Willem Jonker
Milan Petković (Eds.)

LNCS 6358

Secure Data Management

7th VLDB Workshop, SDM 2010
Singapore, September 2010
Proceedings

Springer

Lecture Notes in Computer Science 6358

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Willem Jonker   Milan Petković (Eds.)

Secure Data Management

7th VLDB Workshop, SDM 2010
Singapore, September 17, 2010
Proceedings

Springer

Volume Editors

Willem Jonker
Philips Research Europe
High Tech Campus 34, 5656 AE Eindhoven, The Netherlands
E-mail: willem.jonker@philips.com
and
University of Twente
Department of Computer Science
Enschede, The Netherlands
E-mail: jonker@cs.utwente.nl

Milan Petković
Philips Research Europe
High Tech Campus 34, 5656 AE Eindhoven, The Netherlands
E-mail: Milan.Petkovic@philips.com
and
Eindhoven University of Technology
Faculty of Mathematics and Computer Science (HG 9.85)
Eindhoven, The Netherlands
E-mail: m.petkovic@tue.nl

Library of Congress Control Number: 2010933962

CR Subject Classification (1998): K.6.5, D.4.6, E.3, C.2, H.4, H.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN	0302-9743
ISBN-10	3-642-15545-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-15545-1 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2010

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The VLDB Secure Data Management Workshop was held for the 7th time this year.

The topic of data security remains an important area of research especially due to the growing proliferation of data in open environments as a result of emerging data services such as cloud computing, location based services, and health-related services. Confidentiality is the main driving force behind the research that covers topics such as privacy enhancing technologies, access control, and search in encrypted data.

We received 20 submissions from which the program committee selected 10 papers to be presented at the workshop and included in the proceedings (50% acceptance rate). In addition, we are proud that Elisa Bertino accepted our invitation to give a keynote for which she selected the topic of data trustworthiness. We hope the papers collected in this volume will stimulate your research in this area.

The regular papers in the proceeding have been grouped into two sections. The first section focuses on privacy. The papers in this section present a balanced mix of theoretical work on anonymity and application-oriented work.

The second section focuses on data security in open environments. The papers address issues related to the management of confidential data that is stored in or released to open environments, such as, for example, in cloud computing.

We wish to thank all the authors of submitted papers for their high-quality submissions. We would also like to thank the program committee members as well as additional referees for doing an excellent review job. Finally, let us acknowledge the work of Luan Ibraimi, who helped in the technical preparation of the proceedings.

July 2010

Willem Jonker
Milan Petković

Organization

Workshop Organizers

Willem Jonker	Philips Research/University of Twente, The Netherlands
Milan Petković	Philips Research/Eindhoven University of Technology, The Netherlands

Program Committee

Gerrit Bleumer	Francotyp-Postalia, Germany
Ljiljana Brankovic	University of Newcastle, Australia
Sabrina De Capitani di Vimercati	University of Milan, Italy
Ernesto Damiani	University of Milan, Italy
Eric Diehl	Techicolor, France
Lee Dong Hoon	Korea University, South Korea
Jeroen Doumen	Irdeto, The Netherlands
Csilla Farkas	University of South Carolina, USA
Elena Ferrari	University of Insubria, Italy
Simone Fischer-Hübner	Karlstad University, Sweden
Tyrone Grandison	IBM Almaden Research Center, USA
Dieter Gollmann	Technische Universität Hamburg-Harburg, Germany
Marit Hansen	Independent Centre for Privacy Protection, Germany
Min-Shiang Hwang	National Chung Hsing University, Taiwan
Mizuho Iwaihara	Kyoto University, Japan
Sushil Jajodia	George Mason University, USA
Ton Kalker	HP Labs, USA
Marc Langheinrich	Università della Svizzera italiana (USI), Switzerland
Nguyen Manh Tho	Vienna University of Technology, Austria
Nick Mankovich	Philips Medical Systems, USA
Sharad Mehrotra	University of California at Irvine, USA
Stig Frode Mjølsnes	Norwegian University of Science and Technology, Norway
Eiji Okamoto	University of Tsukuba, Japan
Sylvia Osborn	University of Western Ontario, Canada
Günther Pernul	University of Regensburg, Germany
Birgit Pfitzmann	IBM Watson Research Lab, Switzerland

Bart Preneel	K.U.Leuven, Belgium
Kai Rannenberg	Goethe University Frankfurt, Germany
Andreas Schaad	SAP Labs, France
Nicholas Sheppard	Queensland University of Technology, Australia
Jason Smith	Queensland University of Technology, Australia
Morton Swimmer	John Jay College of Criminal Justice/CUNY, USA
Clark Thomborson	University of Auckland, New Zealand
Sheng Zhong	State University of New York at Buffalo, USA

Additional Referees

Mamadou Diallo	University of California at Irvine, USA
Christian Kahl	Goethe University Frankfurt, Germany
Christian Weber	Goethe University Frankfurt, Germany

Table of Contents

Keynote Paper

Privacy Protection

Data Security in Open Environments

Assuring Data Trustworthiness - Concepts and Research Challenges*

Elisa Bertino and Hyo-Sang Lim

Department of Computer Science, Purdue University, USA
{bertino,hslim}@cs.purdue.edu

Abstract. Today, more than ever, there is a critical need to share data within and across organizations so that analysts and decision makers can analyze and mine the data, and make effective decisions. However, in order for analysts and decision makers to produce accurate analysis and make effective decisions and take actions, data must be trustworthy. Therefore, it is critical that data trustworthiness issues, which also include data quality, provenance and lineage, be investigated for organizational data sharing, situation assessment, multi-sensor data integration and numerous other functions to support decision makers and analysts. The problem of providing trustworthy data to users is an inherently difficult problem that requires articulated solutions combining different methods and techniques. In the paper we first elaborate on the data trustworthiness challenge and discuss a framework to address this challenge. We then present an initial approach for assess the trustworthiness of streaming data and discuss open research directions.

Keywords: Data Trustworthiness, Data Integrity and Quality, Security, Policy.

1 Introduction

Today, more than ever, there is a critical need to share data among organizations so that analysts and decision makers can analyze and mine the data, and make effective decisions. Meanwhile, in many recent applications such as traffic monitoring systems and power grid management systems, a large amount of data that can convey important information for critical decision making is collected from distributed sources. In order for analysts and decision makers to produce accurate analysis, make effective decisions, and take actions, data must be trustworthy. Therefore, it is critical that data trustworthiness issues, which also include data quality and provenance, be investigated for organizational data sharing, situation assessment, multi-sensor data integration, and numerous other functions to support decision makers and analysts. Without trustworthiness, the

* The authors have been partially supported by AFOSR grant FA9550-07-1-0041 "Systematic Control and Management of Data Integrity, Quality and Provenance for Command and Control Applications".

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 1–12, 2010.
© Springer-Verlag Berlin Heidelberg 2010

usefulness of data becomes diminished as any information extracted from them cannot be trusted with sufficient confidence.

It is thus important to provide a comprehensive solution for assessing and assuring the trustworthiness of the information collected in such data sharing systems since decisions and analyses are largely affected by this information. Attacks or unexpected accidents may result in bad data being provided to critical components of the system. These components may in turn take wrong decisions or generate inaccurate analyses that can result in damages to real-world objects such as manufacturing facilities or power plants. For example, in an electric power grid which consists of 270 utilities using a SCADA (Supervisory Control and Data Acquisition) system that can contain up to 50,000 data collection points and over 3,000 public/private electric utilities, any single point of failure can disrupt the entire process flow and can potentially cause a domino effect that shuts down the entire systems [11].

In general the problem of providing "good" data to users and applications is an inherently difficult problem which often depends on the application and data semantics as well as on the current context and situation. In many cases, it is crucial to provide users and applications not only with the needed data, but with also an evaluation indicating how much the data can be trusted. Being able to do so is particularly challenging especially when large amounts of data are generated and continuously transmitted across the system. Also solutions for improving the data, like those found in data quality, may be very expensive and may require access to data sources which may have access restrictions, because of data sensitivity. Also even when one adopts methodologies to assure that the data is of good quality, errors may still be introduced and low quality data be used; therefore, it is important to assess the damage resulting from the use of such data, to track and contain the spread of errors, and to recover. The many challenges of assuring data trustworthiness require articulated solutions combining different approaches and techniques including data integrity, data quality, and data provenance. In this paper we discuss some components of such a solution and highlight relevant research challenges.

Our approach for assuring information trustworthiness is based on a comprehensive framework composed of two key elements. The first element is represented by trust scores that are to associated with all data items to indicate the trustworthiness of each data item. Trust scores can be used for data comparison or ranking. They can be also used together with other factors (e.g., information about contexts and situations, past data history) for deciding about the use of the data items. Our framework provides a trust score computation method which is based on the concept of data provenance, as provenance gives important evidence about the origin of the data, that is, where and how the data is generated. The second element of our framework is the notion of *confidence policy* [8]. Such a policy specifies a range of trust scores that a data item, or set of data items, must have for use by the application or task. It is important to notice that the required range of trust scores depends on the purpose for which the data have to be used. In our framework, confidence policies are integrated with query processing in that query results are filtered by the policies before being returned to the application. Our confidence policy language includes many other components that can be used to formulate articulated conditions about data usage.

We have applied our framework to sensor networks which are being increasingly deployed in a large variety of novel areas, like supervisory systems, e-health, and e-surveillance. In all those areas, it is crucial to assess the trustworthiness of sensed data in order to support correct decisions as sensors may be deployed in hostile environments and are usually connected via wireless networks which can be an easy target for adversaries. In this application, with respect to the assessment of trust scores, we exploits an *interdependency property* between data items and sensors, i.e., the trust score of the data affects the trust score of the sensors that created and manipulated the data, and vice-versa. To reflect the interdependency, we have proposed a cyclic method that generates: (1) trust scores of data items from those of sensors and (2) trust scores of sensors from those of data items whenever new data items arrive.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our framework for assuring information trustworthiness. Section 4 describes the application of our framework to sensor network environments. Section 5 discusses open research issues and directions related to our framework. We finally summarize and conclude the paper in Section 6.

2 Related Work

Currently there is no comprehensive approach to the problem of high assurance data trustworthiness. However, our approach is related to several areas including data integrity, data quality, and data provenance.

Data Integrity. Biba [4] was the first to address the issue of integrity in information systems. He proposed an approach based on a hierarchical lattice of integrity levels so as to ensure integrity by blocking the flow of information from low-integrity objects to high-integrity subjects. However, Biba's approach is not easy to use because it is not clear how to determine the appropriate integrity levels as there are no criteria for determining them. Clark and Wilson [6] proposed another model for data integrity in commercial environments. The model consists of two key notions: well-formed transactions and separation of duty. A well-formed transaction is one that only manipulates data in trusted ways so to preserve the consistency of data. Separation of duty mandates separating all operations into several subparts and executing each subpart by a different subject. However, the model has some limitations in that it categorizes data integrity only according to two levels: valid and invalid. Our approach is more comprehensive than those models in that it provides mechanisms driven by confidence policies and provides an approach to determine the data trust level based on data provenance information.

Data quality. Ranging from data warehousing to supply chain management, data quality is a serious concern for professionals involved in a wide array of information systems. One industry study estimated the total cost to the US economy of data quality problems at over US$600 billion per annum [10]. Data quality has been investigated from different perspectives, depending on the precise meaning assigned to the notion of data quality. Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" [16]. Alternatively, the data are deemed of high quality if they correctly represent the real-world construct to which they refer. A number of theoretical frameworks consider data quality. A framework is based on semiotics to evaluate the quality of the form, meaning and use of the data [22]. Another theoretical approach analyzes the ontological nature of information systems to define data quality rigorously [26]. Tools have also been developed for analyzing and repairing poor quality data, through the use, for example, of record linkage techniques.

Data provenance. Data provenance, also referred to as lineage or pedigree, has been widely investigated. Approaches have been developed for tracking the provenance of the query results, i.e., recording the sequence of steps taken in a workflow system to derive the dataset, and computing confidence levels of the query results [1,2,12,23]. For example, Widom et al. [27] developed the Trio system which supports management of information about data accuracy and lineage (provenance). Sarma et al. [24] developed an approach to compute lineage and confidence in probabilistic databases according to a decoupled strategy. These approaches compute the confidence of query results based on the confidence on the base tuples in the database, i.e., they assume that the confidence of each base tuple (i.e., data item) is known, whereas we actually compute the trust score for each data item. Vijayakumar and Plale [25] propose a system architecture for near-real time provenance collection in data streams. They focus on identifying information which represents provenance of a data item from real time data streams, capturing provenance history of streams, tracing the source of a stream long after the process has completed. Data provenance in data streaming has also been investigated in IBM's Century [5,20], which is a biomedical data stream system for online healthcare analytics.

