existing technical solutions alone are not sufficient to ensure long-term privacy for genomic data donors, and hence their family members. Therefore, there should be a collaborative effort between technical solutions, policies, and legislation (e.g. HIPAA, EU data protection law) to maintain privacy-compliant public genetic databases. As discussed, cryptographic solutions can be an option, but such solutions prevent public availability of genomic data, somehow decreasing the pace of genomic research. This trade-off should also be further investigated.

References

1. Ayday, E., Raisaro, J.L., Hengartner, U., et al.: Privacy-preserving processing of raw genomic data. In: Proceedings of 8th Data Privacy Management (DPM 2013) International Workshop (in conjunction with ESORICS) (2013)
2. Ayday, E., Raisaro, J.L., et al.: Protecting and evaluating genomic privacy in medical tests and personalized medicine. In: Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, pp. 95–106. ACM (2013)
3. Baldi, P., Baronio, R., et al.: Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 691–702. ACM (2011)

4. Erlich, Y., Narayanan, A.: Routes for breaching and protecting genetic privacy. Nat. Rev. Genet. **15**(6), 409–421 (2014)
5. Galperin, M.Y., et al.: The 2015 nucleic acids research database issue and molecular biology database collection. Nucleic Acids Res. **43**(D1), D1–D5 (2015)
6. Gymrek, M., McGuire, A.L., Golan, D., Halperin, E., Erlich, Y.: Identifying personal genomes by surname inference. Science **339**(6117), 321–324 (2013)
7. Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., et al.: Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS Genet. **4**(8), e1000167 (2008)
8. Humbert, M., Ayday, E., et al.: Addressing the concerns of the lacks family: quantification of kin genomic privacy. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 1141–1152. ACM (2013)
9. Iossifov, I., ORoak, B.J., Sanders, S.J., et al.: The contribution of de novo coding mutations to autism spectrum disorder. Nature **515**(7526), 216–221 (2014)
10. Kayser, M., de Knijff, P.: Improving human forensics through advances in genetics, genomics and molecular biology. Nat. Rev. Genet. **12**(3), 179–192 (2011)
11. Kobayashi, E., Sakurada, T., et al.: Public involvement in pharmacogenomics research: a national survey on patients attitudes towards pharmacogenomics research and the willingness to donate DNA samples to a DNA bank in japan. Cell Tissue Banking **12**(2), 71–80 (2011)
12. Naveed, M., Ayday, E., Clayton, E.W., Fellay, J., Gunter, C.A., Hubaux, J.P., Malin, B.A., Wang, X.: Privacy in the genomic era. ACM Comput. Surv. (CSUR) **48**(1), 6 (2015)
13. Nyholt, D.R., Yu, C.E., Visscher, P.M.: On jim watson's APOE status: genetic information is hard to hide. Eur. J. Hum. Genet. **17**(2), 147 (2009)
14. Pakstis, A.J., Speed, W.C., Fang, R., Hyland, F.C., et al.: SNPs for a universal individual identification panel. Hum. Genet. **127**(3), 315–324 (2010)
15. Schadt, E.E., Woo, S., Hao, K.: Bayesian method to predict individual SNP genotypes from gene expression data. Nat. Genet. **44**(5), 603–608 (2012)
16. Storr, C.L., Or, F., et al.: Genetic research participation in a young adult community sample. J. Commun. Genet. **5**(4), 363–375 (2014)
17. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **10**(05), 557–570 (2002)
18. Sweeney, L., Abu, A., Winn, J.: Identifying participants in the personal genome project by name. Available at SSRN 2257732 (2013)

User-Centric Privacy-Preserving Collection and Analysis of Trajectory Data

Cristina Romero-Tris(✉) and David Megías

Estudis d'Informàtica Multimèdia i Telecomunicació,
Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC),
Parc Mediterrani de la Tecnologia. Av. Carl Friedrich Gauss, 5,
08018 Catalonia, Castelldefels (Barcelona), Spain
{cromerotr,dmegias}@uoc.edu

Abstract. Due to the increasing use of location-aware devices such as smartphones, there is a large amount of available trajectory data whose improper use or publication can threaten users' privacy. Since trajectory information contains personal mobility data, it may reveal sensitive details like habits of behavior, religious beliefs, and sexual preferences. Current solutions focus on anonymizing data before its publication. Nevertheless, we argue that this approach gives the user no control about the information she shares. For this reason, we propose a novel approach that works inside users' mobile devices, where users can decide and configure the quantity and accuracy of shared data.

1 Introduction

Over the last few years, location-aware technologies such as global positioning system (GPS) or location-based services (LBS) have caused the amount of data related to trajectories to significantly increase. On the one hand, mining and analyzing these spatio-temporal trajectory datasets can provide a valuable service (e.g., inferring traffic congestion, tracking infections, etc.). On the other hand, trajectory data often contain information about individuals. Knowledge of mobility data, in some cases combined with quasi-identifiers (gender, age, postal code, etc.), may reveal sensitive data which can threaten privacy (e.g., information about home addresses, lifestyle, religious beliefs, ideology, etc.).

To cope with this problem, there is an emergent field of the literature that focuses on proposing new solutions. For example, Abul et al. [1] propose the (k, δ)-anonymity model, which modifies a location polyline to be represented by a single cylinder of radius δ. Then, k trajectories co-localized inside the same cylinder are indistinguishable from each other. Terrovitis and Mamoulis [2] propose an algorithm that suppresses the existence of certain points in the trajectories. The challenge in this case is how to find the optimal set points to delete, with the minimum possible information loss. The authors propose a greedy heuristic that assumes that all the adversarial knowledge is known before data publication. Similarly, Pensa et al. [3] propose to remove frequent sequential patterns. They transform sequences by adding, deleting, or substituting some points of

© Springer International Publishing Switzerland 2016
J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 245–253, 2016.
DOI: 10.1007/978-3-319-29883-2_17

the trajectory. Yarovoy et al. [4] employ the Hilbert curve [5] in order to map a multi-dimensional space to one dimension. The purpose of this is finding the nearest neighbors at every point of the trajectory. Then, the neighbors are used to create anonymization groups to generalize trajectory data of each member.

All these works are limited to privacy protection on already collected data. The proposed algorithms work on the server side, before its publication. Nevertheless, we argue that trajectory anonymization would rather be performed a step earlier, in the user side. This protects users from an adversary that gains access to the records stored in the database. Moreover, the advantage of this approach is that users are able to configure the quantity and accuracy of shared information before it is stored in the database.

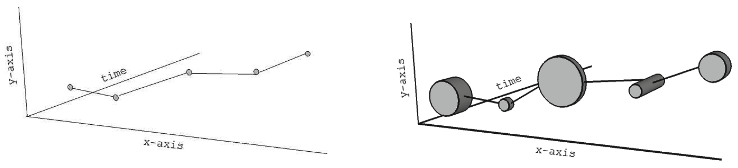
For this purpose, our system relies on a personalized trajectory anonymization method that transforms spatio-temporal points into uncertain points, where the exact location and timing are distorted according to a set of user-defined parameters. Then, users are grouped to execute a protocol and obtain k-anonymity, being k a user-defined parameter according to her privacy requirements.

This paper is organized as follows: Sect. 2 defines relevant concepts for our system. Section 3 describes our proposal in detail. Privacy is analyzed in Sects. 4 and 5 concludes the paper.

2 Problem Definition

This section describes some background tools or concepts that are necessary to understand our system.

Definition 1 (Trajectory). *A trajectory T of length $|T|$ is an ordered list of spatio-temporal points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})$ where (x_i, y_i, t_i) means that the user was at a physical location with Cartesian coordinates (x_i, y_i) at instant t_i. During the time segment $[t_i, t_{i+1}]$ the user is assumed to move along a straight line from (x_i, y_i) to (x_{i+1}, y_{i+1}). Figure 1(a) represents the definition of a trajectory with five points. The three-dimensional space represents the time and the Cartesian coordinates of the position (abscissae and ordinates).*



(a) Trajectory (b) Anonymized trajectory

Fig. 1. Schematic representation of a trajectory before and after anonymization

Definition 2 (Uncertain point). *For a specific spatio-temporal point (x_i, y_i, t_i), its anonymized version is another vector $(cx_i, cy_i, r_i, a_i, b_i)$ where (cx_i, cy_i) are the Cartesian coordinates of the center of a circle of radius r_i that contains (x_i, y_i), and $[a_i, b_i]$ is a time interval that contains t_i.*

Definition 3 (Anonymized trajectory). *An anonymized trajectory T' of length $|T'|$ is an ordered list of uncertain point vectors $(cx_1, cy_1, r_1, a_1, b_1), (cx_2, cy_2, r_2, a_2, b_2), \ldots, (cx_{|T'|}, cy_{|T'|}, r_{|T'|}, a_{|T'|}, b_{|T'|})$. During the time between $[a_i, b_i]$ and $[a_{i+1}, b_{i+1}]$, the user is assumed to move along a line from any point inside the circle defined by (cx_i, cy_i, r_i) to any point inside $(cx_{i+1}, cy_{i+1}, r_{i+1})$.*

Figure 1(b) represents the trajectory of Fig. 1(a) after being anonymized. The anonymization transforms a point into a circle of variable radius, and an instant into a time interval. Thus, for each spatio-temporal point, a cylinder is obtained.

Definition 4 (Anonymized sub-trajectory). *Given an anonymized trajectory T', an anonymized sub-trajectory s' of size $|s'| \leq |T'|$ is an ordered subset of the vectors composing T'. The conditions to be fulfilled are: (1) in order not to be a single point, the size of the sub-trajectory must be $|s'| > 1$; and (2) the order of the vectors in s' must be the same as in T'.*

Definition 5 (Similar anonymized sub-trajectories). *Having initially two anonymized trajectories: $s' = (cx_{11}, cy_{11}, r_{11}, a_{11}, b_{11}), (cx_{21}, cy_{21}, r_{21}, a_{21}, b_{21}), \ldots, (cx_{|s'|}, cy_{|s'|}, r_{|s'|}, a_{|s'|}, b_{|s'|})$, and $s'' = (cx_{12}, cy_{12}, r_{12}, a_{12}, b_{12}), (cx_{22}, cy_{22}, r_{22}, a_{22}, b_{22}), \ldots, (cx_{|s''|}, cy_{|s''|}, r_{|s''|}, a_{|s''|}, b_{|s''|})$, we define two system parameters θ_L and θ_T that represent the maximum distance to consider two points similar in terms of location and time, respectively. Then, we consider that s' and s'' are similar if these conditions are fulfilled:*

1. $|s'| = |s''|$
2. *For* $i = 1, 2, \ldots, |s'|$:
   - *(a)* $\sqrt{(cx_{i1} - cx_{i2})^2 + (cy_{i1} - cy_{i2})^2} < (r_{i1} + r_{i2} + \theta_L)$. *This means that the Euclidean distance between both circles is lower that* θ_L.
   - *(b)* $(((b_{i2} + \theta_T) > a_{i1})$ *and* $((b_{i2} + \theta_T) < b_{i1}))$ *or* $(((b_{i1} + \theta_T) > a_{i2})$ *and* $((b_{i1} + \theta_T) < b_{i2}))$. *This means that the time intervals are separated less than* θ_T *occurring* $|s''|$ *before* $|s'|$ *or* $|s'|$ *before* $|s''|$.

3 Protocol Description

This section describes the proposed system. We assume that User U_i's device already contains her trajectory $T(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})$; and that a server $\boldsymbol{\mathcal{S}}$ requests the trajectory information.

Regarding cryptography, users employ a n-out-of-n threshold ElGamal encryption [6], where n users share a public key y and the corresponding unknown private key α is divided into n shares α_i. Using this protocol, a certain message m can be encrypted with the public key y and it can only be decrypted if all n users collaborate in the process.

3.1 Creation of the Anonymization Group

The process starts when the server $\boldsymbol{\mathcal{S}}$ sends a request to collect trajectory data from users. Then, users that are willing to share their information send a confirmation to the server. Let N be the total number of users who send a confirmation.

The users who want to participate in the process must be included in groups of size n, where n is a predetermined system parameter. In order to prevent $\boldsymbol{\mathcal{S}}$ from grouping users as it wishes, a join coin-tossing protocol adapted from [7] is executed. This protocol assumes that every user U_i already has a personal public key (pk_i) provided by a PKI. The protocol employs two random oracles (which in practice can be computed as pseudo-random functions [8]) $H_1 = 0, 1^* \to 0, 1^k$ (where k is the bit-length of the public key), and $H_2 = 0, 1^* \to 0, 1^{N \cdot \log N}$. The following steps are executed:

1. Every user U_i generates a random r_i and sends $H_1(IP_i, pk_i, r_i)$ to $\boldsymbol{\mathcal{S}}$, where IP_i is a concatenation of the public and private IP address of U_i.
2. U_i waits a short predefined time.
3. $\boldsymbol{\mathcal{S}}$ sends $H_1(IP_i, pk_i, r_i)$ for $i = 1, \ldots, N$ to all the users.
4. Then, each user U_i computes $h = H_2(H_1(IP_i, pk_i, r_i), \ldots, H_1(IP_N, pk_N, r_N)$ and divides the result h into chunks of size $\log N$, denoted $h_1, \ldots, h_N$.
5. User U_i takes h_i as her identifier.
6. Grouping is carried out by taking groups of n parties according to the sorting. That is, for $i = 1, \ldots, \lfloor N/n \rfloor$, the ith group is formed by users with identifiers $(h_{n\cdot(i-1)+1}, \ldots, h_{n\cdot i})$
7. $\boldsymbol{\mathcal{S}}$ sends the IP addresses of each user to the members of her group.
8. The members of each group send each other their IP addresses, public key and the random r_i they used at the beginning of the protocol.
9. Each group member computes $H_1(IP_j, pk_j, r_j)$ for every user U_j in her group, and verifies that it matches what she received from $\boldsymbol{\mathcal{S}}$. Additionally, she computes H_2 as in Step 4 to verify that all the IP addresses assigned to her group are inside it. If any verification fails, she sends *abort* to the group members and exits the system.