3 A Framework for Assuring Information Trustworthiness

Figure 1 shows an overview of our framework for assuring information trustworthiness. The figure illustrates how data are managed, processed, and delivered to users. As shown in the figure, the proposed framework consists of three major components: *trustworthiness assessment*, *query and policy evaluation*, and *data quality management*. The role of each component is as follows:

Trustworthiness assessment associates trust scores with data in the database. A trust score is a numeric value ranging from 0 to 1 where 0 means the lowest trustworthiness and 1 means the highest trustworthiness. Such a trust score is a key notion of our framework since it indicates trustworthiness of the data item and can be used for data comparison or ranking. Trust scores can be obtained by using various factors such as the trustworthiness of data providers and the way in which the data has been collected. In our framework, we especially focus on data provenance for computing trust scores. This idea is based on the principle that the more trustworthy data a source provides, the more trusted the source is considered. There is thus an *interdependency property* between data providers and data items with respect to the assessment of their trust scores, i.e., the trust score of the data affects the trust score of the data providers that created and manipulated the data, and vice-versa. To reflect such interdependency in assessing trustworthiness, we maintain trust scores for both data items and data providers. That is, the trustworthiness assessment component computes (1) trust scores of data items based on those of data providers and (2) trust scores of data providers based on those of data items.

Query and policy evaluation executes user queries, each of which has its own *confidence policy* [3,8]. Confidence policies specify confidence range $[q_{max}, q_{min}]$ that query results must have for use by the application or task. Such a confidence policy is a key notion of our framework since it restricts access to the query results based on the trust score of the query results and application's requirements. We refer to queries enforcing such policies as *confidence policy-compliant queries*. For each user query Q with its confidence policy $[q_{max}, q_{min}]$, the query and policy evaluation component first obtains the resulting data items by evaluating the given query Q, and then, returns data items of which trust scores are in between q_{max} and q_{min}.

Data quality management tries to control data quality manually or automatically. Since some query results will be filtered out by the confidence policy, a user may not receive enough data to make a decision. To prevent such a lack of information in applications, we need to provide a method to dynamically increment the data trust scores. The method includes collecting additional data, changing data delivery paths, and cross-checking the data with other information. The component should consider the cost for improving data quality when determining which data should be selected and how much the trust score should be increased to satisfy the confidence range stated by the confidence policies.

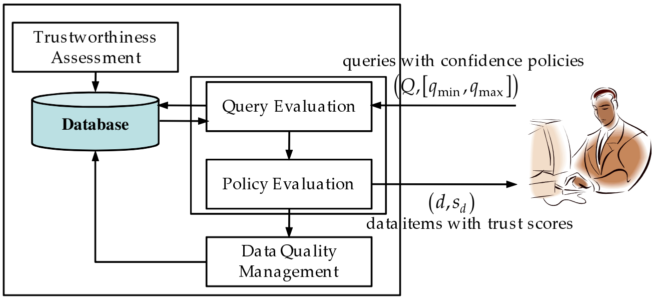


Fig. 1. An overall framework for assuring information trustworthiness

Our framework has many advantages in assuring information trustworthiness. First, it provides a measure for trustworthiness of data (i.e., trust scores) and an approach to determine those values. Second, it provides policies by using which one can specify which the confidence is required for use of certain data in certain tasks. With such policies, our solution supports fine-grained integrity tailored to specific data and tasks. Third, it provides an approach to dynamically adjust the data trustworthiness so to provide users with query replies that comply with the confidence policies.

4 An Application to Sensor Networks

In this section, we discuss the application of our framework to sensor networks.

4.1 Adopting the Framework into Sensor Networks

In sensor networks, it is extremely important to assess data quality since sensors can be deployed in hostile environments and the wireless networks connecting them can be easily attacked. Therefore, it is crucial to assess the quality of sensed data and increase the quality whenever possible, in order to support correct decisions in sensor network applications. Additionally, in sensor network environments, 1) data items arrive incrementally and 2) trustworthiness of sensors and their readings can dynamically change as time goes on. Figure 2 shows how our framework is adopted for such dynamic sensor networks.

In the framework, the trustworthiness assessment component obtains trust scores of data items based on those of network nodes and (periodically) updates the trust scores to reflect the effect of the newly arrived data items. It also maintains and updates trust scores of network nodes based on the scores of data items. Query and policy evaluation components continuously execute queries (each of which has its own confidence policy) whenever a new data item arrives from sensors. The data quality management component improves trustworthiness by adjusting data rates, increasing/decreasing the number of sensor nodes, or changing delivery paths. Obviously, data quality affects trust scores, and many approaches [7,14,21] in the sensor networks area have addressed the issue of controlling data quality. In this section, we focus on trustworthiness assessment so as to get reasonable trust scores for sensor readings.

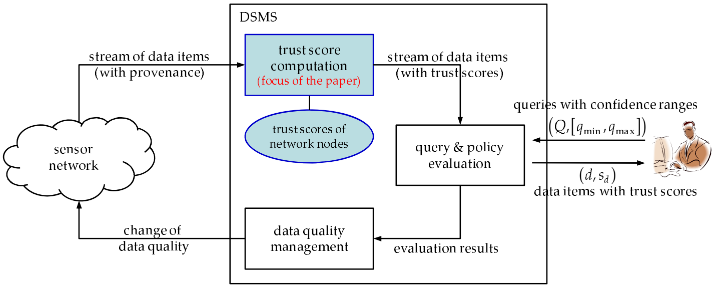


Fig. 2. Adopting our framework into sensor networks

4.2 Modeling Sensor Networks and Data Provenances

Before describing the computation of trust scores, we explain data provenance which is an important information for assessing trustworthiness in our approach. We model the sensor network as a graph of $G(N, E)$ where N is a set of nodes (i.e., sensors) and E is a set of network connections between nodes. We assume that the sensors are in ad-hoc networks which send data items to DSMSs by relaying because of the insufficient data transmission bandwidth. In this kind of network, the sensors are categorized into three types according to their roles: 1) *terminal nodes* that generate data items and send them to one or more intermediate or server nodes; 2) *intermediate nodes* that receive data items from one or more terminal or intermediate nodes, and pass them to other intermediate or server nodes; they may also generate an aggregated data item from the received data items and send the aggregated item to other intermediate or server nodes; 3) *server nodes* that receive data items and evaluate continuous queries based on those items.

We introduce two types of data provenance: the *physical provenance* and the *logical provenance*. The physical provenance of a data item indicates where the item was produced and how it was delivered to the server. We exploit the physical provenance to compute trust scores. The physical provenance can be represented as a path from a terminal node to a server node or a tree if there are more than two terminal nodes involved in the generation of the data item. The logical provenance of a data item represents the semantic meaning of the data item in the context of a given application. For example, the logical provenance can be a chain of employees who used the data item, or a trail of business logics that processed the data item. The logical provenance is used for grouping data items into semantic events with the same meaning or purpose.

4.3 Computing Trust Scores

To obtain trust scores, we propose a cyclic framework based on the *interdependency property* [3,8] between data items and their related network nodes. That is, the trust scores of data items affect the trust scores of network nodes, and similarly the trust scores of network nodes affect those of data items. In addition, the trust scores need to be continuously evolved in the stream environment since new data items continuously arrive to the server. Thus, a cyclic framework is adequate to reflect these interdependency and continuous evolution properties. Figure 3 shows the cyclic framework according to which the trust score of data items and the trust score of network nodes are continuously updated. The trust scores are computed for the data items of the same event (identified by logical provenance) in a given streaming window.

As shown in Figure 3, we maintain three different types of trust scores, that is, *current*, *intermediate*, and *next trust scores*, to reflect the interdependency and continuous evolution properties in computing trust scores. We note that, since new data items are continuously added to the stream, executing the cycle once whenever a new data item arrives is enough to reflect the interdependency and continuous evolution properties in the stream environment.

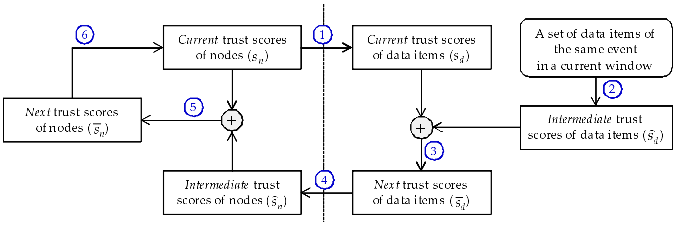

6
Current trust scores of nodes (s_n)
1
Current trust scores of data items (s_d)
A set of data items of the same event in a current window
2
Next trust scores of nodes ($\bar{s}_n$)
5
+
+
Intermediate trust scores of data items ($\hat{s}_d$)
3
Intermediate trust scores of nodes ($\hat{s}_n$)
4
Next trust scores of data items ($\bar{s}_d$)

Fig. 3. The cyclic framework for computing the trust scores of data items and network nodes

Our framework works as follows. Trust scores are initially computed based on the values and provenance of data items; we refer to these trust scores as *implicit trust scores*. To obtain these trust scores, we use two types of similarity functions: *value similarity* inferred from data values, and *provenance similarity* inferred from physical provenances. Value similarity is based on the principle that the more data items referring to the same real-world event have similar values, the higher the trust scores of these items are. We observe that most sensor data referring to the same event follow the *normal distribution*, and propose a systematic approach for computing trust scores based on value similarity under the normal distribution. Provenance similarity is based on the observation that different physical provenances of similar data values may increase the trustworthiness of data items. In other words, different physical provenances provide more independent data items. For more details, please refer to our technical report [18].

5 Research Issues and Challenges

In this section, we discuss open research directions for each component of our framework described in Section 3.

Trustworthiness Assessment

– Data/provenance similarity measures. Measuring data similarity is essential in our trust score computation. If we only handle numeric values, the similarity can be easily measured with the difference or Euclidian distance of the values. However, if the value is non-numeric (such as text data), we need to include modeling techniques able to take into account data semantics. For example, if data are names of places, we need to consider spatial relationships in the domain of interest. Possible approaches that can be used include semantic web techniques, like ontologies and description logics. Similarly, measuring provenance similarity is not easy especially when the provenance is complex. The edit distance which uses the minimum amount of distortion that is needed to transform one graph into another is the most popular similarity measure in graph theory. However, computing the edit distance for general graphs is known to be an NP-hard problem [13]. Therefore, we need an approximate similarity measure method which efficiently compares graph-shape provenances.

- Correlation among data sources. The relationships among the various data sources could be used to create more detailed models for assigning trust to each data source. For example, if we do not have good prior information about the trustworthiness of a particular data source, we may try to use distributed trust computation approaches such as EigenTrust [17] to compute a trust value for the data source based on the trust relationships among data sources. In addition, even if we observe that the same data is provided by two different sources, if these two sources have a very strong relationship, then it may not be realistic to assume that the data is provided by two independent sources. An approach to address such issue is to develop "source correlation" metrics based on the strength of the relationship among possible data sources. Finally, in some cases, we may need to know "how important is a data sources within our information propagation network?" to reason about possible data conflicts. To address such issue one can apply various social network centrality measures such as degree, betweenness, closeness, and information centralities [15] to assign importance values to the various data sources.
- Secure data provenance. Since data provenance is a key evidence for measuring the trustworthiness of data items, it should be protected from tampering when flowing across the various parties. For example, if the provenance of an untrustworthy data item is modified to a trustworthy node, the quality of the data item is assessed as high, even though in reality the data item has a low quality. Conventional digital signature or cryptography techniques can be used to prevent such malicious attacks. Another approach can be based on techniques for controlled and cooperative updates of XML documents in Byzantine and failure-prone distributed systems [19]. One could develop an XML language for encoding provenance information and use such techniques to secure provenance documents. We also should consider reducing the overhead of these techniques since it directly affects to the performance of data collection and query processing.

Query and Policy Evaluation

- Expressive power of confidence policies. We need to improve the expressivity of the confidence policy model since the current policy language is very simple. It should be possible to support a more fine-grained specification of confidence requirements concerning data use whereby for a given task and role, one can specify different trust scores for different categories of data. The model should support the specification of subjects, in terms of subject attributes and profiles other than the subject role. If needed, exceptions to the policies should be supported; a possible approach is to support strong policies, admitting no exceptions, and weak policies, admitting exceptions. If exceptions are allowed for a policy (set of policies), the policy enforcement mechanism should support the gathering of evidence about the need for exceptions. Such evidence is crucial in order to refine confidence policies.

- Policy provisioning. An important issue is related to the confidence policy provisioning. Provisioning refers to assigning confidence policies to specific tasks, users, and data and, because it very much depends from the applications and data semantics, it may be quite difficult. To address such issue, one approach is the use of machine learning techniques.
- Performance. Efficient evaluation of confidence policies is crucial to achieve high performance of responding user query requests. Especially, in sensor network environment, it becomes more important due to fast input rates and real-time processing requirement. One possible solution is tightly combining query processing and policy evaluation so as to retrieve data items which satisfy both query conditions and trust score ranges at once.

Data Trustworthiness Improvement

- Efficiently improve data trustworthiness. We use data provenance for not only measuring but also improving data trustworthiness. As we discussed in Section 3, the data quality management component tries to improve data trustworthiness when the number of results satisfying a confidence policy is too small. Many methods can be adopted to improve trustworthiness: from manually checking data items by hand to automatically comparing data items with other information. The cost of each method depends on the optimization criteria adopted, like time and financial cost. Therefore, suitable cost models need to be developed. Also, trustworthiness improvement algorithms need to be devised for determining suitable data items for which the increase in the trust scores can lead to the minimum cost. Because it is likely that finding the optimal solution may be computationally very expensive, heuristics need to be devised.

6 Concluding Remarks

In this paper, we have discussed a comprehensive framework for assuring information trustworthiness, based on two novel concepts: trust scores and confidence policies. Our framework is very general and can be used in different application domains. In the paper we have discussed the application to the problem of assessing the trustworthiness of data streaming in sensor network. Another application has been developed for assessing the trustworthiness of locations of individuals [9]. Our work has many open research directions, which we have also discussed in the paper. In addition to these directions, developing new applications is an important step for future work.

References

1. Abiteboul, S., Kanellakis, P., Grahne, G.: On the Representation and Querying of Sets of Possible Words. SIGMOD Record 16(3), 34–48 (1987)
2. Barbara, D., Garcia-Molina, H., Porter, D.: The Management of Probabilistic Data. IEEE Trans. on Knowledge and Data Engineering 4(5), 487–502 (1992)

3. Bertino, E., Dai, C., Lim, H.-S., Lin, D.: High-Assurance Integrity Techniques for Databases. In: Proc. of the 25th British Nat'l Conf. on Databases, Cardiff, UK, pp. 244–256 (July 2008)
4. Biba, K.J.: Integrity Considerations for Secure Computer Systems. Technical Report TR-3153, Mitre (1977)
5. Blount, M., et al.: Century: Automated Aspects of Patient Care. In: Proc. 13th IEEE Int'l. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007), Daegu, Korea, pp. 504–509 (August 2007)
6. Clark, D., Wilson, D.: A Comparison of Commercial and Military computer Security Policies. In: Proc. of IEEE Symposium on Security and Privacy (1987)
7. David, Y., Erich, N., Jim, K., Prashant, S.: Data Quality and Query Cost in Wireless Sensor Networks. In: Proc. of the 5th IEEE Int'l Conf. on Pervasive Computing and Communications Workshops, White Plains, New York, USA, pp. 272–278 (March 2007)
8. Dai, C., et al.: Query Processing Techniques for Compliance with Data Confidence Policies. In: Proc. of the 6th VLDB Workshop on Secure Data Management, Lyon, France, pp. 49–67 (2009)
9. Dai, C., Lim, H.-S., Bertino, E., Moon, Y.-S.: Assessing the Trustworthiness of Location Data Based on Provenance. In: Proc. of the 17th ACM SIGSPATIAL Int'l Symposium on Advances in Geographic Information Systems, Seattle, USA, pp. 276–285 (November 2009)
10. Eckerson, W.: Data Warehousing Special Report: Data quality and the bottom line, `http://www.adtmag.com/article.aspx?id=6321`
11. Fernandez, J., Fernandez, A.: SCADA systems: vulnerabilities and remediation. Journal of Computing Sciences in Colleges 20(4), 160–168 (2005)
12. Fuhr, N.: A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In: Proc. of the 16th Int'l. Conf. on Very Large Data Bases, Brisbane, Australia, pp. 696–707 (August 1997)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness (1990)
14. Hwang, S.-Y., Liu, P.-Y., Lee, C.-H.: Using Resampling for Optimizing Continuous Queries in Wireless Sensor Networks. In: Proc. of the 8th Int'l. Conf. on Intelligent Systems Design and Applications, Kaohsiung, Taiwan, pp. 107–110 (November 2008)
15. Jackson, M.O.: Social and Economics Networks. Princeton University Press, Princeton (2008)
16. Juran, J.M.: Juran on Leadership for Quality - an Executive Handbook. Free Press (1989)
17. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: Proc. of Twelfth Int'l. World Wide Web Conf., New York, USA, pp. 640–651 (2003)
18. Lim, H.-S., Moon, Y.-S., Bertino, E.: Assessing the Trustworthiness of Streaming Data. Technical Report TR 2010-09, CERIAS (2010)
19. Mella, G., Ferrari, E., Bertino, E., Koglin, Y.: Controlled and Cooperative Updates of XML Documents in Byzantine and Failure-prone Distributed Systems. ACM Transactions on Information and System Security 9, 421–460 (2006)
20. Misra, A., Blount, M., Kementsietsidis, A., Sow, D.M., Wang, M.: Advances and Challenges for Scalable Provenance in Stream Processing Systems. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 253–265. Springer, Heidelberg (2008)

21. Olston, C., Jiang, J., Widom, J.: Adaptive Filters for Continuous Queries over Distributed Data Streams. In: Proc. Int'l. Conf. on Management of Data, pp. 563–574. ACM SIGMOD, San Diego (June 2003)
22. Price, R., Shanks, G.: A Semiotic Information Quality Framework. In: Proc. IFIP International Conference on Decision Support Systems (DSS 2004), Prato, Italy (2004)
23. Sarma, A.D., Benjelloun, O., Halevy, A., Widom, J.: Working Models for Uncertain Data. In: Proc. of the 22nd Int'l. Conf. on Data Engineering, Atlanta, USA, p. 7 (April 2006)
24. Sarma, A.D., Theobald, M., Widom, J.: Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. In: Proc. of the 24th Int'l. Conf. on Data Engineering, Cancun, Mexico, pp. 1023–1032 (April 2008)
25. Vijayakumar, N., Plale, B.: Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 46–54. Springer, Heidelberg (2006)
26. Wand, Y., Wang, R.: Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM 39(11), 86–95 (1996)
27. Widom, J.: Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In: Proc. of the 2nd Biennial Conf. on Innovative Data Systems Research, Asilomar, USA, pp. 262–276 (January 2005)

On-the-Fly Hierarchies for Numerical Attributes in Data Anonymization

Alina Campan and Nicholas Cooper

Department of Computer Science, Northern Kentucky University,
Highland Heights, KY 41099, USA
campana1@nku.edu, coopern1@mymail.nku.edu

Abstract. We present in this paper a method for dynamically creating hierarchies for quasi-identifier numerical attributes. The resulting hierarchies can be used for generalization in microdata *k*-anonymization, or for allowing users to define generalization boundaries for constrained *k*-anonymity. The construction of a new numerical hierarchy for a numerical attribute is performed as a hierarchical agglomerative clustering of that attribute's values in the dataset to anonymize. Therefore, the resulting tree hierarchy reflects well the closeness and clustering tendency of the attribute's values in the dataset. Due to this characteristic of the hierarchies created on-the-fly for quasi-identifier numerical attributes, the quality of the microdata anonymized through generalization based on these hierarchies is well preserved, and the information loss in the anonymization process remains in reasonable bounds, as proved experimentally.

Keywords: data privacy, *k*-anonymity, hierarchies for quasi-identifier numerical attributes.

1 Introduction

Attribute generalization is a disclosure control technique intensively used in data anonymization methods, together with data suppression. Generalization of quasi-identifier attributes is, for example, applied to mask initial microdata (a dataset where each tuple corresponds to one individual entity) to make it conform to *k*-anonymity-like models: *k*-anonymity [1], [2], *l*-diversity [3], *p*-sensitive *k*-anonymity [4], (α, *k*)-anonymity [5], *t*-closeness [6], (ε, *m*)-anonymity [7], l^+-diversity [8], (τ, λ)-uniqueness [9] etc. The resulting data will present the *k*-anonymity property, which means that each tuple in the masked set will be identical with respect to the quasi-identifier attributes to at least *k*-1 other tuples in the set; also, depending on the anonymity model used, the masked data will also exhibit other statistical properties.

Generalization of a quasi-identifier attribute consists in replacing the actual value of the attribute with a less specific, more general value that is faithful to the original [10]. Initially, this technique was used for *categorical* attributes and employed *predefined* (*static*) domain and value generalization hierarchies [10]. Generalization was extended for *numerical* attributes either by using *predefined hierarchies* [11] or a

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 13–25, 2010.
© Springer-Verlag Berlin Heidelberg 2010

hierarchy-free model [12]. The important distinction between using predefined hierarchies and performing a hierarchy-free generalization is as follows. Predefined hierarchies are constructed, normally manually by users, *before* the data masking process begins, and these hierarchies are used as they are, unchanged, for generalization. The hierarchy-free generalization inherently leads to the construction of a hierarchy *during* the anonymization process; the creation of such a hierarchy is usually guided by local optimum decisions taken during anonymization.

Another important aspect is how hierarchies are used for generalization. For a categorical attribute, generalization is based on a *domain generalization hierarchy* associated to that attribute. The values from different domains/levels of this hierarchy are represented in a tree called a *value generalization hierarchy*. Fig. 1 shows two examples of domain and value generalization hierarchies for attributes *zip_code* and *gender* of a microdata set containing personal information for individuals.

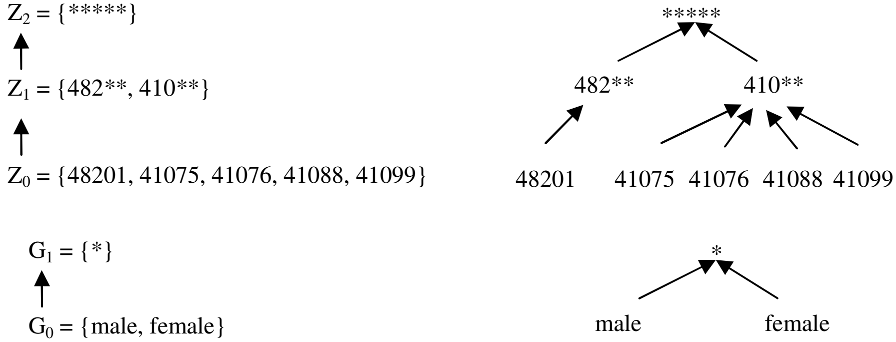


Fig. 1. Examples of domain and value generalization hierarchies

Generalization of categorical attributes, based on domain and value generalization hierarchies, comes in several flavors.

Generalization that maps all values of a quasi-identifier categorical attribute in the microdata set to a more general domain in its domain generalization hierarchy is called *full-domain generalization* [1], [12]. For example, a full-domain generalization for *zip_code* can consist in replacing all initial zip codes in the microdata with their corresponding ancestors located on the middle level in the value generalization hierarchy in Fig. 1: 48201 with 482**, 41075, 41076, 41088, and 41099 with 410**.

Generalization can also map an attribute's values to different domains in its domain generalization hierarchy, each value being replaced by the same generalized value in the entire dataset [11]. This type of generalization was introduced by Iyengar in [11]. In this type of generalization, 48201 would be replaced with one and only one of its ancestors, for example 482**, in the entire dataset, while another one of the original values, let's say 41075, could be replaced with its ***** ancestor across the entire dataset; note that the replacement, more general values, 482** and *****, are located on different levels in *zip_code*'s value generalization hierarchy.

Finally, the least restrictive generalization, called *cell level generalization* [13], extends Iyengar's model [11] by allowing the same value to be mapped to different

Tuples	age	zip_code	gender
r_1	25	41076	Male
r_2	25	41075	Male
r_3	35	41099	Female
r_4	38	48201	Female
r_5	36	41075	Female

Initial non-generalized microdata set, *zip_code* and *gender* are categorical attributes with the hierarchies defined in Fig. 1.

Tuples	age	zip_code	gender
r_1	20-30	*****	Male
r_2	20-30	*****	Male
r_3	30-40	*****	Female
r_4	30-40	*****	Female
r_5	30-40	*****	Female

Masked (i.e. anonymized) microdata set, *k*=2, *full-domain generalization.*

Tuples	age	zip_code	gender
r_1	20-40	410**	*
r_2	20-40	410**	*
r_3	20-40	*****	*
r_4	20-40	*****	*
r_5	20-40	410**	*

Masked microdata set, *k*=2, *Iyengar generalization.*

Tuples	age	zip_code	gender
r_1	20-30	410**	Male
r_2	20-30	410**	Male
r_3	30-40	*****	Female
r_4	30-40	*****	Female
r_5	30-40	*****	Female

Masked microdata set, *k*=2, *cell-level generalization.*

Fig. 2. Examples of different types of generalizations

generalized values, in distinct tuples. Therefore, if cell level generalization is used, the same original zip code value 48201 could be replaced in different tuples with different ancestors: in our case, either 482** or *****.