3.2 Trajectory Anonymization

In this phase, U_i decides the granularity of spatio-temporal disclosure for every point of T. This means that, for every point (x_i, y_i, t_i), the user will obtain a vector $(cx_i, cy_i, r_i, a_i, b_i)$ based on the values that she chooses for:

– *The radius* r_i. This parameter, expressed in kilometers, is the radius of the circle that contains the Cartesian coordinates (x_i, y_i). A larger radius means higher generalization and hence, higher distortion. Based on the value chosen by the user, we randomly select a point (cx_i, cy_i) that fulfills the equation $(x_i - cx_i)^2 + (y_i - cy_i)^2 \leq r_i^2$.
– *The time gap* γ_i. This parameter, expressed in hours (but working with real numbers), indicates the time difference between a_i and b_i. Therefore, to obtain these values we randomly choose a value v between 0 and γ_i. Then, we compute $a_i = t_i - v$, and $b_i = t_i + \gamma_i - v$.

Repeating this process for all the points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_{|T|}, y_{|T|}, t_{|T|})$ in T, we obtain the anonymized trajectory $|T'| = (cx_1, cy_1, r_1, a_1, b_1), (cx_2, cy_2, r_2, a_2, b_2), \ldots, (cx_{|T'|}, cy_{|T'|}, r_{|T'|}, a_{|T'|}, b_{|T'|})$.

Note that the user can also completely remove a spatio-temporal point from the list. Therefore, $|T|$ and $|T'|$ might not be equal.

3.3 Sub-trajectory Extraction

The sub-trajectory extraction depends on two parameters. The first one is the number of sub-trajectories to extract (τ), and the second one is the maximum number of points that each sub-trajectory should contain, μ. Having τ, μ as system parameters, Algorithm 1 shows how to extract the sub-trajectories:

Algorithm 1. Sub-trajectory extraction algorithm

```
procedure SUB-TRAJECTORY EXTRACTION
Input: $\tau$, $\mu$, anonymized trajectory $T'[]$ as a table of spatio-temporal points.
Output: Table subtraj of anonymized sub-trajectories
subtraj:=new table[$\tau$]
count:=0
Loop: i:=0 to $\tau$ by 1
    Loop: j:=0 to $\tau/|T'|$ by 1
        subtraj[i] := $T'[count]$
        count++
    Loop-end: j
Loop-end: i
Loop: i:=0 to $\tau$ by 1
    While (size of subtraj[i] > $\mu$)
        Remove one random element from subtraj[i]
    While-end
Loop-end: i
end procedure
```

3.4 Fake Sub-trajectory Generation

Similarly to the real sub-trajectory extraction, the fake sub-trajectory generation needs two parameters: (1) the number of fake sub-trajectories to generate (τ'); and (2) the maximum number of points that each fake sub-trajectory should contain, μ'. There are many works in the literature that describe how to generate a fake trajectory. The generation of a particular algorithm for this is out the scope of this paper. For our purposes, we employ the method proposed in [9].

3.5 Distribution of Sub-trajectories

In this phase, the real and fake sub-trajectories are distributed among the group of users $\{U_1, \ldots, U_n\}$. In order to prevent one malicious member of the group from learning all the sub-trajectories that belong to another user, the group executes a multi-party privacy-preserving protocol composed by three phases: group key generation, anonymous sub-trajectories retrieval, and query submission.

Group Key Generation.

1. Users $\{U_1, \dots, U_n\}$ generate a large prime p where $p = 2q+1$ and q is a prime too. Next, they pick an element $g \in \mathbb{Z}_q^*$ of order q.
2. In order to generate the group key, each user U_i performs the following steps:
   (a) Generates a random number $a_i \in \mathbb{Z}_q^*$.
   (b) Calculates her own share $y_i = g^{a_i} \bmod p$.
   (c) Broadcasts a commitment $h_i = \mathcal{H}(y_i)$, where $\mathcal{H}$ is a one-way function.
   (d) Broadcasts y_i to the other members of the group.
   (e) Checks that $h_j = \mathcal{H}(y_j)$ for $j = (1, \dots, n)$.
   (f) Calculates the group key using the received shares: $y = \prod_{1 \le j \le n} y_j = g^{a_1} \cdot g^{a_2} \cdot \dots \cdot g^{a_n}$

3.6 Anonymous Sub-trajectory Retrieval

Assuming that each user U_i has $(\tau + \tau')$ sub-trajectories: $s_{i1}, s_{i2}, \dots, s_{i(\tau+\tau')}$

1. User U_i encrypts real and fake sub-trajectories as plaintext. For each s_{ij}, U_i generates a random number r_{ij} and encrypts s_{ij} with y: $c_{ij}^0 = E_y(s_{ij}, r_{ij}) = (g^{r_{ij}}, s_{ij} \cdot y^{r_{ij}}) = (c1_{ij}, c2_{ij})$.
2. For $i = (2, \dots, n)$, $j = (1, \dots, (\tau + \tau'))$ each user U_i sends c_{ij}^0 to the first member of the group (U_1).
3. For $i = (1, \dots, n-1)$, each user U_i performs the following operations:
   (a) Receives the list of ciphertexts $\left\{c_{11}^{i-1}, c_{12}^{i-1}, \dots, c_{n(\tau+\tau')}^{i-1}\right\}$.
   (b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in [6]. The resulting list of ciphertexts is denoted as $\left\{c_{11}^{i-1\prime}, \dots, c_{n(\tau+\tau')}^{i-1\prime}\right\}$.
   (c) The list of ciphertexts $\left\{c_{11}^{i-1\prime}, \dots, c_{n(\tau+\tau')}^{i-1\prime}\right\}$ is re-masked using the re-masking algorithm described in [10] with a key $y' = \prod_{w=i+1}^{n} g^{\alpha_w}$. As a result, U_i obtains a re-encrypted version $\left\{e_{11}^{i-1}, \dots, e_{n(\tau+\tau')}^{i-1}\right\}$.
   (d) Permutes the ciphertexts at random, obtaining $\left\{e_{\sigma(11)}^{i-1}, \dots, e_{\sigma(n(\tau+\tau'))}^{i-1}\right\}$
   (e) Sends $\left\{c_{11}^{i}, \dots, c_{n(\tau+\tau')}^{i}\right\} = \left\{e_{\sigma(11)}^{i-1}, \dots, e_{\sigma(n(\tau+\tau'))}^{i-1}\right\}$ to U_{i+1}.
4. The last user U_n performs the following operations:
   (a) Receives the list of ciphertexts $\left\{c_{11}^{i-1}, \dots, c_{n(\tau+\tau')}^{i-1}\right\}$.
   (b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in [6]. At this point, U_n owns the sub-trajectories cleartexts, so she broadcast them to $\{U_1, \dots, U_{n-1}\}$.

This is the central part of the protocol which has a higher cost and complexity. In this phase, each user performs $(\tau + \tau')$ encryptions, and $n \cdot (\tau + \tau')$ decryptions. Regarding the number of messages, each user U_i sends one long message (containing $n \cdot (\tau + \tau')$ ciphertexts) to U_{i+1}, except for the last user U_n, who sends $n - 1$ short messages (containing each one $n \cdot (\tau + \tau')$ cleartexts).

3.7 Sub-trajectory Submission and Retrieval

1. Each group member U_i must send $(\tau + \tau')$ sub-trajectories to the server $\boldsymbol{\mathcal{S}}$. More specifically, from the received list, user U_i submits the sub-trajectories found between positions $i \cdot n$ and $i \cdot n + \tau + \tau'$.
2. Upon receiving the $(\tau + \tau')$ answers from the server, each user broadcasts them to the rest of the group members. Then, each user takes the answers that corresponds to her original sub-trajectories.
3. The answer of the server for each sub-trajectory is ϕ, the number of sub-trajectories in the database similar to the one submitted according to Definition 5. Sub-trajectories where $\phi < k$ must be removed from the anonymized trajectory, and hence, they are put in a list L to be used in next step.

3.8 Anonymized Trajectory Trimming

Using Algorithm 2 the list L of real sub-trajectories is removed from the anonymized trajectory T' of each user. The resulting anonymized trajectory is sent to the server $\boldsymbol{\mathcal{S}}$. The server can store it in its database for future analysis or publication.

Algorithm 2. Anonymized trajectory trimming algorithm

```
procedure ANONYMIZED TRAJECTORY TRIMMING
Input: table of sub-trajectories to be removed $L[]$, anonymized trajectory $T'[]$
Output: Resulting anonymized trajectory $T'$
$ls :=$ size of $L$
Loop: $i := 0$ to $ls$ by 1
    For every spatio-temporal vector $q$ in $L_i$
        Remove $q$ from $T'$
    Loop-end: $i$
end procedure
```

4 Privacy Analysis

In this section, we analyze the system in terms of privacy. First of all, the ElGamal cryptosystem is semantically secure under the Decisional Diffie-Hellman assumption. This means that a dishonest user cannot know if two different ciphertexts will result into the same cleartext after decryption.

Therefore, every time that a ciphertext c_i is transformed by a group member (i.e., remasked and permuted), the attacker can only link the result to c_i by random guessing, the intermediate re-maskings and permutations preventing her from finding the links between them. Hence, the probability of success is $1/(n(\tau + \tau'))$, since there are $n(\tau + \tau')$ ciphertexts involved in the process.

The proposed protocol also relies on the server to help users achieve k-anonymity by answering their requests. Moreover, the server is in charge of creating the groups. The steps presented in Sect. 3.1 adapted from [7] prevent the server from maliciously grouping users. The security of this protocol is analyzed in [7]. The authors compute the probability of a bad grouping, i.e., having $n-1$ dishonest users together with a single honest party. Assuming that $N \gg t$, the authors state that this probability is approximately $(\frac{t}{N})^{n-2} \cdot N$. For example, if one million users participate in the system, and the server controls one thousand, then the probability of a bad grouping is under 10^{-48}.

5 Conclusions and Future Work

In this paper, we argue that trajectory data would rather be protected in the client-side, before they are stored in the server or disclosed to a third entity. To the best of our knowledge, this is the first work that introduces trajectory anonymization in the user's device, giving users control over the information they send to the server and providing k-anonymity.

However, our work is on an early stage of development and there are some interesting open research problems that need to be addressed in the future. More specifically, experimental results are necessary in order to know how the system behaves for different parameter configurations. In order to do this, we need to implement the system and execute it in a real or simulated environment.

Acknowledgments. This work was partly funded by the Spanish Government through grants TIN2011-27076-C03-02 "CO-PRIVACY" and TIN2014-57364-C2-2-R "SMARTGLACIS".

References

1. Abul, O., Bonchi, F., Nanni, M.: Never walk alone: uncertainty for anonymity in moving objects databases. In: Proceedings of the IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 376–385. IEEE Computer Society, Washington, DC (2008)
2. Terrovitis, M., Mamoulis, N.: Privacy preservation in the publication of trajectories. In: 9th International Conference on Mobile Data Management, MDM 2008, pp. 65–72. IEEE (2008)
3. Pensa, R.G., Monreale, A., Pinelli, F., Pedreschi, D.: Pattern-preserving k-anonymization of sequences, its application to mobility data mining. In: PiLBA, pp. 1–10 (2008)

4. Yarovoy, R., Bonchi, F., Lakshmanan, L.V.S., Wang, W.H.: Anonymizing moving objects: how to hide a MOB in a crowd? In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT 2009, pp. 72–83. ACM, New York (2009)
5. Hilbert, D.: Ueber die stetige abbildung einer line auf ein flächenstück. Mathematische Annalen **38**(3), 459–460 (1891)
6. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
7. Lindell, Y., Waisbard, E.: Private web search with malicious adversaries. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 220–235. Springer, Heidelberg (2010)
8. Berman, I., Haitner, I.: From non-adaptive to adaptive pseudorandom functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 357–368. Springer, Heidelberg (2012)
9. Gkoulalas-Divanis, A., Verykios, V.S.: A privacy-aware trajectory tracking query engine. ACM SIGKDD Explor. Newsl. **10**(1), 40–49 (2008)
10. Abe, M.: Mix-networks on permutation networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999)

The Leaking Battery
A Privacy Analysis of the HTML5 Battery Status API

Łukasz Olejnik[1](✉), Gunes Acar[2], Claude Castelluccia[1], and Claudia Diaz[2]

[1] INRIA Privatics, Grenoble, France
{lukasz.olejnik,claude.castelluccia}@inria.fr
[2] KU Leuven, ESAT/COSIC and iMinds, Leuven, Belgium
{gunes.acar,claudia.diaz}@esat.kuleuven.be

Abstract. We highlight privacy risks associated with the HTML5 Battery Status API. We put special focus on its implementation in the Firefox browser. Our study shows that websites can discover the capacity of users' batteries by exploiting the high precision readouts provided by Firefox on Linux. The capacity of the battery, as well as its level, expose a fingerprintable surface that can be used to track web users in short time intervals.