We illustrate in Fig. 2 the differences between the above-mentioned types of generalization.

Generalization of numerical attributes using predefined hierarchies is similar to the generalization for categorical attributes.

The hierarchy-free generalization of numerical attributes replaces the set of values to be generalized to the smallest interval that includes all the initial values. For instance, the values: 35, 38, 36 for the attribute *age* are generalized to the interval [35-38]; again, the decision to replace these values with the mentioned interval is taken during the anonymization process and might apply only to a subset of the tuples in the microdata set. Other occurrences of the original values (35, 38, and 36) may be replaced with other values/intervals in other tuples in the microdata set. Note that overlapping of the intervals formed during generalization is possible.

While creating and using generalization hierarchies for categorical attributes is expected to be easy and natural enough for users of anonymization methods, constructing and employing predefined hierarchies for numerical attributes is not as straightforward. Usually, creating and understanding a hierarchy for categorical attributes is easier than for numerical attributes: the values of the categorical attribute are well established, discrete, have a natural hierarchical organization, while numerical attributes might have more diverse values, maybe are continuous, and rarely have a natural hierarchical structure. However, employing a "good" hierarchy in the anonymization process significantly impacts the quality of the masked data; depending on how well the hierarchy fits the distribution and grouping of the attribute's values in the microdata set, the masked data can lose more or less from the information it inherently contains.

Hierarchy-free generalization for numerical attributes is a more flexible disclosure control technique, and helps to minimize the information loss that occurs in the masking process. It presents, however, an important down come: certain data anonymity models, such as constrained *k*-anonymity (which relies on boundaries imposed on the amount of generalization allowed in the anonymization process) require pre-existing hierarchies for numerical attributes [14].

We propose in this paper a new method for creating hierarchies for numerical attributes, based on its values in the dataset to be anonymized. The resulting hierarchy reflects the attribute's values distribution and grouping (clustering), such that to support a reasonable information loss when used in anonymization.

The paper is structured as follows. The next section introduces our method for dynamically creating hierarchies for quasi-identifier numerical attributes. Section 3 explains how numerical hierarchies are used in *k*-anonymization and introduces information loss – a metric frequently employed in the data privacy domain for assessing the masked microdata quality, and which can be used both to guide the anonymization process and to quantify the anonymized data's value. Section 4 shows *k*-anonymization results comparatively, for the same dataset, when quasi-identifier numerical attributes have predefined hierarchies, no hierarchies at all (hierarchy-free generalization is used in this situation), and respectively hierarchies created on-the-fly with our method. Results are compared with respect to the information loss incurred in the anonymization process, which is impacted not only by the anonymization procedure/algorithm, but also by the hierarchies used to perform generalization. To make the comparison accurate, the same anonymization algorithm ([15]) has been used in all three cases. The paper ends with conclusions and references.

2 Creating On-the-Fly Hierarchies for Numerical Attributes

Let $\mathcal{IM}$ be the initial microdata set and $\mathcal{MM}$ be the released (a.k.a. masked) microdata. We assume that $\mathcal{MM}$ is *k*-anonymous, or possibly agrees with a *k*-anonymity-like model. $\mathcal{IM}$ consists in a set of tuples over an attribute set. These attributes are classified into the following three categories:

- I_1, I_2,..., I_p are *identifier* attributes such as *name* and *ssn* that can be used to identify a record. These attributes are removed from the masked microdata because they express information which can lead to a specific entity.
- K_1, K_2,…, K_n are *key* or *quasi-identifier* attributes such as *zip_code* and *age* that may be known by an intruder. Quasi-identifier attributes are present both in the masked microdata as well as in the initial microdata.
- S_1, S_2,…, S_r are *confidential* or *sensitive* attributes such as *diagnosis* and *salary* that are assumed to be unknown to an intruder. Confidential attributes are present in the masked microdata as well as in the initial microdata.

Let $\mathcal{KN} = \{N_1, N_2, ..., N_s\}$ be the set of numerical quasi-identifier attributes and $\mathcal{KC} = \{C_1, C_2,..., C_t\}$ be the set of categorical quasi-identifier attributes. We denote by $\mathcal{K}= \{K_1, K_2,..., K_n \} = \mathcal{KN} \cup \mathcal{KC}$.

Let N_i be one of the numerical quasi-identifier attributes for which a hierarchy is necessary. The motivations for employing such a hierarchy can be diverse: for use to generalize attribute's values in the *k*-anonymization process, or for describing constraints on that attribute for the constrained *k*-anonymity model. We denote by $V = \{v_1, v_2, ..., v_m\}$ the distinct values of N_i in the dataset $\mathcal{IM}$. Each one of these values can have a single or multiple occurrences for N_i in $\mathcal{IM}$.

We adopt an agglomerative hierarchical clustering approach ([16]) for dynamically creating hierarchies for numerical attributes such as N_i. In the agglomerative approach, the building (construction) of a hierarchy is guided by the distribution and clustering of the values of the target attribute (for which the hierarchy is generated) in the dataset to anonymize.

Specifically, our algorithm to create on-the-fly hierarchies for numerical attributes is as follows. First, a set of *m* nodes is constructed, one node for each of the *m* unique values within the dataset for the numerical attribute of interest. These nodes will be the leaves of the tree-like domain value hierarchy $\mathcal{H}_{Ni}$ for the attribute N_i. From this initial set of nodes, the tree is built from bottom to top (i.e. from leaves to root), by merging at each step the "closest" two nodes in the current set of nodes. The hierarchy is completely built when only one node is left in our current set of nodes; this single node is the root of the hierarchy. We will clarify next what closeness and distance between two nodes means in our approach, and how a merging of two nodes is carried on. An important fact to note is that the resulting hierarchy is a tree, called a dendrogram, which is usually not balanced, and which can have its leaves on any level under the root.

Definition 1 *(a node in the numerical hierarchy).* We will denote each of the *m* leaf nodes in $\mathcal{H}_{Ni}$ by its representative value $v_i \in V$, and the set of leaves in $\mathcal{H}_{Ni}$ by *V*. Each node in $\mathcal{H}_{Ni}$, leaf of internal, is characterized by four values:

- a quantity *q* (*count*), representing the number of occurrences of the value or set of values that the node represents in the dataset $\mathcal{IM}$, for the attribute N_i,
- *min* and *max*, the *minimum* and *maximum* numerical values represented by the node, and
- *avg,* the *average* value represented by the node, as a quantification / approximation of the centroid of the cluster (i.e. the group of values in the dataset that are summarized by the current node in the hierarchy).

For a leaf node v_i, *min*, *max*, and *avg* are the same value (v_i), and *q* is the actual number of occurrences of v_i in the dataset $\mathcal{IM}$.

Definition 2 *(distance between two nodes in the numerical hierarchy).* We compute the distance between two nodes $X_i = (q^i, min^i, max^i, avg^i)$ and $X_j = (q^j, min^j, max^j, avg^j)$ as $|avg^i - avg^j|$, i.e. the distance between the nodes' centroids.

The closest two nodes in the current set of nodes are those that present the minimum distance between their centroids.

Definition 3 *(merging nodes in the numerical hierarchy).* If the closest two nodes are determined to be $X_i = (q^i, min^i, max^i, avg^i)$ and $X_j = (q^j, min^j, max^j, avg^j)$, they are merged to create a new node $X_k = (q^k, min^k, max^k, avg^k)$, where:

- $min^k = \min(min^i, min^j)$,
- $max^k = \max(max^i, max^j)$,
- $n^k = n^i + n^j$,
- $avg^k = (avg^i * n^i + avg^j * n^j) / (n^i + n^j)$; it can be very simply proved that avg^k is the average of the cluster of values represented by X_k.

Both X_i and X_j are made descendants of X_k and are removed from the current node set when merged.

The size of the current set of nodes is therefore reduced by one when two nodes are merged, and after *m*-1 iterations, only one node will remain in the set. This node becomes the root of the hierarchy, its *min*, *max*, and *avg* values represent the minimum, maximum, and respectively the average of all the values in the dataset for the attribute N_i. Its *q* value represents the number of tuples within the initial set $\mathcal{IM}$.

We give next the pseudocode for the hierarchy construction algorithm.

```
Algorithm NumericalHierarchy is
   Input:  $\mathcal{IM}$, attribute $N_i$
           The distance metric $d$ between two nodes in $\mathcal{H}_{Ni}$
   Output: $\mathcal{H}_{Ni}$;
   Extract from $\mathcal{IM}$ the leaf nodes in $\mathcal{H}_{Ni}$, $V = \{v_1, v_2, …, v_m\}$; each $v_i \in V$ has:
         - $v_i.min = v_i.max = v_i.avg$ = value $v_i$;
         - $v_i.q$ = number of occurrences of $v_i$ in $\mathcal{IM}$ for $N_i$;
   $\mathcal{H}_{Ni} = V$;
   Repeat
        $(X_i^*, X_j^*) = \text{argmin}_{(Xi, Xj) \in V \times V} (d(X_i, X_j))$;
        // merge, in the sense described by Definition 3,
        // the two closest nodes in $V$, $X_i^*$, $X_j^*$, into $X_{new}$
        $X_{new} = X_i^* \cup X_j^*$; $X_{new}$ has:
         - $X_{new}.min$ = min $(X_i^*.min, X_j^*.min)$;
         - $X_{new}.max$ = max $(X_i^*.\text{max}, X_j^*.max)$;
         - $X_{new}.q = X_i^*.q + X_j^*.q$;
         - $X_{new}.avg = (X_i^*.avg * X_i^*.q + X_j^*.avg * X_j^*.q) /$
           $(X_i^*.q + X_j^*.q)$;
        Make $X_{new}$ parent in $\mathcal{H}_{Ni}$ for $X_i^*$ and $X_j^*$;
        $V = V - \{X_i^*, X_j^*\} \cup \{X_{new}\}$;
   Until $(|V| = 1)$;
   The remaining node in $V$ is the root of $\mathcal{H}_{Ni}$;
End NumericalHierarchy.
```

The hierarchies produced by this algorithm are shaped as binary trees and can be very deep, due to how they are created – they can actually have a maximum of m-1 levels. One possible way to limit the size (height) of this kind of automatically generated hierarchy is to cut out some of the levels of the tree, for example, every second or third level of the tree. This translates to linking the parents of the nodes on the deleted levels to their "grandchildren"; leaf nodes on the levels to delete would have to be preserved though, in order not to render the hierarchy incomplete and loose values from the attribute's domain.

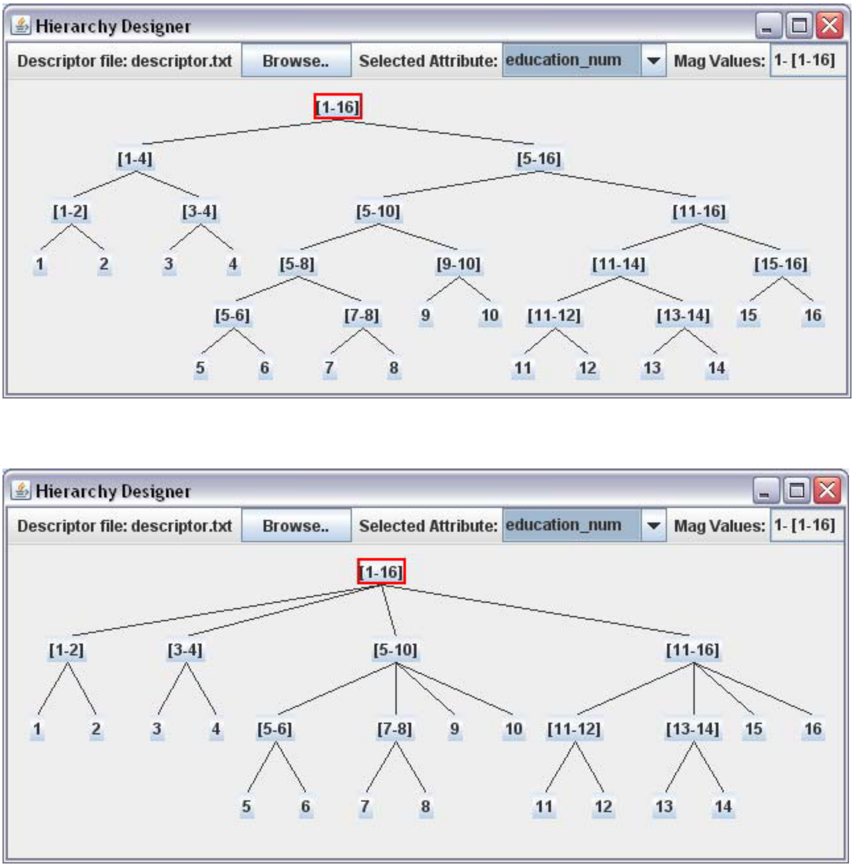
Fig. 3 shows an example of two hierarchies – the original dendrogram and the dendrogram with every second level cut out, for the numerical attribute *education_num*, which is a quasi identifier attribute in our experimental dataset. The figures are as produced by our GUI application that allows the user to: create new hierarchies for numerical attributes, customize them, define generalization boundaries (if the hierarchy will be used in constrained k-anonymity), save and load hierarchies in a proper format that can be used next for anonymization.