Our analysis shows that the risk is much higher for old or used batteries with reduced capacities, as the battery capacity may potentially serve as a tracking identifier. The fingerprintable surface of the API could be drastically reduced without any loss in the API's functionality by reducing the precision of the readings. We propose minor modifications to Battery Status API and its implementation in the Firefox browser to address the privacy issues presented in the study. Our bug report for Firefox was accepted and a fix is deployed.

1 Introduction

HTML5 Battery Status API enables websites to access the battery state of a mobile device or a laptop. Using the API, websites can check the battery level of a device and use this information to switch between energy-saving or high-performance modes. All the information exposed by the Battery Status API is available without users' permission or awareness.

The *"Security and privacy considerations"* section of the W3C specification that describes the Battery Status API, states the following: *"The information disclosed has minimal impact on privacy or fingerprinting, and therefore is exposed without permission grants"* [14]. Our findings, however, show that the API, as implemented by the Firefox browser on GNU/Linux operating system, enables fingerprinting and tracking of devices with batteries in short time intervals.

As of June 2015, Firefox, Chrome and Opera are the only three browsers that supported the Battery Status API [3]. Although the potential privacy problems of the Battery Status API were discussed by Mozilla and Tor Browser developers as early as in 2012 [1,2,22], neither the API, nor the Firefox implementation,

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 254–263, 2016.
DOI: 10.1007/978-3-319-29883-2_18

has undergone a major revision. We hope to draw attention to this privacy issue by demonstrating the ways to abuse the API for fingerprinting and tracking.

We present an analysis of Battery Status API as implemented by Firefox on GNU/Linux. Our analysis indicate that seemingly innocuous information provided by the Battery Status API can serve as a tracking identifier when implemented incorrectly.

The core contributions of this work are:

1. *We present a new device fingerprinting vector based on the Battery Status API.* We show that the Firefox's implementation of the Battery Status API allows the discovery of battery's capacity, provides short-term identifiers that facilitates tracking and potentially can be used for reinstantiating identifiers (*respawning*).
2. *We propose a solution that reduces the Battery Status API's fingerprintable surface* by rounding the level readings provided by the API. Our fix does not cause any loss in the effective functionality of the API. We filed a bug report for Mozilla Firefox to communicate the problem and the proposed solution [20]. The fix was quickly implemented and deployed by Mozilla engineers in response to our bug report.

2 Related Work

The Panopticlick [9] study by Eckersley demonstrated the feasibility of browser fingerprinting for online tracking by measuring the entropy present in the browser properties such as screen size, list of system fonts and browser plugins. Other researchers demonstrated the many ways browsers can be fingerprinted using different properties, such as clock skew [13], font metrics [10], network protocol characteristics [7], JavaScript engine performance [16], WebGL and canvas rendering [17].

Recently, studies measured the prevalence of the browser fingerprinting on the Web [4,5,19], suggesting that questionable practices such as proxy circumvention or stealthy techniques to exercise browser fingerprinting are commonly used by the websites.

In a similar vein, researchers studied *zombie cookie* (or *evercookie*) which is another tracking mechanism that can be used to reconstruct tracking identifiers - even if the user decides to clear her history [12] - with the use of Flash cookies [21], ETags [6] and other vectors.

A recent work, independent to ours, includes a very short note about the possible use of Battery API as a potential privacy risk vector [18]. The problem is not further described or analyzed, and the authors only mention potential risks due to monitoring of charging and discharging rates. In essence, our analysis is more extensive and detailed. Moreover, we describe a clear risk in relation to Firefox browser and study it in detail.

3 Background

3.1 Battery Status API

World Wide Web Consortium's (W3C) Battery Status API allows the reading of battery status data. Among the offered information are the current battery level and predicted time to charge or discharge. The respective properties `level`, `chargingTime` and `dischargingTime` can be accessed in JavaScript by first calling the `navigator.getBattery()` method[1] to get a `BatteryManager` object which then exposes these properties.

The API does not require user permission to read the battery information, any website or third-party scripts included on them, can use the API. The API also does not require browsers to notify users when the battery information is accessed. That allows website and third-party scripts to access the battery information transparently - without users' awareness.

The Battery Status API also provides JavaScript event handlers that allow the monitoring of updates to battery status. The API defines the `level` property as a double-precision floating-point number, taking values between 0 (depleted) and 1.0 (full) [14].

3.2 Power Information Under Linux

In our exploratory survey of the Battery Status API implementations, we observed that the battery level reported by the Firefox browser on GNU/Linux was presented to Web scripts with *double* precision. An example battery level value observed in our study was 0.9301929625425652. We found that on Windows, Mac OS X and Android, the battery level reported by Firefox has just two significant digits (e.g. 0.32).

Analyzing the Firefox source code, we found out that the battery level is read from *UPower*, a Linux tool allowing the access to the UPower daemon [11]. The UPower daemon provides access to comprehensive power-management data about the device. Specifically, it enables the access to detailed information about the battery status such as capacity, level, voltage and provides estimates about the discharge and charge times.

Analyzing the UPower source code (`linux/up-device-supply.c`) to understand how it computes the battery level, we compiled the following equations:

$$BatteryLevel = 0.01 \times Percentage \tag{1a}$$

$$Percentage = 100.0 \times \frac{Energy}{EnergyFull} \tag{1b}$$

$$Energy = \frac{ChargeNow}{1,000,000} \times DesignVoltage \tag{1c}$$

[1] Firefox does not implement `navigator.getBattery()` method, instead, it exposes a `navigator.battery` object.

$$EnergyFull = \frac{ChargeFull}{1,000,000} \times DesignVoltage \tag{1d}$$

The *Energy* is the current amount of energy present in the battery and measured in *watt-hours*. *EnergyFull* is also measured in *watt-hours* and represents the maximum possible amount of energy that can be stored in the battery. The *ChargeNow* and *ChargeFull* are measured in μAh and represent the current and maximum charge capacities of the battery respectively. Note that, due to the aging of the battery, *EnergyFull* tend to be lower than the design capacity of the battery, moreover, it can also change after a discharge, followed by a full charge – possibly for calibration purposes. Although many batteries share the same design capacities (e.g. 48.84 Wh or 62.16 Wh), as they age in time, their capacities may be reduced in different amounts, resulting in a diverse number of possible *EnergyFull* values, which are internally stored with four decimal places (e.g. 42.1678).

Since Firefox browser under Linux is accessing the UPower-provided data, it reads the *Percentage* value in 64 bit double precision floating point format and multiplies it by 0.01 to obtain the battery *level* as shown in Eq. (1a). The *level* value is then exposed to website scripts through the Battery Status API in double precision.

As noted above, the *EnergyFull* value may change, as the battery capacity degrades. The UPower daemon updates the current capacity by comparing the *EnergyFull* to the latest value stored when the battery is fully charged.

4 Tracking with the Battery Status API

We measure the extent to which it is possible to link (and track) a device with battery using the battery level and charge/discharge time readouts. We observe how it could be leveraged for fingerprinting and tracking across sites. Moreover, we present a method to recover the battery's effective capacity (*EnergyFull*) using the precise battery level readouts provided by Firefox on Linux.

4.1 Tracking Across Sites

In this section, we discuss several potential fingerprinting and tracking scenarios. A third-party script that is present across multiple websites can link users' visits in a short time interval by exploiting the battery information provided to Web scripts. In order to do that, scripts can use the values of battery level, dischargingTime and chargingTime. The readings will be consistent on each of the sites, because of the fact that the update intervals (and their times) are identical. This could enable the third-party script to link these concurrent visits. Moreover, in case the user leaves these sites but then, shortly afterwards, visits another site with the same third-party script, the readings would likely be utilized to help in linking the current visit with the preceding ones.

Below we analyze more specific cases.

Frequency of Battery Status Changes. We analyzed the update rates under different computing loads (such as watching a movie, simply browsing the Web, etc).

We tested the rate of these changes by setting up a simple page and registering JavaScript event handlers for battery status changes; we monitored JavaScript readouts of level and dischargingTime, as well as the timestamps of these events. We analyzed the collected data for relative time differences between level, chargeTime and dischargeTime changes. The results indicated that for about 30 s, battery status may serve as a static identifier, allowing (e.g.) a third-party script to link visits from the same computer in short time intervals.

Number of Possible Identifiers. In our test setting, the lowest indication of dischargeTime we observed was 355 (in seconds), and highest 40277 s. Assuming all the values spanning a range $(355, 40277)$ are possible, this gives 39922 numbers. We can also assume that users seeing a near-drained battery generally connect their notebooks to AC power. Assuming users start to charge their devices when the battery level is 0.1, this leaves 90 available battery level states (0.11 to 1.0). The number of potential levels denoted by a tuple $(level, dischargeTime)$ would then be a simple multiplication 90×39922 and the final number of possible states would be 3592980, which only accounts for the discharging state. Using the information about the battery charge $(chargingTime)$ could effectively double the number of possible states. The probability of a $(level, dischargeTime)$ collision (between different users, and assuming a uniform distribution) is therefore low and for a short time frame this would effectively be a unique identifier.

However, we emphasize that the dischargeTime levels can be subject to frequent changes, in response to change in the users' computer use patterns. This means that, in practice, the risk of long-term tracking with this information may be negligible. Moreover, depending on the battery level, some chargeTime or dischargeTime values may not be observed in practice[2]. Yet, the available combinations could be used to distinguish users behind a NAT (Network Address Translation). In such a setting, the computers may have similar fingerprints [9] and often identical public IP addresses. The readouts from the battery may allow distinguishing these users.

Reconstructing User Identifiers in Short-Time Intervals. Users who try to re-visit a website with a new identity may use browsers' private mode or clear cookies and other client side identifiers. When consecutive visits are made within a short interval, the website can link users' new and old identities by exploiting battery level and charge/discharge times. The website can then reinstantiate users' cookies and other client side identifiers, a method known as *respawning* [21]. Note that, although this method of exploiting battery data as a linking identifier would only work for short time intervals, it may be used against power users who can not only clear their cookies but can go to great lengths to clear their evercookies.

[2] For instance, 355 s dischargeTime may be too short for a full battery or, 40277 s dischargeTime may be too long for a battery with level 0.1.

5 Detecting Battery Capacity

In addition to using battery level and charge/discharge times for linking visits in short time intervals, Battery Status API can be used to infer the current battery capacity (*EnergyFull*) of a device if it allows high precision level readouts. In this section, we analyze the possibility of fingerprinting a device by exploiting high precision battery level readouts provided by the Firefox on Linux operating system.

We found that using the 64-bit double precision floating point battery level readouts from Firefox on Linux, it is possible to discover the value of *EnergyFull*, which signifies the actual battery capacity. We emphasize that our method only works for UPower and Firefox on Linux, and during our study we encountered some computers for which we cannot recover the capacity with our method. This can be due to the differences in how processors handle floating point calculations[3] or measurement errors in UPower.

The attack works by using the Eq. 1a–1d by reading the battery level and finding candidate *Energy*, *EnergyFull* and *Voltage* levels which may give this floating point number reading. In order to do this, attacker may either brute-force the candidate values by testing all possible values or precompute a lookup table.

5.1 Test Method

Assuming a uniform space of EnergyFull values (X, Y), we tested all the hypothetical level readouts to detect the possible identifiers. It is obvious that, for a given level reading, several possibilities for EnergyFull level may exist. However, if the attacker has access to multiple battery level readouts, the number of collisions becomes significantly smaller. We analyzed the number of potential EnergyFull candidates as a function of the battery level readouts.

In other words, we computed the number of collisions for one battery level readout, $State_1$. For each such possible readout level, we simulated another different readout level (different than the one in preceding state), $State_2$ (battery levels in $State_1$ and $State_2$ are different). We compared the candidate EnergyFull values in $State_1$ and $State_2$ and intersecting the sets of possible EnergyFull levels, we effectively decreased the number of candidate EnergyFull values. The number of EnergyFull candidates for a total of 1559 battery level readings are displayed on Fig. 1. Figure 2, on the other hand, shows the reduction of EnergyFull candidates when a script is able read the battery level multiple times from the same device. The figure is based on 1559 battery level readings collected from a laptop running Ubuntu 12.04 operating system. We highlight that such analysis is made possible due to the fixed space a floating-point value can represent, and relatively limited capacities of batteries used in practice[4].

[3] See, for example, [8,15] on the "floating-point determinism problem."

[4] Observe that, possible capacities in this calculations include the reduced battery capacities (e.g. not limited to battery capacities on the market). Still, we could find

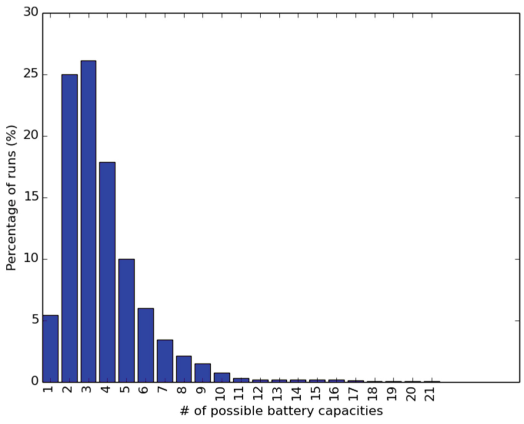


Fig. 1. Distribution of number of candidate battery EnergyFull values for a total of 1559 battery level readings (runs). In 5 % of the cases the attacker can detect the battery capacity with just one reading.