The complexity of the *NumericalHierarchy* algorithm is O(m^2). This is because, in each merging step, the two nodes to be merged (= the two closest nodes in the current set of nodes) can only be neighbor nodes in the list of current nodes V, sorted by the nodes' avg. (Therefore, the list V is initially created and will be maintained sorted.) Consequently, finding the closest pair of nodes in the current list of nodes V implies computing and comparing $|V|$-1 pairs of nodes. As the size of V evolves from m to 1, the overall cost is $\sum_{l=1}^{m-1} l$, therefore the O(m^2). The complexity can still be too large if the number of distinct values of N_i in $\mathcal{IM}$ is large; this situation can easily occur when the numerical attribute under consideration is continuous. In such cases, the algorithm can start with a seed set V of nodes that each represents a small bin of values of N_i, instead of a single value in N_i.

3 *K*-Anonymity and Numerical Quasi-identifier Attributes with Hierarchies

We performed a series of experiments to identify the impact of the new method for generating hierarchies for numerical quasi-identifier attributes. In each case, a dataset has been k-anonymized with the same algorithm and considering different types of hierarchies for the quasi-identifier numerical attributes (predefined or generated on-the-fly with our method) or no hierarchies at all. The obtained masked microdata have been next measured to determine the information loss incurred in the anonymization process. For computing the information loss, we used the measure defined in the following. (Note that this kind of *IL* measure is a metric often employed in the data privacy domain for assessing the masked microdata quality, and is not related to Shannon's traditional entropy measure.)

Let $\mathcal{K} = \{K_1, K_2, \ldots, K_n\}$ be the set of quasi-identifiers for the microdata set to anonymize, $\mathcal{IM}$. We consider all of the quasi-identifiers to be numerical.



Hierarchy Designer

Descriptor file: descriptor.txt Browse.. Selected Attribute: education_num Mag Values: 1- [1-16]

Hierarchy Designer

Descriptor file: descriptor.txt Browse.. Selected Attribute: education_num Mag Values: 1- [1-16]

Fig. 3. Examples of different value generalization hierarchies generated dynamically

Definition 4 *(information loss due to generalization).* Let $cl = \{e_{i1,} e_{i2, \ldots,} \; e_{im}\}$ be a set of m entities from $\mathcal{IM}$, $e_{ir} \,|\mathcal{K} = (e_{ir}{}^{1},\ e_{ir}{}^{2},\ \ldots,\ e_{ir}{}^{n})$, for all $r = 1..m$; $|\mathcal{K}$ denotes the relational projection operation of a tuple on the set of attributes $\mathcal{K}$. The information loss caused by generalizing $e_{ir}\,|\mathcal{K}$, $r = 1..m$, to the same "tuple", denoted by $IL(cl)$, is defined as follows:

- If $K_1, K_2, \ldots, K_n$ do not have hierarchies (i.e., hierarchy-free generalization), then $e_{ir}{}^{k}$, $r = 1..m$, are generalized to the interval $[m^k, M^{\,k}] = [\min(e_{i1}{}^{k}, e_{i2}{}^{k}, \ldots, e_{im}{}^{k}),\ \max(e_{i1}{}^{k}, e_{i2}{}^{k}, \ldots, e_{im}{}^{k})]$, for all $k = 1..n$ (hierarchy-free generalization), and

  $$IL(cl) = |cl| \times \sum_{k=1}^{n} \frac{M^k - m^k}{max_{e \in IM} e|N_k - min_{e \in IM} e|N_k}$$

- If K_1, K_2,…, K_n have hierarchies (predefined or created on-the-fly), then $e_{ir}{}^{k}$, r = 1..*m*, are generalized to their first common ancestor in the tree hierarchy $\mathcal{H}_{Kk}$ of the attribute K_k, let it be denoted by the node anc^k. In this case, $IL(cl) = |cl| \times \sum_{k=1}^{n} \frac{anc^k.max - anc^k.min}{root(H_{K_k}).max - root(H_{K_k}).min}$
  As it is usually the case, we assume that the internal nodes of predefined hierarchies for numerical attributes are also intervals (similarly to the nodes of our hierarchies dynamically generated, but without the extra information that we maintain when creating the hierarchy).

We needed to define the new *IL* measure for numerical attributes with hierarchies, adapted from existing ones ([15]), but different from the *IL* as defined for categorical quasi-identifier attributes, to correctly assess the information loss based on hierarchies that are not trees with all leaves on the same level (the norm for predefined value generalization hierarchies for both numerical and categorical attributes).

We did not define *IL* for quasi-identifiers consisting in a combination of numerical and categorical attributes, or consisting in numerical attributes some of which had hierarchies and some of which did not have hierarchies. However, such definitions can be easily produced by extending and combining the previous definitions.

Our reason to exclusively limit quasi-identifiers to homogeneous combinations of numerical attributes, with or without hierarchies, is that we aimed to isolate and study the impact on masked microdata quality of using different types of hierarchies in the anonymization process.

Property 1. The maximum information loss for a dataset $\mathcal{IM}$ over the quasi-identifier attribute set $\mathcal{K} = \{K_1, K_2,…, K_n\}$ is $maxIL(IM) = |IM| \times n$, where n = | $\mathcal{K}$| (the number of quasi-identifier attributes).

This *maxIL* corresponds to the case when all tuples in $\mathcal{IM}$ would have each quasi-identifier attribute generalized to the interval that covers all of its values in the set, or, respectively, generalized to the root value of its value generalization hierarchy. The proof for the previous statement follows easily from the definition of information loss for a cluster of tuples, if the cluster consisted in all the tuples in $\mathcal{IM}$.

The anonymization algorithm that we used in our experiments is the greedy *k*-anonymization algorithm presented in [15]. This algorithm works by creating groups/clusters of entities from $\mathcal{IM}$, of size *k* or more, that will be generalized to the same common values for all quasi-identifier attributes. These groups are created one at a time, starting from a seed tuple and absorbing one new tuple at a time, until the desired size of the group (i.e. *k*) is reached. A new tuple to include in a group is selected to minimize an objective function. The objective function in our case is the *IL* function; therefore, a new tuple is added to a cluster in construction *cl* if it produces a local minimum increase of *IL*(*cl*).

As it can be noticed, the *k*-anonymization algorithm we used is guided by the same measure that we use to assess the masked microdata quality.

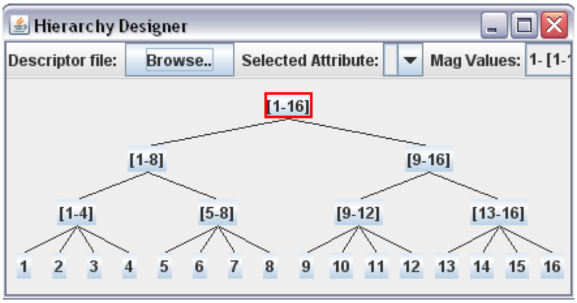
4 Experimental Results

To test the impact of numerical hierarchies on anonymization results, we used a subset of 1000 tuples from the adult dataset [17]. The quasi-identifier consisted in 3 numerical attributes: *age*, *education_num*, *hours_per_week*. As said before, we considered all of the quasi-identifiers to be numerical, as to avoid the categorical ones to impact in any way the anonymization process and the quality of the masked microdata.

We experimented with four settings for the quasi-identifier attributes:

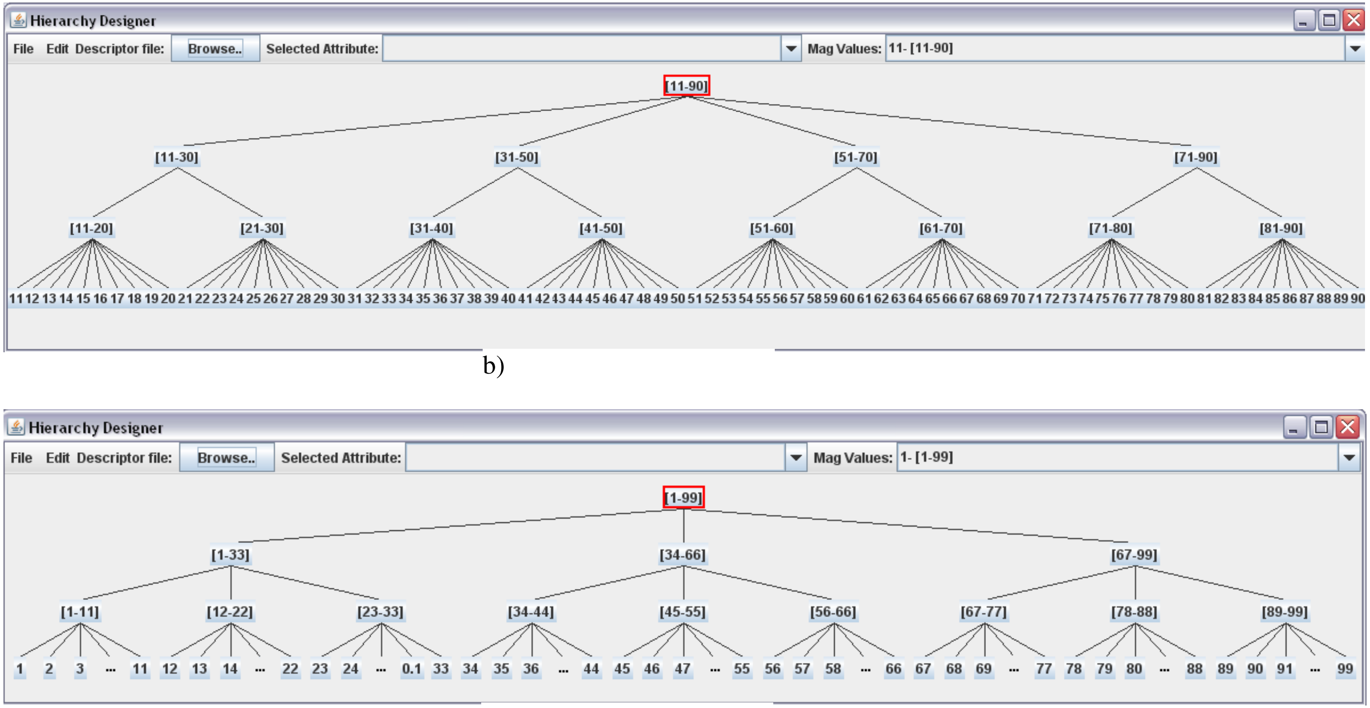
- Each one of them had predefined hierarchies – these hierarchies are shown in Fig. 4;
- Each one of them had a hierarchy dynamically created with our method;
- Each one of them had a hierarchy dynamically created, and then adjusted to have less levels; essentially, every second level of each hierarchy was removed;
- Quasi-identifier attributes did not have hierarchies (i.e. hierarchy-free generalization).

In each setting, we anonymized the microdata set using the same algorithm ([15]), for all possible *k* values in the range 3-20. Fig. 5 presents comparatively the percentage of information loss (actual *IL* reported to the maximum possible *IL*) for all four cases, for all *k* values considered in the experiments. It can be seen that the hierarchies produced on-the-fly behave reasonably well compared with predefined hierarchies that might be created without having a proper understanding of the data distribution in the current microdata set. As expected, the best results are obtained when hierarchy-free generalization is employed in the anonymization process. However, hierarchy-free generalization is not always an option, for example when a hierarchy is needed to define and apply generalization bounds for the anonymization procedure (such as for the constrained *k*-anonymous model). The quality of the masked microdata decreases when dynamically created hierarchies, from which every other level is cut, are used. But this is only to be expected, as roughly half of important information reflecting the data distribution is removed from the hierarchies.



a)

Fig. 4. Predefined hierarchies for quasi-identifier attributes *education_num* (a), *age* (b) , and *hours_per_week* (c).