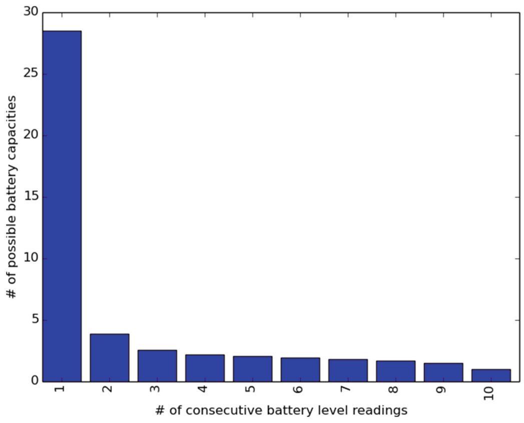


Fig. 2. Average number of candidate battery EnergyFull values as a function of consecutive battery level readings. Attacker can significantly reduce the number of candidate battery capacities if he can read the battery level multiple times.

6 Defense

In the following subsections we outline possible defenses against the exploitation of the Battery Status API for fingerprinting and tracking.

the candidate capacities on a off-the-shelf computer without a significant computation overhead. We believe, an adversary with moderate storage resources can easily build a lookup table to further reduce the computation time.

6.1 Limiting the Precision of Level Readouts

In order to limit the tracking and fingerprinting potential of the Battery Status API, the implementations should avoid providing high-precision values. By simply rounding the `level` value of the battery, the threat would be minimized, without losing any functionality of the API. This comment especially applies to platforms where the OS provides high-precision read-outs about the battery.

We filed an appropriate bug report to Firefox implementation, pointing out the inconsistency of level reporting across different platforms [20]. The fix was implemented and deployed as of June 2015.

Moreover, we believe the Battery Status API could mention the risk of exposing high precision readouts in the "Security and privacy considerations" section of the standard. We plan to communicate the results of the study to the editors of the API.

6.2 Asking for User Permission to Access the Battery Status API

We also discussed potential scenarios where even the reduced precision of the level readout and charge/discharge times could constitute a tracking identifier in short time intervals. In these scenarios, the exposed battery information may allow an attacker to reinstantiate tracking identifiers in a manner similar to evercookies.

In order to prevent this, browser vendors might require user permissions for accessing the Battery Status API. Although this has been suggested by some concerned Mozilla developers [2], final decision was to make the API available without permissions. We believe, as a minimum, users should be able to choose to be asked for battery access by Web scripts. As an alternative, browsers can enforce the user permission requirement in their private browsing modes.

To the best of our knowledge, the only browser that has a strong defense against fingerprinting by the Battery Status API is Tor Browser. Tor Browser completely disables the API [22] to thwart possible fingerprinting attempts.

Finally, the information on the API use could be made available to the user to aid transparency. We are advocating for streamlining the information to users, either directly via the browser's user interfaces, or at least by allowing to read the respective information by custom-made browser extensions. In this way, software could allow the users to learn and be aware about the use of the battery information on devices they own.

7 Conclusion

We analyzed the privacy implications of the Battery Status API, with a focus on its implementation in Firefox for Linux operating system. Our analysis shows that the high precision battery level readings provided by Firefox can lead to an unexpected fingerprinting surface: the detection of battery capacity.

In short time intervals, Battery Status API can be used to reinstantiate tracking identifiers of users, similar to evercookies. Moreover, battery information can be used in cases where a user can go to great lengths to clear her evercookies. In a corporate setting, where devices share similar characteristics and IP addresses, the battery information can be used to distinguish devices behind a NAT, of traditional tracking mechanisms do not work.

The analysis of Web standards, APIs and their implementations can reveal unexpected Web privacy problems by studying the information exposed to Web pages. The complex and sizable nature of the new Web APIs and their deeper integration with devices make it hard to defend against such threats. Privacy researchers and engineers can help addressing the risks imposed by these APIs by analysing the standards and their implementations for their effect on Web privacy and tracking. This may not only provide an actionable feedback to API designers and browser manufactureres, but can also improve the transparency around these new technologies.

References

1. Proposal for a smaller battery API (2012). https://groups.google.com/forum/#!searchin/mozilla.dev.webapi/Why20is20the20battery20API20exposed20to20unprivileged20content3F/mozilla.dev.webapi/6gLD78z6ASI/Sz1DH2gWN9wJ. Accessed 24 June 2014
2. Why is the battery API exposed to unprivileged content? (2012). https://groups.google.com/forum/#!topic/mozilla.dev.webapi/V361K7c0olQ/discussion. Accessed 26 March 2014
3. Battery Status API - Can I use... Support tables for HTML5, CSS3, etc (2014). http://caniuse.com/#search=battery. Accessed 24 June 2014
4. Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., Diaz, C.: The web never forgets: persistent tracking mechanisms in the wild. In: 21st ACM Conference on Computer and Communications Security (CCS), pp. 674–689. ACM (2014)
5. Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., Preneel, B.: FPDetective: dusting the Web for fingerprinters. In: 20th ACM Conference on Computer and Communications Security (CCS), pp. 1129–1140. ACM (2013)
6. Ayenson, M., Wambach, D.J., Soltani, A., Good, N., Hoofnagle, C.J.: Flash cookies and privacy II: now with HTML5 and ETag respawning. In: World Wide Web Internet and Web Information Systems (2011)
7. Chen, Y.-C., Liao, Y., Baldi, M., Lee, S.-J., Qiu, L.: OS fingerprinting and tethering detection in mobile networks, pp. 173–179 (2014)
8. Dawson, B.: FloatingPoint Determinism – Random ASCII (2013). https://randomascii.wordpress.com/2013/07/16/floating-point-determinism/. Accessed 31 August 2015
9. Eckersley, P.: How unique is your web browser? In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 1–18. Springer, Heidelberg (2010)
10. Fifield, D., Egelman, S.: Fingerprinting web users through font metrics. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 107–124. Springer, Heidelberg (2015)
11. Hughes, R.: UPower Reference Manual (2010). http://upower.freedesktop.org/docs/. Accessed 22 June 2014
12. Kamkar, S.: Evercookie (2010). http://samy.pl/evercookie. Accessed 24 June 2014

13. Kohno, T., Broido, A., Claffy, K.C.: Remote physical device fingerprinting. IEEE Trans. Dependable Secure Comput. **2**(2), 93–108 (2005)
14. Kostiainen, A., Lamouri, M.: Battery Status API (2012). https://bugzilla.mozilla.org/show_bug.cgi?id=1124127
15. Monniaux, D.: The pitfalls of verifying floating-point computations. ACM Trans. Program. Lang. Syst. (TOPLAS) **30**(3), 12 (2008)
16. Mowery, K., Bogenreif, D., Yilek, S., Shacham, H.: Fingerprinting information in JavaScript implementations. In: Web 2.0 Workshop on Security and Privacy (W2SP), vol. 2. IEEE (2011)
17. Mowery, K., Shacham, H.: Pixel perfect: fingerprinting canvas in HTML5. In: Web 2.0 Workshop on Security and Privacy (W2SP). IEEE (2012)
18. Nakibly, G., Shelef, G., Yudilevich, S.: Hardware fingerprinting using HTML5 (2015). CoRR, arxiv.1503.01408
19. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G., Cookieless monster: exploring the ecosystem of web-based device fingerprinting. In: IEEE Symposium on Security and Privacy (SP), pp. 541–555. IEEE (2013)
20. Olejnik, L.: Bug 1124127 - Round Off Navigator Battery Level on Linux (2015). https://bugzilla.mozilla.org/show_bug.cgi?id=1124127. Accessed 30 February 2015
21. Soltani, A., Canty, S., Mayo, Q., Thomas, L., Hoofnagle, C.J.: Flash cookies and privacy. In: Intelligent Information Privacy Management, AAAI Spring Symposium (2010)
22. Tor Bugs: TorBrowser Bundle. #5293 Hook charging+discharching rates in Battery API (2012). https://trac.torproject.org/projects/tor/ticket/5293. Accessed 24 June 2014

Secure Refactoring with Java Information Flow

Steffen Helke[1](✉), Florian Kammüller[2], and Christian W. Probst[3]

[1] Brandenburgische Technische Universität Cottbus-Senftenberg, Cottbus, Germany
steffen.helke@b-tu.de
[2] Middlesex University, London, UK
f.kammueller@mdx.ac.uk
[3] Technical University of Denmark, Kongens Lyngby, Denmark
cwpr@dtu.dk

Abstract. Refactoring means that a program is changed without changing its behaviour from an observer's point of view. Does the change of behaviour also imply that the security of the program is not affected by the changes? Using Myers and Liskov's distributed information flow control model DLM and its Java implementation Jif, we explore this question practically on common patterns of Refactoring as known from Fowler. We first illustrate on an example the "Extract method" refactoring and how it can endanger confidentiality. We then show how to construct a secure version of this major refactoring pattern by employing Jif to control information flows. Finally, we can show that security leaks as encountered at the outset are not possible anymore.

1 Introduction

Security is a cross-cutting concern. To ensure the security of systems we need to consider placement of security controls at all levels, i.e., the physical, organisational, and application (software) level. Language-based security has been a field of research since the early days of computing [3] but has more recently become literally a mandatory technique as increasingly back doors in program code lead to subtle but crucial leaks of secret information. Hence, even the original Information Flow Control (IFC) techniques [3] are re-implemented for Java [2] nowadays. However, one problem with the brute-force IFC is the tremendous complexity of the analysis process, which makes it inapplicable beyond well defined small applications.

The Decentralized Label Model (DLM) [10] provided a major step forward with respect to applications of IFC for practical programming. Rather than analysing code in an *ad hoc* fashion, DLM provides a method for programmers to *label* their implementation with labels corresponding to *security classes* as specified in the access control policy. DLM supports decentralisation and role-based access control by allowing the labels to directly refer to the *principals*, i.e., users of the system; it allows to specify delegations through *acts-for* relations. DLM has been practically implemented for Java.

Refactoring [4] is a pragmatic technique to increase the quality of software while preserving the properties of its initial design and implementation. It allows

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 264–272, 2016.
DOI: 10.1007/978-3-319-29883-2_19

to improve the code of an implementation by various techniques while preserving the observable behaviour of the code. We aim at integrating refactoring with practical IFC labeling à la DLM to establish a feasible process of constructing secure software: a program that is initially labelled by a team of programmers and security experts can subsequently be improved by programmers *without risking to violate its initial security properties*. In this paper, we show that this process is feasible based on existing techniques of refactoring and IFC for Java.

This paper is structured as follows: Sect. 2 introduces refactoring and gives some pointers to related work. Further we illustrate refactoring on an example and show a potential security issue. We then show how to implement secure refactoring with Java Information Flow (Jif). After a short introduction to Jif, we show in Sect. 3 the major refactoring pattern "Extract method" on practical examples. We provide several simple examples and finally generalize a method for securing this main refactoring pattern with Jif labels. We finally summarize the results of our method for secure refactoring with Jif, illustrating its effectivity on the running example (Sect. 4).

2 Refactoring

An important software engineering task is to increase the quality of software. The external quality of a software artifact, like correctness with respect to a given specification, is directly related to its internal quality. In particular, in the process of maintenance of software it is crucial to preserve the external properties, while being able to enhance its internal quality, for example by restructuring of code. Refactoring [4,9] is a technique that is applied to improve the internal structure of a software artifact to enhance readability of the code, make it more amenable to extensions, and thus support its maintainability.

An important part of the work on refactoring focuses on the programming language or modeling language that is used to describe refactorings. The classic book by Fowler [4] approaches this description by means of examples. There are more formal approaches [7,8], but they usually address refactoring at the specification level. The description of refactorings on specifications is important for developers to understand the patters of refactoring. For the application of refactorings to code, descriptions at the level of the programming language are more suitable since they can be directly implemented in software support tools, so-called Integrated Development Environments (IDEs) like Eclipse. To support this practical application of refactorings, general purpose refactoring languages such as Re$\mathcal{L}$ [11] are used. Re$\mathcal{L}$is a domain specific language independent of any specific programming language, and serves as an outline for our approach.

2.1 Example for Extract Method Refactoring

In this section we explain the basic ingredients of refactoring descriptions. We illustrate refactoring by means of one of the most challenging refactorings

```
public class Printer {
  public void printInfo(){
      int i = 5;
      print (i);

      int count = 0;
      while (i != 0){
        count++;
        i--; }

      print(count);
  }
  public void printInfoComplete(){
    for (int i = 0; i < 10; i++){
      print (i);

      int count = 0;
      while (i != 0){
        count++;
        i--; }

      print(count); }
  }
}
```

```
public class Printer{
  public void printInfo(){
    int i = 5;
    print (i);

    int count = getCountOf(i);

    print (count);
  }
  public void printInfoComplete(){
    for (int i = 0; i < 10; i++){
      print (i);

      int count = getCountOf(i);

      print (count); }
  }
  private int getCountOf(int index){
    int i = index;
    int count = 0;
    while (i != 0){
      count++;
      i--; }
    return count;
  }
}
```

Fig. 1. The original class Printer (left hand side) has two methods sharing code blocks. In the refactored class shown on the right hand side, this block is extracted to the new method getCountOf

"Extract method"[1] for Java which extracts a statement block into a new method. This is particularly useful, if this statement block appears frequently in the software artifact. We use the class Printer (see Fig. 1) taken from [11] to illustrate the code extraction refactoring. This class is a suitable candidate for this refactoring since the methods printInfo() and printInfoComplete() use identical blocks of statements: the four lines framed lines in the code appear in both methods.

Using refactoring, we can (1) *extract* the common code from the two methods, (2) place it into a *newly defined method*, and (3) *replace* the code blocks with a call to the new method. The result of applying this refactoring to the class Printer is depicted on the right hand side in Fig. 1. The transformation does not change the behaviour of the class Printer, however in the following we will illustrate that a slight change to the refactoring of program Printer can cause a security leak while the Extract method refactoring is still valid.

Security leaks usually concern confidentiality, i.e., the loss of secret information to unauthorized third parties. A simple change to the refactoring illustrates how easily a information leak can be introduced: if we extract the common code block into a **public** method rather than a **private** one, also an attacker can now use the public method getCountOf to get the value of count. Nevertheless, this is a valid refactoring since the behaviour of the class badPrinter has the same external behaviour as the original class Printer. This is exactly one of the problems with definitions of behaviour that are based on traditional ideas of correctness. Correctness talks about fulfilling specified behaviour descriptions in a positive sense: what the program must do. However, security goes beyond cor-

[1] "Extract method" has been coined the *refactoring rubicon* [4].

rectness; it necessitates to specify what a program *must not* do, i.e., only what is specified and *nothing more*.

Another decisive reason why this example is not such an obvious information leak is that the leak is not a direct information flow caused by an assignment from secret to public variables. If we consider the secret information being contained in the variables called i of the methods printInfo and printInfoComplete, the class variable count is just an auxiliary variable that is never directly assigned to any of the variables i. Nevertheless, it may contain secret information from i because in the while loop the content of i is *copied* into the publicly accessible variable count. This represents a so called *implicit* information flow. In general, such implicit flows are hard to detect, since they transport information not by direct assignment or parameter passing but the information is *disguised as control flow* [1]. In the following section, we will see how our approach of applying IFC with Jif can help here.

2.2 Secure Refactoring with Jif

In our running example, one could argue that the extracted method must just be made **private** to avoid security problems. The idea of DLM and Jif is not far from that, but it adds the possibility of specifying the access control on a finer scale. An important aspect when considering security of Java is that it supports downloading code from a remote site, creating the risk that the downloaded code transports confidential data to the remote site. Java offers the sandboxing security concept to prevent such issues. However, this corresponds to leaving everything as private – outside code cannot access anything. For efficient distributed code, we need the possibility to specify more liberally and yet precisely restricted access control for systems of different parties. In the context of distributed security, the parties are called *principals*. DLM provides just such a model for specification of security policies tailored to distributed principals.

To enable specification of security, the DLM uses two sets of principals: *owner* and *reader*. Put together in a pair, they are called *labels* and represent security classes that can be attached to expressions in program code. The labels are ordered by a partial order relation $\sqsubseteq$, a special kind of lattice order relation. For example, the label {Bob $->$ Bob, Preparer} attached to an expression x in a program means that this expression is "owned" by Bob and that Bob and Preparer can read it. Reading literally means "read access" while owning means that the owner can control the data; this allows declassification (supported additionally by the *acts-for* relation on principals). The lattice operations *join* ($\sqcup$) and *meet* ($\sqcap$) allow combining labels thus supporting inference of labels for compound expressions:

$$\begin{aligned} owners(L_1 \sqcup L_2) &= owners(L_1) \cup owners(L_2) \\ readers(L_1 \sqcup L_2, O) &= readers(L_1, O) \cap readers(L_2, O) \end{aligned}$$

A DLM labeled Java program can be analysed for security. The idea is that the labels define the ways information is allowed to flow through the program. The security policy is that information *may only flow upwards*, i.e., from a position labeled L_1 to a position labeled L_2 iff $L_1 \sqsubseteq L_2$. Thus, to judge whether a program is secure, it suffices to check all possible flows and verify that they follow this flow policy. This is done by *relabeling*: values can change their label, when they become assigned to new variables either by an assignment or by passing them as parameters to methods.

Labels can be attached in Jif to usual Java programs much like type annotations. The Jif compiler statically checks whether these labels are secure, i.e., whether the annotated labels match the actual information flow of the program.

3 Extract Method Refactoring with Jif

In this section, we approach a general solution how to build Jif labels into the code so that refactoring with the Extract method is possible without changing the security. The argument for security is revealed gradually while we develop the solution step by step along a series of simple examples following the descriptive structure of the Re$\mathcal{L}$ [11] framework. Finally, we will wrap up the individual steps into a combined secure refactoring labeling for Extract method.

We use variables a, b, and c to illustrate also implicit information flows. We specify the Jif labels such that a, b, and c are owned by Bob, a and c can be read by Alice and Bob, while b can be read only by Bob.

```
class Test {
  private int {Bob -> Alice ,Bob} a = 0;
  private int {Bob -> Bob} b;
  private int {Bob -> Alice ,Bob} c; ...
```

The following examples contain implicit information flows where assignment of one variable is dependent via the control flow of the value of another variable. This necessitates consideration of the pc-label as the so-called “begin-label” of the extracted methods. Taking implicit flows into account allows us to generalize in the end since the non-dependent case is entailed in this more difficult case.

3.1 Assignments

As a first application example we consider that the code block that has been extracted in the refactoring into a method setB is a single assignment.

```
  public void f {} () { if (a == 0) { setB(); }}
  public void setB {Bob -> Bob} () { b = 4; }
```

When putting the assignment into the body of the newly extracted method, we need to specify a begin-label for this method. This label must be an upper bound for all pc-labels on which the method can be called. The label {Bob $->$ Bob} is allowed, because it is stronger (greater with respect to $\sqsubseteq$) than {Bob $->$ Alice,Bob} which is caused by the condition of the if-statement above. By contrast, the upper bound {Bob $->$ _} (where “_” stands for “all principals”) for the pc-label is not allowed because {Bob $->$ _} is weaker ($\sqsubseteq$) than {Bob $->$

Alice,Bob} (since there are more principals besides Alice and Bob). Consequently if we use this labeling for the extracted method setB, we obtain the following warning from the Jif compiler: *PC at call site more restrictive than begin-label of* setB().

An important question is, how we can calculate the begin-label for this refactoring in a constructive way without any help from the user of our method. There are two options. First, we can evaluate the context of the method, which means to consider all pc-labels, where the method is used. In this example we would have obtained the begin-label {Bob − > Alice,Bob}. The second option is to evaluate the body of the method, what we did. In this case we obtain the stronger label {Bob − > Bob}, which is derived from the left side of the assignment. The advantage of this second option in comparison to the first one is that we obtain the strongest begin-label that is possible at all. This increases the reusability of the method, because the begin-label represents the upper bound of all pc-labels on which the method can be called. Another advantage is that we can divide the refactoring Extract-Method in two sub-refactorings: Create-New-Method and Replace-Code-With-Method-Call. For the first subrefactoring we have usually not much knowledge of the context of the method. Hence, it is an advantage to calculate the begin-label just based on information inside the method. Note, in the rest of the paper the begin-label will be calculated from the method body only.

In contrast to the previous program, the extracted method setBC in following example includes two assignments.

```
public void f {} () { if (a == 0) { setBC(); }}
public void setBC {Bob -> Bob meet Bob -> Alice,Bob}(){b = 4; c = 3;}
```

The upper bound for the pc-label is built by the disjunction of the b-label {Bob − > Bob} and the c-label {Bob − > Alice,Bob}, using the meet operator. Further the rule is generalizable in order to be applicable for an arbitrary number of assignments.

3.2 Delegated (Iterated) Extract Method Refactoring

In contrast to the programs of Sect. 3.1, the extracted method setBC of the following example does not include the extracted assignment block directly. Instead, the method setBC invokes two other new methods, the methods setB and setC, delegating the assignments of the variables b and c.

```
public void f {} () { if (a == 0) { setBC(); }}
public void setB {Bob -> Bob} () { b = 4; }
public void setC {Bob -> Alice, Bob} () { c = 3; }
```

The upper bound for the pc-label of the method setB is given by the b-label {Bob − > Bob} and the upper bound for the pc-label of the method setC is built by the c-label {Bob − > Alice,Bob}. In the actual extracted method setBC, the upper bound for the pc-label is defined by the disjunction of the begin-label of method setB and the begin-label of method setC.

```
public void setBC {Bob -> Bob meet Bob -> Alice,Bob}(){setB(); setC();}
```

Again, the disjunction of two security policies can be built by the meet-operator and the calculation rule can be generalized to arbitrary method calls.

3.3 Return Label of Extracted Method

In this final example, we consider an implicit information flow caused by return-statements. If the method contains an if-statement, we need to propagate the pc-label to the calling context. In this example we do this by the *end-label* of the extracted method getB. In contrast to the begin-label of a method seen above, the specification of the end-label represents a lower bound for the context, where the method is used. Accordingly in the following example, the Jif-compiler has to check that the end-label of method getB is less restrictive than the labels of b and c. In other words, the end-label of getB defines a lower bound for the labels of b and c. First, the end-label for the pc-label is built based on the information outside the method. In this example, we construct it as the meet of the b-label {Bob − > Bob} and the c-label {Bob − > Alice,Bob}.

```
public void f {} () { b = getB(); c = getB(); }
public int getB(): {Bob -> Bob meet Bob -> Alice,Bob }{
    if (a == 0) return 4; else return 10; }
```

However, this is not the weakest label. Hence, we recommend to calculate the end-label from information inside the method based on the pc-labels of the return-statements.

```
public int getB (): {Bob -> Alice,Bob}{
    if (a == 0) return 4; else return 10; }
```

It may happen that the method-body contains more than one return-statement and these are labeled with different labels. In this case we have to choose the strongest one.

3.4 Wrapping It up

As a general method for constructing Jif labels for Extract method, we can summarize a general procedure from the examples presented above.

1. The entry label of the extracted method needs to be the upper bound of the variables assigned to in the extracted code block. This is constructed by the meet operator ($\sqcap$) of these variables' labels (see Sect. 3.1).
2. For an iterated Extract method refactoring, the label contruction of the higher level extracted method is derived also by iteration of the previous step, i.e., we assign the meet of the entry labels of the embedded methods (see Sect. 3.2).
3. The end-label of the extracted method should be constructed as the pc-label of the return statement within the extracted method (see Sect. 3.3). In case of several return statements the strongest one needs to be chosen (this is equivalent to the join $\sqcup$).

4 Results and Conclusions

The initial refactoring example can be made secure by applying the presented method. We first label the class Printer to provide a new version called goodPrinter where the variables have Jif labels. Further we can see that the extracted method has now a secure labeling following our method developed in Sect. 3.

```
class goodPrinter {
 public void printInfo{}() {
    int {Bob -> Alice,Bob} i = 5;
    print(i);
    int {Bob -> Bob}
          count = getCountOf(i);
    print(count); }

 public void printInfoComplete{}(){
  int {Bob -> Alice,Bob} i;
  int {Bob -> Bob} count;
  for (i = 0; i < 10; i++) {
    print(i);
    count = getCountOf(i);
    print(count);   }}


 public int {Bob -> Bob}
  getCountOf {Bob -> Alice,Bob}
   (int {Bob -> Alice,Bob} index):
                 {Bob -> Alice,Bob}{
    int {Bob -> Alice,Bob} i = index;
    int {Bob -> Bob} count = 0;
    while (i != 0) {
      count++;
      i = i--;
    }
    return count; }
```

In this paper, we have considered a method to provide security enhanced refactoring for Java programs. We believe that we are the first to address the problem of preserving security properties in the process of code refactoring. There is a plethora of works addressing security properties of programs and some consider structural aspects. But these works usually focus on composition of programs and the security properties of the composition in relation to that of the components, so-called compositionality of security, e.g., [6]. Our approach takes a completely different angle since we address refactoring, i.e. code changes that leave the program the same.

References

1. Boudol, G., Castellani, I.: Noninterference for concurrent programs. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, p. 382. Springer, Heidelberg (2001)
2. Chothia, T., Kawamoto, Y., Novakovic, C.: LeakWatch: estimating information leakage from Java programs. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 219–236. Springer, Heidelberg (2014)

3. Denning, D.E., Denning, P.J.: Certification of programs for secure information flow. Commun. ACM **20**(7), 504–513 (1977)
4. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison Wesley, Reading (2004)
5. Helke, S.: Jif examples. http://www.informatik.tu-cottbus.de/~helke/jif/
6. Mantel, H.: On the composition of secure systems. Security and Privacy (2002)
7. McComb, T.: Refactoring object-Z specifications. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 69–83. Springer, Heidelberg (2004)
8. Mens, T., Eeetvelde, N.V., Demeyer, S., Janssens, D.: Formalising refactorings with graph transformations. J. Softw. Maintenance **17**(4), 247–276 (2005)
9. Mens, T., Tourvé, T.: A survey of software refactoring. IEEE Trans. Softw. Eng. **30**(2), 126–139 (2004)
10. Myers, A.C., Liskov, B.: A decentralized model for information flow control. In: ACM symposium on Operating Systems Principles, SOSP. ACM (1997)
11. Ruhroth, T., Wehrheim, H., Ziegert, S.: Rel: A generic refactoring language for specification and execution. In: EUROMICRO. IEEE (2011)

You Never Surf Alone. Ubiquitous Tracking of Users' Browsing Habits

Silvia Puglisi(✉), David Rebollo-Monedero, and Jordi Forné

Department of Telematics Engineering, Universitat Politècnica de Catalunya (UPC), C. Jordi Girona 1-3, 08034 Barcelona, Spain
silvia.puglisi@upc.edu, {david.rebollo,jforne}@entel.upc.edu
http://www.upc.edu

Abstract. In the early age of the internet users enjoyed a large level of anonymity. At the time web pages were just hypertext documents; almost no personalisation of the user experience was offered. The Web today has evolved as a world wide distributed system following specific architectural paradigms. On the web now, an enormous quantity of user generated data is shared and consumed by a network of applications and services, reasoning upon users expressed preferences and their social and physical connections. Advertising networks follow users' browsing habits while they surf the web, continuously collecting their traces and surfing patterns. We analyse how users tracking happens on the web by measuring their online footprint and estimating how quickly advertising networks are able to profile users by their browsing habits.