Selected Attribute:

Mag Values: 11- [11-90]

b)

Selected Attribute:

Mag Values: 1- [1-99]

c)

Fig. 4. *(continued)*

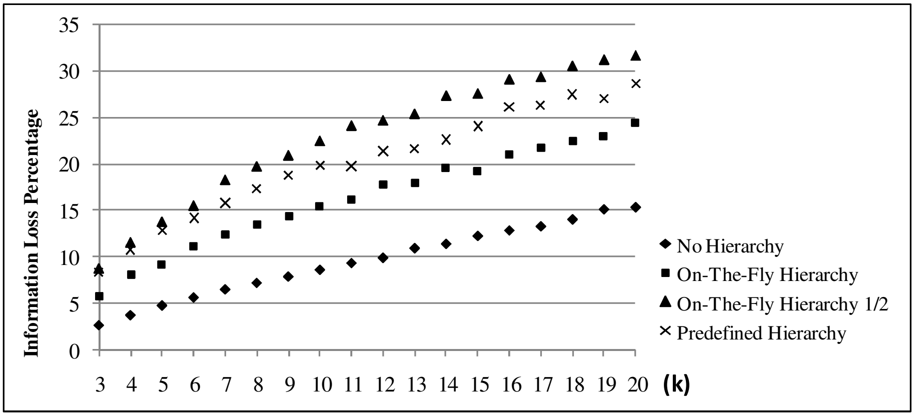


Fig. 5. *IL* percentage for *k*-anonymization with different hierarchy types

5 Conclusions

We presented in this paper a new method for dynamically creating hierarchies for numerical quasi-identifier attributes, to be used in microdata anonymization. The resulting hierarchies represent a valid alternative to predefined hierarchies, and their usage generally results in good quality masked microdata, with reasonable information loss. Such hierarchies can be easily produced on-the-fly when hierarchies are necessary, instead of forcing the user to artificially develop ones that might not reflect the properties of the data, therefore negatively impacting the quality of the masked microdata. A situation when hierarchies are needed for quantitative attributes is when masking the data to the constrained *k*-anonymity model; this anonymity model requires the use of hierarchies for defining maximal allowed generalization boundaries.

References

1. Samarati, P.: Protecting Respondents Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
2. Sweeney, L.: *k*-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems 10(5), 557–570 (2002)
3. Machanavajjhala, A., Gehrke, J., Kifer, D.: *L*-Diversity: Privacy beyond *K*-Anonymity. In: Proceedings of the International Conference on Data Engineering (IEEE ICDE 2006), p. 24 (2006)
4. Truta, T.M., Bindu, V.: Privacy Protection: *P*-Sensitive *K*-Anonymity Property. In: Proceedings of the Workshop on Privacy Data Management, with ICDE 2006, p. 94 (2006)
5. Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K.: (*α*, *k*)-Anonymity: An Enhanced k-Anonymity Model for Privacy-Preserving Data Publishing. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2006), pp. 754–759 (2006)

6. Li, N., Li, T., Venkatasubramanian, S.: *T*-Closeness: Privacy Beyond *k*-Anonymity and *l*-Diversity. In: Proceedings of the 23rd International Conference on Data Engineering (IEEE ICDE 2007), pp. 106–115 (2007)
7. Li, J., Tao, Y., Xiao, X.: Preservation of Proximity Privacy in Publishing Numerical Sensitive Data. In: Proceedings of the ACM SIGMOD, pp. 473–486 (2008)
8. Liu, J.Q., Wang, K.: On Optimal Anonymization for *l*+-Diversity. In: Proceedings of the International Conference on Data Engineering, IEEE ICDE 2010 (2010)
9. Wei, Q., Lu, Y., Lou, Q.: (τ,λ)-Uniqueness: Anonymity Management for Data Publication. In: Proceedings of the IEEE International Conference on Computer and Information Science (2008)
10. Sweeney, L.: Achieving *k*-Anonymity Privacy Protection Using Generalization and Suppression. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems 10(5), 571–588 (2002)
11. Iyengar, V.: Transforming Data to Satisfy Privacy Constraints. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 279–288 (2002)
12. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian Multidimensional *K*-Anonymity. In: Proceedings of the IEEE International Conference of Data Engineering, Atlanta, Georgia (2006)
13. Lunacek, M., Whitley, D., Ray, I.: A Crossover Operator for the *k*-Anonymity Problem. In: Proceedings of the GECCO Conference, pp. 1713–1720 (2006)
14. Miller, J., Campan, A., Truta, T.M.: Constrained *K*-Anonymity: Privacy with Generalization Boundaries. In: Jonker, W., Petković, M. (eds.) SDM 2008. LNCS, vol. 5159, Springer, Heidelberg (2008)
15. Byun, J.W., Kamra, A., Bertino, E., Li, N.: Efficient k-Anonymity using Clustering Techniques. CERIAS Technical Report 2006-10 (2006)
16. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2005)
17. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases, UC Irvine (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

eM^2: An Efficient Member Migration Algorithm for Ensuring k-Anonymity and Mitigating Information Loss

Phuong Huynh Van Quoc and Tran Khanh Dang

Faculty of Computer Science & Engineering, HCMUT
VNUHCM, Ho Chi Minh City, Vietnam
huynhvan.quocphuong@gnt.com.vn, khanh@cse.hcmut.edu.vn

Abstract. Privacy preservation (PP) has become an important issue in the information age to prevent expositions and abuses of personal information. This has attracted much research and k-anonymity is a well-known and promising model invented for PP. Based on the k-anonymity model, this paper introduces a novel and efficient member migration algorithm, called eM^2, to ensure k-anonymity and avoid information loss as much as possible, which is the crucial weakness of the model. In eM^2, we do not use the existing generalization and suppression technique. Instead we propose a member migration technique that inherits advantages and avoids disadvantages of existing k-anonymity-based techniques. Experimental results with real-world datasets show that eM^2 is superior to other k-anonymity algorithms by an order of magnitude.

Keywords: Privacy preservation; k-anonymity; information loss; member migration technique; eM^2.

1 Introduction

The vigorous development of information technology has brought many benefits such as the ability of storing, sharing, mining data by data mining techniques. However, this has a big obstacle of leaking out and abusing privacy. So, as a vital need, PP was born to undertake great responsibility of preserving privacy and maintaining data quality for data mining techniques. Concurrently focusing on privacy and data quality is a tradeoff in PP. In order to preserve privacy, identification attributes such as *id*, *name*, etc. must be removed. However, this does not ensure privacy since combinations of remaining attributes such as: *gender*, *birthday*, *postcode*, etc. can uniquely or nearly identify some individuals. Therefore, sensitive information of the individuals will be exposed. Such remaining attributes are called quasi-identifier attributes [1].

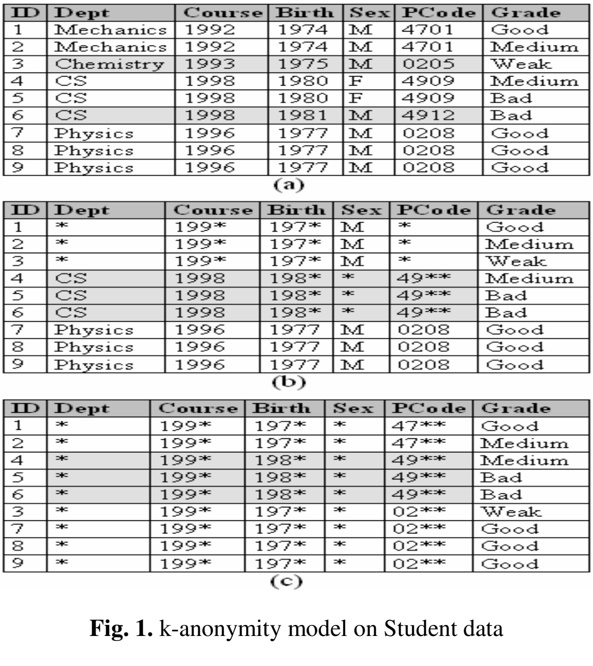
The k-anonymity model [1,2,3] is an approach to protect data from individual identification. The model requires that a tuple in a table, representing an individual with respect to quasi-identifier attributes, has to be identical to at least (k-1) other tuples. The larger the value of k, the better the protection of privacy. To obtain a k-anonymity model, there are various techniques classified into two types: *Generalization* and *Suppression*. The Generalization technique builds a hierarchical

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 26–40, 2010.
© Springer-Verlag Berlin Heidelberg 2010

system for values of an attribute value domain based on the generality of those values and replaces specific values with more general ones. This technique is classified into two generalization levels: attribute and cell levels. The attribute level replaces the current value domain of a quasi-identifier attribute with a more general one. For example, the domain of attribute *age* is mapped from years to 10-year intervals. The cell level just replaces current values of some essential cells with more general ones. Both levels obtain the k-anonymity model. However, the cell level has the advantage over the attribute level of losing less information as it does unnecessarily replace many general values, but it has the disadvantage of creating inconsistent values for attributes because general values co-exist with original ones. The attribute level has consistency of attribute values but loses much information as there are many changes to origin data. So, it easily falls into a too general state. Generally, the cell level is preferred for its upper hand of less information loss though it is more complicated than the attribute level, optimal k-anonymity by the cell level generalization is NP-hard [16]. Many proposed algorithms as shown in [6,7,8,11,12,14] have used this cell level generalization. The Suppression technique executes suppression on original data table. The Suppression can be applied to a single cell, to a whole tuple or column [5].

Fig. 1 illustrates a k-anonymity example for Student data with attributes Dept, Course, Birth, Sex, PCode regarded as quasi-identifier attributes. Table (a) is the original data. Tables (b) and (c) are 3-anonymity and 2-anonymity versions of table (a), where anonymization is achieved using the generalization at the cell level and attribute level, respectively.

In this paper, we propose eM^2 algorithm, a variant of M3AR (*a Member Migration technique for Maintaining Association Rules*) algorithm that we have introduced recently in [4]. M3AR tries to achieve the k-anonymity model while maintaining



ID	Dept	Course	Birth	Sex	PCode	Grade
1	Mechanics	1992	1974	M	4701	Good
2	Mechanics	1992	1974	M	4701	Medium
3	Chemistry	1993	1975	M	0205	Weak
4	CS	1998	1980	F	4909	Medium
5	CS	1998	1980	F	4909	Bad
6	CS	1998	1981	M	4912	Bad
7	Physics	1996	1977	M	0208	Good
8	Physics	1996	1977	M	0208	Good
9	Physics	1996	1977	M	0208	Good

(a)

ID	Dept	Course	Birth	Sex	PCode	Grade
1	*	199*	197*	M	*	Good
2	*	199*	197*	M	*	Medium
3	*	199*	197*	M	*	Weak
4	CS	1998	198*	*	49**	Medium
5	CS	1998	198*	*	49**	Bad
6	CS	1998	198*	*	49**	Bad
7	Physics	1996	1977	M	0208	Good
8	Physics	1996	1977	M	0208	Good
9	Physics	1996	1977	M	0208	Good

(b)

ID	Dept	Course	Birth	Sex	PCode	Grade
1	*	199*	197*	*	47**	Good
2	*	199*	197*	*	47**	Medium
4	*	199*	198*	*	49**	Medium
5	*	199*	198*	*	49**	Bad
6	*	199*	198*	*	49**	Bad
3	*	199*	197*	*	02**	Weak
7	*	199*	197*	*	02**	Good
8	*	199*	197*	*	02**	Good
9	*	199*	197*	*	02**	Good

(c)

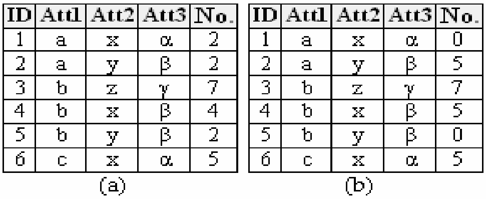
Fig. 1. k-anonymity model on Student data

Association Rules set of datasets as much as possible. Both eM^2 and M3AR algorithms use the novel Member Migration (MM) technique that has been proposed in [4] in order to get the k-anonymity model. While M3AR algorithm, however, represents a new approach to the PP [4] which preserves data privacy and maintains data quality toward a specific data mining technique such as Association Rule, Cluster, etc., eM^2 algorithm is to represent the traditional approach which also preserves data privacy (by obtaining k-anonymity) but it does not concentrate on any specific data mining technique. The traditional approach uses general metrics for its evaluation of data quality. Some general metrics introduced are Distortion [6], Information Loss (IL) [7], Uncertainty [8], CAVG [12]. The MM technique proposed with a main purpose is to make it suitable for maintaining Association Rule set. Therefore, it has been designed to possess advantages of less information loss of the cell level Generalization and consistent attribute values of the attribute level Generalization. Beside, the proposed technique, say MM, is also completely suitable for the traditional approaches to PP that we will analyze and confirm in this paper.

The rest of this paper is organized as follows. Section 2 discusses the Member Migration technique. Section 3 introduces crucial bases used in the eM^2 algorithm. Section 4 presents the eM^2 algorithm. Section 5 shows our experimental results with real-world datasets. Finally, section 6 give concluding remarks of the paper.