Keywords: Privacy · Ubiquitous-tracking · Privacy metrics

1 Introduction

When users surf the web an intricate network of *personalisation services* tracks their preferences by *following* their browsing habits. These data is used to provide tailored suggestions, in terms of products users could buy, resources they might find interesting, social connections they might be interest to form. Personalisation services sometimes rely on different techniques to track users across different websites and applications. Many of these techniques use cookies. For example, Google Analytics service [7] uses cookies to measure user-interactions on websites. Another set of these techniques uses web or app sessions left open by the user. As an example someone might decide to check their web email account and then continue to surf the web without signing out, therefore leaving their session open. Yet another set of these techniques uses personalised features of the user's device or browser to restrict the pool of possible candidates among their visitors. Features that might be used by advertising networks include personalised language or fonts settings, browser extensions and so on. By identifying user through their accounts or unique features, analytics technologies can distinguish unique users across multiple devices or sessions.

J. Garcia-Alfaro et al. (Eds.): DPM and QASA 2015, LNCS 9481, pp. 273–280, 2016.
DOI: 10.1007/978-3-319-29883-2_20

1.1 Contribution

We have observed how users are tracked across the web and how the displayed advertising is tailored even after they have visited a few websites with a certain interest bias. In our study we analyse how quickly advertising networks can identify a user and start tracking them by measuring the distance between the measured user profile and the advertising profile. We introduce a set of metrics to express this distance and measure the number of web sites visited after which the distance between the advertising profile and the user profile is less then a certain threshold.

It is important to know that we have consider the case for which users are not registering, neither connecting any external account, as it could be the case with services like: Facebook, Google+, Twitter, and so on. We shall also point out that we have concentrated our study onto a single advertising network: Google. We reserve to future studies the possibility to include and analyse also other networks.

The main contributions of this paper are the following.

1. Introducing a model of the user online footprint.
2. Measuring how quickly a user is uniquely identified and tracked by an advertising network.
3. Introducing a measure of similarity between the user profile and the observed advertising profile.

2 Background

Information regarding locations, browsing habits, communication records, health information, financial information, and general preferences regarding user online and offline activities are shared by different parties online. This level of access is often directly granted from the user of such services. In a wide number of occasion though, private information are captured by online services without the direct user consent or even knowledge. We believe that the privacy and sensitiveness of the information becoming accessible to third parties can be easily overlooked.

Personal computers and more generally communication devices that are carried around by people are capable of being located, identified and tracked across different locations, networks and services [8]. All these devices can therefore be used for a variety of surveillance activities, which are in itself detrimental to the user's interests. Until recently in fact, the cost of surveillance and tracking of people and activities was proportional to the cost of directly reaching, asking or following a single person or a group of people. Technology therefore enhances the surveillance capabilities by introducing tools that allow the collection of information arising from a person's activities. This information can furthermore be combined and inferred, therefore offering a more complete picture of that person.

For example, to personalise their services or offer tailored advertising, web applications could use tracking services that identify a user through different networks [5,14]. These tracking services usually combine information from different profiles that users create, for example their Gmail address or their Facebook or LinkedIn accounts. In addition specific characteristics of the user's device can be used to identify them through different sessions and websites, as described by the Panopticlick project [4].

Browser fingerprinting is a technique implemented by analytics services and tracking technologies to identify uniquely a user while they browser different websites. Different features of a specific browser setup can be used to identify uniquely a user. Supported languages, browser extensions or installed fonts [2] can be used to identify a browser setup among others. More advanced techniques distinguish between browsers' JavaScript execution characteristics [9]. These features are particularly interesting since they are more difficult to simulate or mitigate in practice. Targeting JavaScript execution characteristics actually means looking at the innate performance signature of each browser's JavaScript engine, allowing the detection of browser version, operating system and microarchitecture. These attacks can also work in situations where traditional forms of system identification (such as the user-agent header) are modified or hidden. Other techniques exploit the whitelist mechanism of the popular NoScript Firefox extension.This mechanism allow the user to selectively enabling web pages' scripting privileges to increase privacy by allowing a site to determine if particular domains exist in a user's NoScript whitelist.

It is important to note that while tracking creates serious privacy concerns for internet users, the customisation of results is also beneficial to the end user [3]. In fact, while tailored services offer to the user only information relevant to their interests, it also allows some companies and institutions to concentrate an enormous amount of information about internet users in general. [12] investigate user profiling and access mechanisms offered by online data aggregator to users' collected data. Both the collected data and its accuracy was analysed together with the user's concerns. In their findings about 70 % of the participants to the study expressed some concerns about the collection of sensitive data, its level of detail and how it might be used by third parties, especially for credit and health information.

It has been shown how most successful tracking networks exhibit a consistent structure across markets, with a dominant connected component that, on average, includes 92.8 % of network vertices and 99.8 % of the connecting edges [6]. [6] have measured the chance that a user will become tracked by all top 10 trackers in approximately 30 clicks on search results to be of 99.5 %. More interesting, [6] have shown how tracking networks present properties of the small world networks. Therefore implying a high-level global and local efficiency in spreading the user information and delivering targeted ads.

3 Modelling the User's Footprint

We model the user's activity as series of events belonging to a certain identity. Each event is a document containing different information. We can formally defined this as a hypermedia document i.e. an object possibly containing graphics, audio, video, plain text and hyperlinks. We call the hyperlinks selectors and we use these to build the connections between the user's different identities or events. Each identity is a profile that the user has created onto a service or platform. This can be an application account or a social network account, such as their LinkedIn or Facebook unique IDs. An event is an action performed by the user, like visiting a website or creating a post on a blog.

We aggregate keywords each time the user creates a new event by visiting a different url. These keywords constitute the user profile of interests (Fig. 1). A tractable model of the user profile as a probability mass function (PMF) is proposed in [10,11] to express how each keyword contributes to expose how many times the user has indirectly expressed a preference toward a specific category. We consider that the user expresses a preference when they visit a webpage categorised with a certain keywords. This model follows the intuitive assumption that a particular category is weighted according to the number of times this has been counted in the user profile.

We define the profile of a user u_m as the PMF $p_m = (p_{m,1}, \dots, p_{m,L})$, conceptually a histogram of relative frequencies of tags across the set of tag categories $\mathcal{T}$.

Similarly, we define the profile of an ads i_n as the PMF $q_n = (q_{n,1}, \dots, q_{n,L})$, where $q_{n,l}$ is the percentage of tags belonging to the category l which have been assigned to this specific advertising item. Both user and ads profiles can then be seen as normalised histograms of tags across categories of interest. Our profile model is in this extent equivalent to the tag clouds that numerous collaborative tagging services use to visualise which tags are being posted, collaboratively or individually by each user. A tag cloud, similarly to a histogram, is a visual representation in which tags are weighted according to their relevance.

In view of the assumptions described in the previous section, our privacy attacker boils down to an entity that aims to profile users by representing their interests in the form of normalised histograms, on the basis of a given categorisation.

3.1 A Metric of Similarity

We consider the third party advertising network to operate like a recommendation system, that suggest products or services that might be of interest for the user, based on their preferences. A recommendation system can be described as an information filtering system that seeks to predict if the user is interested or not in a particular resource. We assume that the ad server suggest advertising based on a measure of *similarity*.

We measure the user profile, as previously described, as an histogram of their recorded preferences, and the advertising profile as an histogram of the ads that the user has received. We use the $1 - norm$ as a measurement of how the advertising network is tracking the user profile:

$$\|p_m, q_n\|_1 = \sum_i |p_{m_i} - q_{n_i}|$$

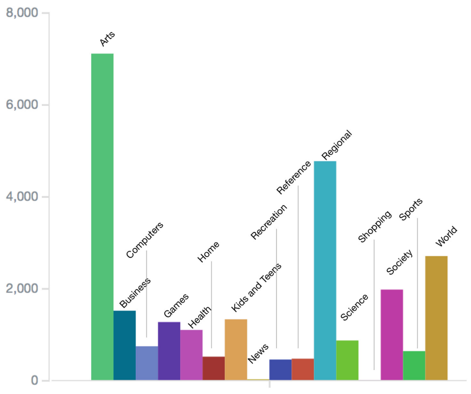


Fig. 1. Here we show an example of user profile expressed in absolute terms by counting the number of keywords in each category for a browsing session. We model user and advertising profiles as histograms of tags keywords a set of predefined categories of interest.

4 Experimental Methodology and Results

We analysed the browsing habits of 86 users of Twitter, by observing the set of websites they share in their feed. We assumed that the articles shared on twitter are a subset of the website that each users visit every day. Yet if they are active Twitter users, these websites will express their interest bias towards certain categories. Please note that we only consider the links shared on the platform as a sequence of website that the user might have visited. These sites are therefore surfed in our simulation environment. Here we pretend that a user is going through their reading list of sites and we measure how the advertising changes in the page and adapts to their profile. The user is simulated by a software agent opening the urls and surfing the page for a certain arbitrary amount of time.

In our simulated environment the users are not logged in Twitter or any other account.

For each users we analysed the websites and collected keywords for the shared articles. We used both the meta information contained in the page, as well as extracted keywords from the actual text of the page by using the Rapid Automatic Keyword Extraction (RAKE) [13] algorithm. Each keyword was evaluated against Open Directory Project (DMOZ) [1] for classification within top levels categories.

Once the user profile was calculated the advertising profile is evaluated. The advertising profile is extracted from url parameters contained in third party requests. We have considered only Google ads for the purpose of this study. These parameters are again evaluated against DMOZ for classification within top levels categories.

At each step the *linear norm* between the advertising profile and the actual user profile is evaluated.

By profiling users' browsing events using a hypermedia document structure we were able to show how each event contains a set of features regarding the user identity and the page that was visited. We have therefore categorised each event by using the keywords contained in the meta information present in the page and the page text itself (Fig. 1). We have observed how a large and sophisticated advertising network such as Google is able to profile users quickly and only in a few visits (Figs. 2 and 3).

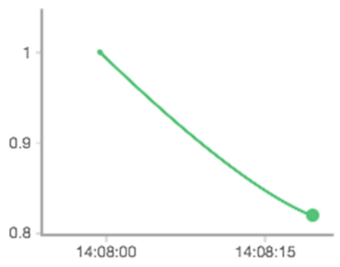


Fig. 2. The figure illustrates how the norm between the advertising and the browsing profile decrease of approximately 20 % in two subsequents visits and only in 15 s

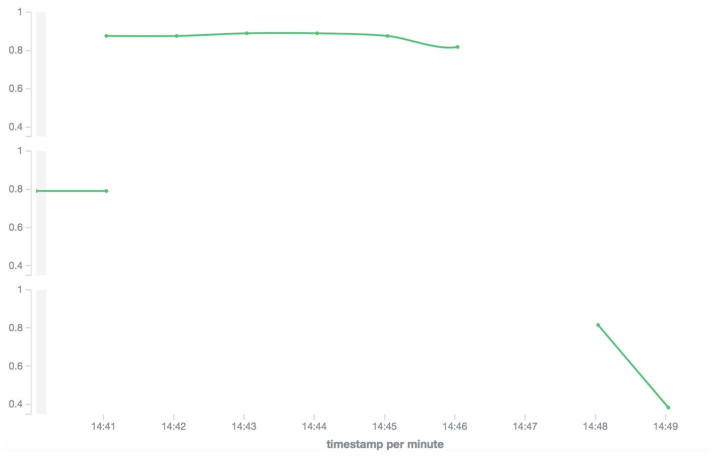


Fig. 3. The figure compare the 1-norm decrease for three different users in a short timespan. This shows how, when a user surf websites in a specific category the advertising slowly adapts to the new category and the norm decreases. When the category changes the advertising needs to adjust again.

We have found how our hypermedia model is particularly suited for big data analysis and how it allows a user to keep their online footprint under control by understanding precisely which websites have introduced certain keywords in their profile. Eventually this technique would allow the user to implement more precise Privacy Enhancing Techniques (PETs) to masquerade their profile to advertising networks.

5 Conclusions and Future Work

Using a hypermedia model to capture the footprint that users leave online while surfing the web has proven a promising technique. Particularly the model is able to capture both the single categorisation that each webpage introduce as well as time series analytics and breaking up of third party tracking per advertising network. We have also shown how web tracking by large advertising networks happens very quickly in a few subsequent visits to websites in the network (Figs. 2 and 3). In future research we would like to further explore the hypermedia model introduced, while continuing to understand how quickly web advertising is able to match the served ads with the actual user profile. This would allow us to understand if different profiles for the same users can be somehow linked together within similar advertising networks. We are also particularly interested in measuring how social networks sharing buttons and/or commenting services, included on websites, are able to track users even when these have not signed in with their account. We reserve the study of their capabilities to future investigations. More over we want to enlarge the set of users analysed by testing on logs from a real world small computer network, while also introducing new metrics to our study. In particular we are already planning to consider: 2-norm, KL-Divergence between the advertising profile and the observed user profile, Fisher information. We also believe in the importance to provide users with simple visualisation tools able to show the user their online footprint and allowing them to take action to masquerade their interests profile or simply block certain networks.

Acknowledgments. This work was partly supported by the Spanish Government through projects CONSEQUENCE (TEC2010-20572-C02-02) and EMRISCO (TEC2013-47665-C4-1-R).

References

1. Open directory project. http://www.dmoz.com
2. Boda, K., Földes, A.M., Gulyás, G.G., Imre, S.: User tracking on the web via cross-browser fingerprinting. In: Laud, P. (ed.) NordSec 2011. LNCS, vol. 7161, pp. 31–46. Springer, Heidelberg (2012)
3. Castelluccia, C.: Behavioural tracking on the internet: a technical perspective. In: Gutwirth, S., Leenes, R., De Hert, P., Poullet, Y. (eds.) European Data Protection: In Good Health?, pp. 21–33. Springer, The Netherlands (2012)

4. Eckersley, P.: Panopticlick (2011)
5. Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice and open challenges. Proc. VLDB Endowment **5**(12), 2018–2019 (2012)
6. Gomer, R., Mendes Rodrigues, E., Milic-Frayling, N., Schraefel, M.: Networkanalysis of third party tracking: user exposure to tracking cookies through search. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 1, pp. 549–556. IEEE (2013)
7. Inc., G.: Google Analytics cookie usage on websites (2015). https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage
8. Michael, K., Clarke, R.: Location and tracking of mobile devices: Überveillance stalks the streets. Comput. Law Secur. Rev. **29**(3), 216–228 (2013)
9. Mowery, K., Bogenreif, D., Yilek, S., Shacham, H.: Fingerprinting information in Javascript implementations. In: Proceedings of W2SP, vol. 2 (2011)
10. Parra-Arnau, J., Perego, A., Ferrari, E., Forné, J., Rebollo-Monedero, D.: Privacy-preserving enhanced collaborative tagging. IEEE Trans. Knowl. Data Eng. **26**(1), 180–193 (2014). http://dx.doi.org/10.1109/TKDE.2012.248
11. Parra-Arnau, J., Rebollo-Monedero, D., Forné, J., Muñoz, J.L., Esparza, O.: Optimal tag suppression for privacy protection in the semantic Web. Data. Knowl. Eng. **81**–**82**, 46–66 (2012). http://dx.doi.org/10.1016/j.datak.2012.07.004
12. Rao, A., Schaub, F., Sadeh, N.: What do they know about me? Contents and concerns of online behavioral profiles (2015)
13. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic keyword extraction from individual documents. In: Text Mining, pp. 1–20 (2010)
14. Veeningen, M., Piepoli, A., Zannone, N.: Are on-line personae really unlinkable? In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W.M. (eds.) DPM 2013 and SETOP 2013. LNCS, vol. 8247, pp. 369–379. Springer, Heidelberg (2014)

LockPic: Privacy Preserving Photo Sharing in Social Networks

Carlos Pares-Pulido and Isaac Agudo(✉)

Network, Information and Computer Security Lab.,
University of Malaga, Malaga, Spain
carlosparespulido@gmail.com, isaac@lcc.uma.es

Abstract. There are many privacy concerns related to the use of social networks, in particular the posting of pictures and controlling who has access to them. In this paper we introduce a solution for the distribution of personal or sensitive pictures. Our aim is to provide a method for secure and privacy friendly picture sharing through social networks, that allows users to encrypt sensitive regions in pictures (particularly, faces) in a reversible, non-intrusive way, leaving the rest of the picture unaltered. This way, any image can be freely published and distributed on any social network, and viewed by as many users as the platform allows, while the protected parts are only accessible with the corresponding key. Once the key for a particular region has been acquired, the receiver of the picture can decrypt this region without downloading any additional information. The core of our proposal is a C library, which efficiently integrates an encryption/decryption algorithm with the encoding/decoding process. We have also released an Android application, LockPic, and a companion key server that showcase all the functionality mentioned in this work.

Keywords: Partial image encryption · Privacy in social networks · Reversible image scrambling

1 Introduction

Nowadays social networking is booming; and with it, the public sharing of personal photographs. From a privacy point of view, as soon as a user publishes a picture, he or she loses all control over it. Even if that content is later removed from the servers, other users that have stored it can share it again indefinitely, even contrary to the wishes of the original owner. The situation worsens when the picture involves other people who may not wish the content to be shared (or even be aware of it) or when legal restrictions apply to the subjects, as is the case for children in most countries; or when some of the subjects have certain notoriety.

Traditional protection mechanisms do not fit well in this scenario. Instead of protecting the whole picture only critical parts of it should be protected or hidden. This will allow other members of the social network to preview the picture without compromising the privacy of the users in the picture. Usually, faces are the target for protection but other elements might need to be hidden too, e.g. number plates. The advantage of reversible hiding is that authorized users are able to recover the whole picture while non-authorized users only get the public parts.

There are three parameters that we need to take into account when proposing a solution:

- **Usability.** The hiding technique should be as unobtrusive as possible and should be lightweight enough to run smoothly in mobile platforms.
- **Interoperability.** It should be easy to integrate the hiding process with the photo sharing work flow.
- **Security.** It should be hard enough to recover the original picture without knowing the key material used for hiding the critical parts.

The centre of our proposal is a C library, based on OpenSSL and open-source JPEG codecs, which integrates cryptographically strong encryption (AES) in the encoding process of JPEG pictures. The LockPic app[1] is built upon this C library an is able to encrypt sensitive parts of a picture and decrypt them afterwards. We have also implemented a companion Key Server that is used to convey the cryptographic keys used to protect the pictures.

2 Related Work

There are many proposals for partial or selective encryption of pictures and videos in the literature [8]. If we focus on still pictures, most of them rely on the use of the JPEG2000 format [3], that provides a better basis for partial encryption but has a major drawback: it is mostly unsupported by current social networks and web browsers. In this section we focus on solutions based on JPEG [5].

In the following paragraph we briefly summarize how JPEG images are encoded, in order to explain how the encryption mechanism can be integrated. An image has between 1 and 3 channels: a luminance channel, and up to 2 chrominance channels. Each channel is broken down into square blocks, called MCU (Minimum Coded Units) upon which encoding is performed separately. Each MCU undergoes a discrete cosine transformation, and the 64 coefficients corresponding to the lowest frequencies (the ones which human eyes can distinguish best) are kept. These values are rounded, to maximize the number of zeros between them; this step, called quantization, introduces losses. Finally, this sequence of 64 numbers is compressed without loss, using Huffman encoding. This clearly points to an optimal place where to perform the encryption process as the set of 64 coefficients obtained after quantization suffer no further losses beyond this point.

In [2] authors present an approach that could be used for privacy preserving video surveillance. Later in [7] they adapted their approach to support JPEG

[1] https://www.nics.uma.es/lockpic.

and provide a prototype iOS application, Proshare. Their solution is based on flipping the sign of the coefficient in the encoded image, i.e. using any cryptographically secure pseudo random bit sequence generator, the sign of a coefficient is changed for a 1, or kept as a 0. Although this scheme works well in general, its application to high resolution images is limited because the scrambling becomes less noticeable for all proposed encryption levels, except for the ultra-high level that encrypts the DC components using a one time pad. Unfortunately, there is not enough information about what kind of PRNG is used for sign flipping and one time pad encryption in the ultra-high level in order to evaluate the real strength of this proposal. Another important aspect is that scrambling and descrambling on client application and automatic key distribution is not supported in the prototype at the time of writing.There are some other approaches [6,11] that also base their encryption algorithm on a pseudo-random shift on the JPEG coefficients, trying to keep enough information for the whole image to be "homogeneous" but at the same time trying to limit the chances to recognize the scrambled portion.

The main security risk of schemes based on flipping/shifting the signs of the JPEG coefficients based on some random patterns is that brute force attacks can reconstruct the pattern taking advantage of the characteristics of the image by looking at the neighbouring pixels of the scrambled area in the first phase and iterating from the edge to the centre. Furthermore, its visual output is fairly obtrusive, as seen in Fig. 1a.

To mitigate the visual impact, we can follow the idea described in [11]: always keeping the most significant coefficient (DC), and pseudorandomly shifting the remaining 63. The effects of this alteration make the output visually acceptable, as shown in Fig. 1b. However, this not only makes the algorithm less secure from a theoretical point of view (by exposing even more information), but also in practice. Encrypting only the signs of the AC coefficients has been proven insecure for video contents [4]. The DC, which holds colour information, remains unaltered. This means that the average colour in any encrypted MCU will be the same as prior to encryption. For a large enough image, an MCU basically behaves as a pixel; therefore making the encryption irrelevant to the naked eye (even if information is altered) as we mentioned before. This is shown in Fig. 1c. This is roughly the same effect that all encryption levels will suffer from, apart from the ultra-high in [7].

One of the key aspects of all these proposals is that the scrambled image contains all the information needed to recover the original image given the encryption key. From a practical perspective, this seemed to be a good starting point, since the algorithm for encryption and decryption is very simple and efficient. In [9], the authors present a solution that stores some encrypted information needed to recover the image in the Cloud. Their solution is based on a proxy that intercepts all the images downloaded from the social network and decrypts them on the fly, providing a transparent decryption service. Although the authors claim that their solution only adds minimal photo storage overhead in the Cloud, it is true that they add a new dependency. Also, the transparent proxy is harmless

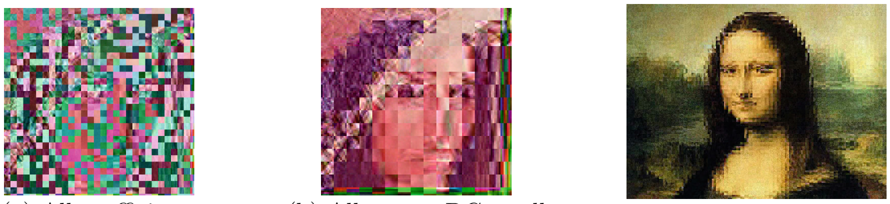

(a) All coefficients (b) All except DC small (c) All except DC medium

Fig. 1. Pseudorandomly shifted coefficient signs

in HTTP connections but can cause some security and trust issues when using HTTPS, which is the current standard in social networks.

In [10] they take a different approach, instead of encrypting some parts of the image they used a JPEG file as a container for an encrypted image. Their focus is to preserve recoverability of the container JPEG in common social networks, i.e. resistant to JPEG re-compression. Their solution is based on Javascript, providing browser side decryption of the image without requiring any server to store encrypted parts of the image but fails to protect only parts of the images, their proposal is encrypt all or nothing. As the container JPEG is not representative of the shared picture this scheme is somewhat equivalent to just sharing a link to the protected picture.

Another issue that is usually disregarded in most proposals is the usability of the application. Some applications assume there is a given box in the picture to be protected but how the coordinates are defined and how the protection is activated is not mentioned. In [1] the authors propose the use of QR codes as wearable Tags to activate the protection mechanism. In their approach, users who wear a *Privacy.Tag*, i.e. a printed QR code, are recognized when taking a picture and their face is automatically protected based on their preferences.

3 The Encryption/decryption Module

Our proposal is based on an C library that performs JPEG encoding/decoding at the same time as encryption/decryption. This core C library requires symmetric keys for the encoding and decoding of JPEG pictures.

We plan the following requirements for the encryption/decryption process: It has to be **reversible**; the output of the encryption must still be a **valid image** in a standard format; and the process should be **cryptographically secure**. We aim to distort, beyond recognition, any encrypted region in the picture, leaving the rest fundamentally unaltered. The encryption/decryption procedure should be lightweight, and produce encrypted files of a manageable size. Furthermore, ideally, obfuscated regions in the encrypted image should be visually unobtrusive with respect to the rest of the picture; i.e., colours and the rough outline of the picture should remain similar to the original. This latter requirement, in a way, conflicts with the desired distortion. The higher the distortion level gets, the more obtrusive the output becomes. We have tried to find a compromise in the distortion level, favouring security over aesthetics.

Our first approach was to use an encryption mechanism based on secure permutations of pixels in the regions to be obfuscated; but that meant relying on the BMP format, which posed several problems - principally speed and output size. In Fig. 2a we can see that the randomization of pixels preserve the colours of the original picture but not the shape of the underlying face. This can be partially solved by decomposing area of interest in small tiles, applying the pseudo randomization only in the tiles. In Fig. 2b we can see that the encryption is less intrusive in this case. As mentioned before the main problem of working at the bitmap level is the size of the pictures. If we later convert the resulting images into a compressed format we have some issues with colour distortion, Fig. 2c that can not be easily solved. After the evaluation of the pros and cons, we took a completely different approach to encryption, instead of considering the image as a matrix of pixels, we directly targeted JPEG-encoded images, and worked on the integration of the encryption process inside the JPEG codec.