2 The Member Migration Technique

In this section, we will review the MM technique [4], which will be the cornerstone of the eM^2 algorithm. This technique firstly groups tuples in original data *D* into separate groups based on the identity of values on a quasi-identifier attribute set and then performs a MM operation between every two groups where there is at least one group having the number of tuples less than k. A group can perform a MM operation with one or more other groups. If a tuple *t* in a group *A* migrates to a group *B*, values of *t* have to change to ones of tuples in group *B* considering on a quasi-identifier attribute set. This technique is illustrated in Fig. 2.



ID	Att1	Att2	Att3	No.
1	a	x	α	2
2	a	y	β	2
3	b	z	γ	7
4	b	x	β	4
5	b	y	β	2
6	c	x	α	5

(a)

ID	Att1	Att2	Att3	No.
1	a	x	α	0
2	a	y	β	5
3	b	z	γ	7
4	b	x	β	5
5	b	y	β	0
6	c	x	α	5

(b)

Fig. 2. The MM technique to obtain a k-anonymity model

Table (a) is the result after grouping tuples in sample data into six separate groups based on the identity of values on a quasi-identifier attribute set {Att1, Att2, Att3}. Table (b) obtains a 5-anonymity model with four groups after applying three MM operations between groups as follows. One member (tuple) in group 5 migrated to group 4, values changed from (b,y,β) to (b,x,β). One member in group 5 migrated to group 2, values changed from (b,y,β) to (a,y,β). Two members in group 1 migrated to group 2, values changed from (a,x,α) to (a,y,β).

The MM technique only replaces necessary cell values by other ones in the current value domain, so it is easily to see that this technique includes advantages of Generalization techniques: less information loss as the cell level Generalization, consistent attribute values as the attribute level Generalization. Besides, it has its own advantages as follows: no difference between numerical and category attribute; no need to build hierarchies for attribute values based on generality; and finally, when a receiver gets data D' modified by this technique, he/she will sense that D' has never been modified.

Definition 1. A group is a subset of tuples (the number of tuples is greater than zero) of a table that has same values considering on a given quasi-identifier attribute set.

Definition 2. A group is k-unsafe if its number of tuples is fewer than k; otherwise, it is k-safe where k is a given anonymity degree.

Risk. Assume that the desired anonymity degree is k. A group that has the number of tuples m (m > 0) will be estimated risk through the function (cf. Appendix A):

$$F_R(m) = \begin{cases} 0 & when \quad m \geq k \\ 2k - m & when \quad 0 < m < k \end{cases} \tag{1}$$

Consider data D, after grouped, has a set of groups $G = \{g_1, g_2, \ldots, g_n\}$, $|g_i|$ is the number of tuples in the i^{th} group. Then the total risk $Risk_D$ of data D is

$$Risk_D = \sum_{i=1}^{n} F_R(|g_i|) \tag{2}$$

Observation 1. Assume Risk$_{D'}$ is the risk of data D'. Risk$_{D'}$ = 0 if and only if D' has achieved a k-anonymity model.

Proof is simple, using the disproof method.

Definition 3. Let $mgrt(g_i \rightarrow g_j){:}T$ be a Member Migration operation from g_i to g_j $(i \neq j, T \subseteq g_i)$ if values of all tuples in T are changed to values of tuples in g_j considering a given quasi-identifier attribute set.

Let $mgrt(g_i,g_j)$ be possible migrant directions between two groups g_i and g_j. Then, there exists two separate migrant directions, those are from g_i to g_j with symbol $mgrt(g_i \rightarrow g_j)$ and from g_j to g_i with symbol $mrgt(g_i \leftarrow g_j)$. So $mgrt(g_i,g_j) = mgrt(g_i \rightarrow g_j) \cup mgrt(g_i \leftarrow g_j)$. And with two groups g_i, g_j $(i \neq j)$, it is easy to see that there are at least two MM operations that can be performed between them.

Definition 4. A Member Migration operation $mgrt(g_i \rightarrow g_j){:}T$ is "*valuable*" when the risk of data is decreased after performing that Member Migration operation.

When performing a MM operation $mgrt(g_i \rightarrow g_j){:}T$, there are only changes on risks of two groups g_i and g_j. Therefore, risk reduction of all data is equal to the sum of risk reductions in two group g_i and g_j after performing that MM operation.

Theorem 1. If a Member Migration operation $mgrt(g_i \rightarrow g_j):T$ is "*valuable*" then the number of k-unsafe groups in set $\{g_i, g_j\}$ after this operation can not exceed one.

Proof. The case when two groups g_i and g_j are both k-safe is obviously true. Moreover, this case can never happen. Consider the cases when there is at least one k-unsafe group, using the disproof method. Assume after performing a "***valuable***" MM operation on g_i and $g_{j,}$ we still have two k-unsafe groups. We have two cases as follows:

- Case 1: When both g_i and g_j are k-unsafe, the total risk of two groups before migrating is $Risk_{before} = 4k\text{-}|g_i|\text{-}|g_j|$. Assume l is the number of migrant tuples. Without loss of generality, assume the migrant direction is $mgrt(g_i \rightarrow g_j)$. Since the number of k-unsafe groups after migrating is two, the risk after migrating is $Risk_{after} = 4k–(|g_i|\text{-}l)–(|g_j|+l) = 4k\text{-}|g_i|\text{-}|g_j| = Risk_{before.}$ However, this is a "***valuable***" MM operation, so we have a contradiction.
- Case 2: One k-safe group and one k-unsafe one. Assume g_i is k-unsafe and g_j is k-safe. $Risk_{before} = 2k\text{-}|g_i|$. Because we have two k-unsafe groups after the MM operation, it is obvious that the migrant direction is $mgrt(g_i \leftarrow g_j)$ with the number of tuples is l and satisfies two conditions: $0<|g_i|+l<k$ and $0<|g_j|\text{-}l<k$. So we have $0<|g_i|+|g_j|<2k$ (*). Besides $Risk_{after} = 4k\text{-}|g_i|\text{-}|g_j| < Risk_{before} = 2k\text{-}|g_i|$, that means $|g_j|>2k$ (**). From (*) and (**), we have a contradiction.

From two cases above, the theorem is proven.

3 Preliminaries

Let the MM technique be open, we only define its bases. The technique do not define how to choose two groups for executing a MM operation, or how to determine the migrant direction, how many tuples are migrated, or which tuples belonging to a group will be chosen for migrating. The technique leaves all these questions to algorithms using it in order to have flexible algorithms and a big number of variants. This section will present crucial bases, giving answers to those questions, in order to adapt the MM technique for the eM^2 algorithm.

3.1 Policies

This subsection presents three policies that are basic to operation mechanism of the eM^2 algorithm. Give a group g, *original* tuples of g is all tuples that g has when it has never executed a MM operation with any other groups. Because a group can receive from or give to other groups some tuples, let *origin*(g) be all *remaining original* tuples of g and *foreign*(g) be all tuples that g has received from other groups. Those policies are as follows:

1. A k-unsafe group once has received tuple(s), it can only continue receiving tuple(s); otherwise, when its tuple(s) migrate to another group, it can only continue giving its tuple(s) to other groups. Note that this policy does not apply to k-safe groups.

2. Given two groups g_i, g_j. Assume we have $mgrt(g_i \rightarrow g_j){:}T$ then $T \subseteq origin(g_i)$. Because all tuples in $origin(g_i)$ are identical, T can be chosen randomly or from the first $|T|$ tuples in $origin(g_i)$.
3. Consider two groups g_i, g_j. Since there is at least one k-unsafe group, assume that g_i is a k-unsafe group. The number of migrant tuples (*mgrtN*) is determined as follows:
   - Case 1: g_j is a k-unsafe group. If $mgrt(g_i \rightarrow g_j)$ then $mgrtN = Min(|g_i|, k\text{-}|g_j|)$. If $mgrt(g_i \leftarrow g_j)$ then $mgrtN = Min(|g_j|, k\text{-}|g_i|)$.
   - Case 2: g_j is a k-safe group. If $mgrt(g_i \rightarrow g_j)$ then $mgrtN = |g_i|$. If $mgrt(g_i \leftarrow g_j)$ then the $mgrtN \leq Min(k\text{-}|g_i|,\ |g_j|\text{-}k,\ |origin(g_j)|)$, when $Min(|g_j|\text{-}k,\ |origin(g_j)|)=0$ then $mgrt(g_i \leftarrow g_j)$ is impossible.

Besides determining the number of migrant tuples, policy 3 guarantees every MM operation be "***valuable***".

3.2 Metrics

In this subsection, we present three metrics Distortion, IL and Uncertainty used respectively in KACA [6], OKA [7] and Bottom-Up [8] algorithm. Because modified data of MM technique is different to that of Generalization techniques, the formulas of this metrics are adapted to MM technique that will be proposed in the subsection. Besides this three metrics, we also use the CAVG metric employed in [6,8,11,12].

Distortion Metric

All allowable values of an attribute form a hierarchical value tree. Each value is represented as a node in the tree, and a node has a number of child nodes corresponding to its more specific values. Let t_1 and t_2 be two tuples. $t_{1,2}$ is the closest common generalization of t_1 and t_2 for all i. The value of the closest common generalization $t_{1,2}$ is calculated as follows:

$$v_{1,2}^i = \begin{cases} v_1^i & \mathit{if}\ v_1^i = v_2^i \\ \mathit{the\ value\ of\ the\ closest\ common\ ancestor} & \mathit{otherwise} \end{cases} \tag{3}$$

$$\mathit{where},\ v_1^i, v_2^i\ \mathit{and}\ v_{1,2}^i\ \mathit{are\ the\ values\ of\ the}\ i-th\ \mathit{attribute\ in}\ t_1, t_1\ \mathit{and}\ t_{1,2}$$

Let h be the height of a domain hierarchy, and let levels $1, 2, \ldots, h-1, h$ be the domain levels from the most general to most specific, respectively. Let the weight between domain level i and $i-1$ be predefined, denoted by $w_{i,i-1}$, where $2 \leq i \leq h$. When a cell is generalized from level p to level q, where $p>q$, the weighted hierarchical distance of this generalization is defined as:

$$WHD(p,q) = \frac{\sum_{i=q+1}^{p} w_{i,i-1}}{\sum_{i=2}^{h} w_{i,i-1}} \tag{4}$$

$$\mathit{where}\ w_{i,i-1} = 1/(i-1)^{\beta}\ \mathit{with}\ 2 \leq i \leq h, \beta\ \mathit{is\ a\ real\ number} \geq 1$$

Let $t = \{v_1,v_2,...,v_m\}$ be a tuple and $t' = \{v'_1,v'_2,...,v'_m\}$ be a generalized tuple of t where m is the number of attributes in the quasi-identifier. Let $level(v_j)$ be the domain level of v_j in an attribute hierarchy. The Distortion of this generalization is defined as:

$$Distortion(t,t') = \sum_{j=1}^{m} WHD(level(v_j), level(v'_j)) \tag{5}$$

Let D' be generalized from table D, t_i be the *i-th* tuple in D and t_i' be the *i-th* tuple in D'. The Distortion of this generalization is defined as:

$$Distortion(D,D') = \sum_{i=1}^{|D|} Distortion(t_i, t'_i) \tag{6}$$

However, if D' is modified by MM technique then every tuple t_i' in D' will be an identical tuple or a non-generalized tuple of t_i in D. Therefore, if using (6) then $Distortion(D,D') = 0$, this is not right. The right formula is defined as:

$$\begin{aligned} &Distortion(D,D') = \sum_{i=1}^{|D|} (\, Distortion(t_i, t_i^*) + Distortion(t'_i, t_i^*)\,) \\ &\textit{where } t_i^* \textit{ is the closest common genrealization of } t_i, t'_i \end{aligned} \tag{7}$$

Let g_1 be a group containing $|g_1|$ identical tuples t_1 and g_2 be a group containing $|g_2|$ identical tuples t_2. $t_{1,2}$ is the closest common generalization of t_1 and t_2. The distance between two groups in KACA is defined as:

$$Distortion(g_1, g_2) = |g_1| \times Distortion(t_1, t_{1,2}) + |g_2| \times Distortion(t_2, t_{1,2}) \tag{8}$$

However, with the MM technique, (8) is no longer right because there is only some tuples in g_1 or g_2 modified with respect to given quasi-identifier attribute set. The right formula for the MM technique is defined as:

$$\begin{aligned} Distortion(mgrt(g_1 \rightarrow g_2):T) &= |T| \times (\, Distortion(t_1, t_{1,2}) + Distortion(t_2, t_{1,2})\,) \\ Distortion(mgrt(g_1 \leftarrow g_2):T') &= |T'| \times (\, Distortion(t_1, t_{1,2}) + Distortion(t_2, t_{1,2})\,) \end{aligned} \tag{9}$$

Uncertainty Metric

Let $(A_1,..., A_n)$ be quasi-identifier attributes, give a numerical attribute A_i. Suppose a tuple $t=(..., x_i, ...)$ is generalized to tuple $t'=(..., [y_i, z_i], ...)$ such that $y_i \le x_i \le z_i$ $(1 \le i \le n)$. On attribute A_i, the normalized certainty penalty is defined as:

$$NCP_{A_i} = \frac{z_i - y_i}{|A_i|} \quad where\ |A_i| = \max_{t\in T}\{t.A_i\} - \min_{t\in T}\{t.A_i\} \tag{10}$$

Give a categorical attribute A_i. Let $v_1, ..., v_n$ be a set of leaf nodes in a hierarchy tree of A_i. Let u be the node in the hierarchy tree such that u is an ancestor of $v_1, ..., v_n$ and u does not have any descendant that is still an ancestor of $v_1, ..., v_n$. *u* is called closest common ancestor of $v_1, ..., v_n$, denoted by $ancestor(v_1, ..., v_n)$. The number of leaf nodes that are descendants of u is called the size of u, denoted by $size(u)$.