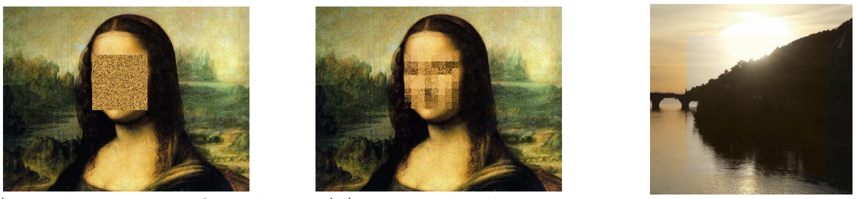

(a) Randomization of pixels (b) Using tessellation (c) JPEG decoding halo

Fig. 2. Bitmap level encryption

In our proposal, the sequence of all non-zero coefficients in each MCU is encrypted using a standard symmetric cipher, specifically we use AES in OFB mode, resulting in a new set of coefficients. It is important to note that this scheme does not in any way respect the original distribution of colours. However, irrespective of the size of the image, the encrypted regions will remain fully obfuscated. In Appendix B there is a sample encryption output.

We decided to encrypt only the non-zero coefficients in order to preserve the effectiveness of the compression algorithm. Huffman encoding takes advantage of long runs of zeros, but those patterns are removed if we encrypt all coefficients. In practice, encrypting all coefficients resulted in JPEG images which, even if smaller than their BMP counterparts, were still too big to be conveniently manageable. We admit that this weakens the scheme (as opposed to also encrypting zeros) because we are openly revealing which coefficients are null and which are not. Still, this information is largely insufficient to reconstruct an image, and in exchange, it allows the encrypted images to be the size of a regular JPEG file.

This scheme fulfils most of the requirements we initially planned for the encryption procedure: it is indeed reversible, completely obfuscates the encrypted regions, and does not affect any other region in the image. The size of the encrypted files is almost the same as the original file and the encryption/decryption process is fast and lightweight. There are also some negative aspects to our proposal. Firstly, it implies some degree of quality loss (unavoidable due to the conversion to JPEG, which is always lossy). Secondly, the encrypted areas are very obtrusive in the picture. It should also be noted that this implementation is not tolerant to further compression of the image.

The fact that some social networks recompress pictures in order to optimise storage and network bandwidth may force us to use some external services for storing protected pictures (e.g. Flickr, ImageShack, Dropbox, etc.), sharing only the link to the picture instead of the picture itself. This way the social network will only store the preview of the picture, whereas the original picture can always be retrieved from the external service. However there are social networks that do not recompress the pictures and keep the original metadata, (e.g. Google+) in which we do not need an external service.

4 LockPic Elements and Security Model

We assume pictures are stored by their owner in a service of their choice. It can be a social network or a photo sharing service, but the user can also share pictures using email or other means. We assume this server will not alter the pictures but it might be interested in learning the protected contents.

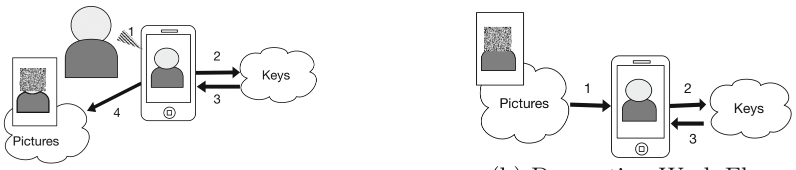
The key server is queried by the LockPic application using a secure channel, whenever a picture needs to be decrypted or encrypted. The key server does not know anything about the contents of the picture, only some metadata and the encryption keys. This way, pictures and the keys needed to encrypt/decrypt them, are stored in different trust domains.

The key server and the photo sharing service are considered honest-but-curious servers. We assume they will not collude to try to compromise the user's privacy. The Key Server will only provide keys to authorized users and will generate keys and IDs for pictures as instructed. The photo sharing service is trusted not to alter pictures but there are no additional requirements as to whom is granted access to the encrypted pictures. Other users can interact with both services using the standard APIs and are not supposed to be trusted. Users can collude among themselves and can have access to all pictures stored in the photo sharing service but can only get keys according to the access granted in the Key Server.

The encryption work flow (Fig. 3a) consists of four steps. It usually starts when a picture is taken using the LockPic application but users can also select a picture already stored in their phone. The application will then show the picture to the users and prompt them to select which areas need protection and who is going to be able to access them. After that, the Key Server is queried by the application using a secure channel, in order to get the encryption key for each area of interest. The application sends the name and the dimensions of the picture as well as the list of coordinates for each of the boxes that need to be encrypted together with the list of users that are granted access to them. The Key Server generates a random ID for the picture and stores the information about the protected boxes in the database. Then, the server sends back the ID and the list of encryption keys corresponding to each of the boxes requested, one key per box. We explain the details of the key generation in Appendix A.

The application passes the encryption keys, coordinates and ID to our modified JPEG encoding library that will encrypt the boxes using the corresponding keys and include the ID in the system of the picture. The ID needs to be present in the metadata of the picture in order to allow other users to query the decryption keys. Finally, the application shows the encrypted image and offers the user some sharing choices.

As shown in Fig. 3b the decryption work flow consists of three main steps and is triggered each time the user receives an encrypted picture. First, the picture is loaded by the LockPic application. The ID of the picture is extracted from the metadata. Second, the application authenticates the user against the Key Server and submits the picture ID in order to get the keys for the boxes they have access to. The Key Server gets from the data base the list boxes present in the picture based on the picture ID. For each box it checks whether the user has been granted access or not. Then, it returns to the application the list of boxes that the user has been granted access to, together with the corresponding decryption keys. Finally, the application passes on the list of boxes and keys to the modified JPEG decoding library.



(a) Encryption Work Flow (b) Decryption Work Flow

Fig. 3. LockPic Encryption and Decryption Work Flows

Access rights can be modified by the owner of the picture at any time. Users can request from the Key Server the list of picture IDs that has been generated for them, together with the coordinates of the encrypted boxes and the users that are currently granted access. They can add new users or delete existing ones from current boxes, however they cannot create new boxes. With the current API, the application needs to request a new picture ID when the users want to modify or delete existing boxes, or create new ones.

The Key Server provides the encryption/decryption keys to the users based on the picture ID and the user ID, hence the first step is to be able to authenticate users against the Key Server. We have used Google Accounts for authentication in order to take full advantage of the Google ecosystems. This way, authentication of users is transparently managed by the Android operating system and the Google App Engine where we have deployed an instance of the Key Server.

5 Conclusions and Future Work

The LockPic application has been implemented as a prototype to showcase the functionality of the C library for simultaneous JPEG encryption/decryption and encoding/decoding. We have released the source code of the application and the key server in order to demonstrate the feasibility of our approach and help other built upon our work.

As for future work, we would like to explore other architectures, authentication flows, and key distribution methods that can work without the need of an on-line key server and include our functionality in messaging applications (e.g. Telegram). We can use an hybrid approach where the symmetric keys used to encrypt each region are encrypted with the public key of the intended recipients. Challenges in this scenario will be to optimize the size of the metadata needed to encode the cryptographic keys. We would also like to broaden the reach of our work - release the encryption module as a standalone C library, and release the application for other operating systems. Another important issue that is not tackled in the present paper is how to keep track of who is accessing your pictures. We think our approach can be easily adapted to provide a better view of who has accessed the pictures based on the release of decryption keys. However, this will require modifications both on the client - and server - side.

Acknowledgements. The research leading to these results has received funding from the Junta de Andalucía through the projects FISICCO (P11-TIC-07223) and PISCIS (P10-TIC-06334). We also thanks the anonymous reviewers for their valuable comments.

A Managing Encryption/decryption Keys

Apart from providing a proper security level and an efficient implementation, one relevant challenge is to properly manage all the encryption keys used in the system. We propose a centralised approach where all keys are stored in the trusted Key Server.

It is essential that the server is able to uniquely identify images in order to be able to generate unique keys for each picture and region in it. As we have mentioned, the Key Server randomly generates a unique identifier for each protected picture that is sent back to the LockPic application at encryption time. This unique ID is included in the metadata of the encrypted picture. Another approach could be to use the hash of the picture as ID. The problem of using the hash as the ID is that the hash has to be performed in the mobile application, which might be an expensive operation depending on the size of the picture, and could present security problems in the case that hash collisions are found. More importantly, the key server would be able to analyse some usage patterns as it would be able to recognize if two different users encrypt the same picture.

As mentioned, the key generation process is performed on the server side. Our initial approach was to generate a separate key for each protected region in every image. This, however, posed some problems because, due to the speed at which random numbers may be needed, the Random Number Generator (RNG) might act as a bottleneck. It would also be difficult to estimate the size of the key store as it would grow in proportion to the number of regions protected. Since having different keys for different regions is mandatory in order to allow for fine grain access control to regions, we have taken the following approach.

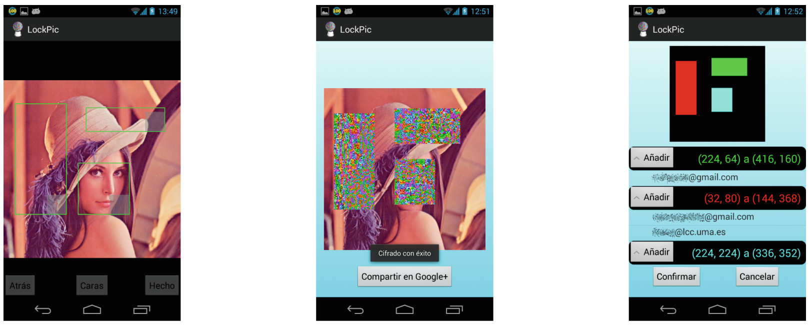
For each user, U, a master secret, MS_U is randomly generated at the first access. For every region to encrypt, this secret is concatenated with the picture identifier, ID, and the coordinates of the region, $r = \{x_0, y_0, x_1, y_1\}$; a secure hash function is subsequently applied on this string of bits, and its output is used as the encryption key for the region, i.e.

$$key_{U,ID,r} = hash(MS_U \parallel ID \parallel x_0 \parallel y_0 \parallel x_1 \parallel y_1)$$

The main advantage of this design is that it only uses the RNG once per user and that the number of keys managed by the Key Servers is linear on the number of users, thus independent from the number of pictures or encrypted boxes.

B The LockPic App

The LockPic App uses a very simple user interface with three different choices: Encrypt, Decrypt and My Pictures. The first choice triggers the encryption mechanisms, users are prompted to choose a picture from the gallery and are required to select which regions need to be protected. The selection of protected (Fig. 4a) areas can be performed manually, by placing a box over the desired regions and scaling it by dragging the lower-right corner. Another option is to rely on Android face detection APIs in order to get boxes over the detected faces. In any case, boxes can be easily rearranged and scaled with one finger movement.



(a) Regions selection (b) Encrypted result (c) My Pictures

Fig. 4. LockPic user interface

Once the regions have been selected, the user is prompted to select which contacts are authorized to decrypt each of the regions. This step can be skipped and new permissions can be set up later on. Then, the encrypted image (Fig. 4b) that will be stored in the LockPic folder is shown.

Decryption is performed by checking the picture ID included in the metadata and requesting from the key server the corresponding decryption keys. The decrypted image is shown to the user but never stored in the file system. LockPict also provides users with the opportunity to review their access control policies (Fig. 4c). It retrieves from the key server all picture IDs created by the user together with their associated encrypted regions and the list of authorized users and gives the user the choice to modify (add or remove) the users allowed to view each of the regions.

References

1. Bo, C., Shen, G., Liu, J., Li, X.-Y., Zhang, Y., Zhao, F.: Privacy.tag: privacy concern expressed and respected. In: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, SenSys 2014, pp. 163–176 (2014)
2. Dufaux, F., Ouaret, M., Abdeljaoued, Y., Navarro, A., Vergnenègre, F., Ebrahimi, T.: Privacy enabling technology for video surveillance. In: Defense and Security Symposium, International Society for Optics and Photonics (2006)
3. Engel, D., Sttz, T., Uhl, A.: A survey on JPEG2000 encryption. Multimedia Syst. **15**(4), 243–270 (2009)
4. Hofbauer, H., Unterweger, A., Uhl, A.: Encrypting only AC coefficient signs considered harmful. In: IEEE International Conference on Image Processing (2015)
5. ITU. Iso/iec 10918–1: (e) ccit recommendation t.81 (1993)
6. Khan, M.I., Jeoti, V., Khan, M.A.: Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In: International Conference on Intelligent and Advanced Systems (ICIAS) (2010)
7. Korshunov, P., Ebrahimi, T.: Scrambling-based tool for secure protection of JPEG images. In: IEEE International Conference on Image Processing (ICIP) (2014)
8. Massoudi, A., Lefebvre, F., De Vleeschouwer, C., Macq, B., Quisquater, J.-J.: Overview on selective encryption of image and video: challenges and perspectives. EURASIP J. Inf. Secur. **2008**(1), 179290 (2008)
9. Ra, M.-R., Govindan, R., Ortega, A.: P3: toward privacy-preserving photo sharing. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, NSDI 2013 (2013)
10. Tierney, M., Spiro, I., Bregler, C., Subramanian, L.: Cryptagram: photo privacy for online social media. In Proceedings of the First ACM Conference on Online Social Networks, COSN 2013 (2013)
11. Van Droogenbroeck, M., Benedett, R.: Techniques for a selective encryption of uncompressed and compressed images. In: Advanced Concepts for Intelligent Vision Systems (ACIVS) (2002)

Author Index