Suppose a tuple t has value v on a categorical attribute A_i. When it is generalized in anonymization, the value will be replaced by $ancestor(v_1, ..., v_n)$, where $v_1, ..., v_n$ are the values of tuples on the attribute in the same generalized group. The normalized certainty penalty of *t* is defined as:

$$NCP_{A_i}(t) = \frac{size(u)}{|A_i|} \qquad (11)$$
$$where\ |A_i|\ is\ the\ number\ of\ distinct\ values\ wrt.\ attribute\ A_i$$

Let *D* be a table, *D* consists of both numerical and categorical attributes, *D'* be a generalized table of *D*. The total weighted normalized certainty penalty of *D'* is:

$$NCP(D') = \sum\nolimits_{t' \in T'} \sum\nolimits_{i=1}^{n} (w_i.NCP_{A_i}(t')) \qquad (12)$$

Depends on whether A_i is a numerical or categorical attribute, $NCP_{A_i}(t')$ will be computed by (10) or (11); w_i is weight of attribute A_i. (12) is suitable for generalization technique. But with the MM technique, (12) is no longer right because *NCP*(*D'*) will be zero. For proper with the MM technique, $NCP_{A_i}(t')$ in (12) is adapted as:

$$NCP(D') = \sum\nolimits_{t' \in T', t \in T} \sum\nolimits_{i=1}^{n} (w_i.NCP_{A_i}(t',t))$$
$$NCP_{A_i}(t',t) = \begin{cases} \dfrac{|t.A_i - t'.A_i|}{|A_i|} & A_i\ is\ a\ numeric\ attribute \\ \dfrac{size(ancestor(t.A_i,\ t'.A_i))}{|A_i|} & A_i\ is\ a\ categorical\ attribute \end{cases} \qquad (13)$$
$$where\ t'\ in\ D'\ corresponds\ with\ t\ in\ D.\ D'\ is\ a\ version\ of\ D\ modified\ by\ MM\ technique$$

Let g_1 be a group containing $|g_1|$ identical tuples t_1 and g_2 be a group containing $|g_2|$ identical tuples t_2. The total normalized certainty penalty of a MM operation is defined as:

$$\begin{aligned} &NCP(mgrt(g_1 \rightarrow g_2):T) = |T|.\sum\nolimits_{i=1}^{n}(w_i.NCP_{A_i}(t_1,t_2)) \\ &NCP(mgrt(g_1 \leftarrow g_2):T') = |T'|.\sum\nolimits_{i=1}^{n}(w_i.NCP_{A_i}(t_1,t_2)) \\ &where\ A_i\ is\ i^{th}\ attribute \end{aligned} \qquad (14)$$

IL Metric

Let *D* denote a set of records which is described by m numerical quasi-identifier attributes N_1, ...,N_m and *q* categorical quasi-identifier attributes C_i, ..., C_q. Let **P** = {P_1,...,P_k} be a partitioning of *D*, namely, $\bigcup_{i \in [1...k]} P_i = D,\ P_i \cap P_j = \Phi\ (i \neq j)$. Each categorical attribute C_i is associated with a taxonomy tree T_{Ci} that is used to generalize the values of this attribute. With a partition $P \subset$ **P**, let $\breve{N}_i(P), \breve{N}_i(P), \overline{N}_i(P)$ respectively denote the max, min, and average values of the tuples in *P* with respect to the numerical attribute N_i. Let $\breve{C}_i(P)$ be the set of values of the records in *P* with respect to the categorical attribute C_i. Let $T_{Ci}(P)$ be the maximal subtree of T_{Ci} rooted at the lowest common ancestor of values of $\breve{C}_i(P)$. Then the *diversity* of *P*, denoted by $\dddot{D}(P)$, is defined as:

$$\dddot{D}(P) = \sum_{i\in[1,m]} \frac{\widehat{N}_i(P) - \breve{N}_i(P)}{\widehat{N}_i(D) - \breve{N}_i(D)} + \sum_{i\in[1,q]} \frac{H(T_{Ci}(P))}{H(T_{Ci})} \tag{15}$$

$$\textit{where } H(T) \textit{ is the height of tree } T$$

Let *r'*, *r** be two records, then the distance between *r'* and *r** is defined as the *diversity* of the set $\{r', r^*\}$, i.e., $\dddot{D}(\{r', r^*\})$. To anonymize the records in *P* means to generalize these records to the same values with respect to each quasi-identifier attribute. The amount of information loss occurred by such a process, denoted as *L*(*P*), is defined as:

$$L(P) = |P|.\dddot{D}(P) \quad \textit{where } |P| \textit{ is the number of records in } P \tag{16}$$

Therefore, the total information loss of *D* is defined as:

$$L(D) = \sum\nolimits_{i\in[1,k]} |P_i|.\dddot{D}(P_i) \quad \textit{where } |P_i| \textit{ is the number of records in } P_i \tag{17}$$

Let *D'* be a k-anonymity version of *D* modified by the MM technique. Assume *D'* has a set of groups *G'*={g'_1, …, g'_m}. If we apply (17) to *D'*, it means that we apply (16) for each *g'* in *G'*, this is not right because there is only some tuples in *g'* modified with respect to quasi-identifier attributes, and all remaining tuples in *g'* do not have any modification. In order to satisfy the MM technique, *L*(*D'*) is adapted as:

$$L(D') = \sum\nolimits_{t\in D, t'\in D'} \dddot{D}(\{t, t'\}) \quad \textit{where } t' \textit{ corresponds with } t \textit{ in } D \tag{18}$$

Let g_1 be a group containing $|g_1|$ identical tuples t_1 and g_2 be a group containing $|g_2|$ identical tuples t_2. The total information loss of a MM operation is defined as:

$$\begin{aligned} L(mgrt(g_1 \rightarrow g_2) : T) &= |T|.\dddot{D}(\{t_1, t_2\}) \\ L(mgrt(g_1 \leftarrow g_2) : T') &= |T'|.\dddot{D}(\{t_1, t_2\}) \end{aligned} \tag{19}$$

The eM^2 algorithm uses all the three metrics: Distortion, Uncertainty and IL. So, in order to be convenient, we use a common symbol *DIF* to denote all this three metrics. Give two groups g_i, g_j, the number of migrant tuples belonging to each migrant direction is determined by policy 3. From (9), (14), or (19), we can see that, the chosen migrant direction is a migrant direction with less number of migrant tuples than the others.

CAVG Metric

The metric is defined as the following:

$$CAVG = \left(\frac{total\ records}{total\ groups}\right) / k \tag{20}$$

The quality of k-anonymity is measured by the average size of groups produced, an objective is to reduce the normalized average group size. It is mathematically sound and not intuitive to reflect changes being made to *D*. However, the metric reflects that the more its value approaches one the more approximation among sizes of groups is.

4 The Proposed eM^2 Algorithm

The eM^2 algorithm is divided into two processing stages. First is the Initialization stage: partition tuples of *D* into groups, classify those groups into k-safe and k-unsafe groups. Time complexity of this stage is $O(|D|)$ as it linearly depends on the size of data *D*. Second is the Process stage: in each while loop, if *SelG* is *null* then randomly select a group in *UG* to assign to *SelG* and find a group *g* in the rest groups so that *DIF*(*mgrt*(*SelG*,*g*):*T*) is minimized. If a group *g* can not be found then the algorithm exits from the while loop. Otherwise, perform the MM operation between *SelG* and *g*. Note that there is at most one k-unsafe group in {*SelG*, *g*} after the MM operation (cf. Theorem 1), so if there exists the k-unsafe group, it will be assigned to reference *SelG* and processed in the next loop, otherwise *SelG* = *null* so that a new k-unsafe group in *UG* is selected randomly and processed in next loop. When the while loop ends, if *SelG* is not *null*, it will be dispersed by ***Disperse*** function. Time complexity of Process stage is mainly in while loop because processing time of ***Disperse*** function is so much fewer than that of while loop. Moreover, *SelG* is almost *null* after the loop. It is easy to conclude that the time complexity of this stage is $O(|UG|*|G|) \approx O(|G|^2)$. So it is also the time complexity of the algorithm.

eM^2 Algorithm

```
Input:      The original data D; parameter k;
Output:     D' achieved k-anonymity model;
Variable:   G=∅, SG=∅, UG=∅ are three sets of groups; SelG=null;
Begin
   Initialization:
   1. G = set of groups obtained from D;
   2. Divide G into set of k-safe groups SG and
      set of k-unsafe groups UG;
   Process:
  while(|UG|>0) or (SelG!=null){
      if(SelG==null){
         SelG=randomly select a group in UG; UG=(UG\SelG);
      }
      Find in UG, SG a group g so that DIF(mgrt(SelG,g):T) is minimized;
      If g is not found then the while loop is broken;
      if(g∈ UG){
         UG=UG\g;
         if(mgrt(SelG→g):T){
            Migrate t∈ T to g;
            if(|SelG|=0){
               if(|g|<k) SelG=g;
               else{ SelG=null; SG=(SG∪g);}
            }else SG=(SG∪g);
         }else{ /* mgrt(SelG←g):T */
            Migrate t∈ T to SelG;
            if(|g|=0){
```

```
17.             if(|SelG|=k){ SG=(SG∪SelG); SelG=null;}
18.          }else{ SG=(SG∪SelG); SelG=g;}
          }
19.    }else{ /* g∈SG */
20.       if(mgrt(SelG→g):T){ Migrate t∈T to g; SelG=null;}
21.       else{ /* mgrt(SelG←g):T */
22.          Migrate t∈T to SelG;
23.          if(|SelG|=k){ SG=(SG∪SelG); SelG=null;}
          }
       }
   }/*end while*/
24.if(SelG!=null) Disperse(SelG,SG);
End

Disperse(SelG,SG)
Begin
1.Return all t∈foreign(SelG) to its initial group.
  Then we have a set of groups G; G=G∪SelG.
2.For each g in G {
3.   if(|g|<k){
4.       Find a group g' in SG so that DIF(mgrt(g→g'):g) is minimized;
5.       Migrate t∈g into g';
6.   }
  }
End
```

Observation 2. After the while loop in the eM^2 algorithm finishes, if Disperse function is called then the k-unsafe group processed by this function is the final and only k-unsafe one.

The reason for existing a k-unsafe group which can not perform a MM operation with any other groups is risen from case 2 in policy 3.

Proof. Assume *SelG* is a k-unsafe group that is processed by **Disperse** function. It means that *SelG* can not perform a MM operation with any remaining groups. So we have three conditions as follows:

1. *SelG* must have received some tuples from other groups. Because if *SelG* has never received tuple(s) from other group(s), it can give its tuple(s) to other groups. This means that *SelG* can always perform a MM operation with other groups. Now, *SelG* can only receive some tuples (policy 1).
2. Every k-safe group *g* in set of k-safe groups (SG) must satisfy Min($|g|$-k, $|origin(g)|$) = 0 so that *SelG* can not perform a MM operation $mgrt(SelG \leftarrow g){:}T$ with any k-safe group *g* in SG.
3. There does not exist any k-unsafe group apart from *SelG*. Because if there exists a k-unsafe group *g* in set of k-unsafe groups (UG) then *g* can give some tuples to *SelG*. It means that, we still find a group *g* which can perform a MM operation $mgrt(SelG \leftarrow g){:}T$ with *SelG*.

Only with condition 3, we can see that *SelG* is the final and only k-unsafe group. And line 5 in the eM^2 algorithm says that if group *g* is not found then *SelG* is the final and only k-unsafe one. Therefore, the while loop will be exited and all tuples in *SelG* will be processed by the **Disperser** function.

5 Experiments

This section presents some experiments using the real world database Adult [13] used popularly in experiments of k-anonymity model to verify the performance of our eM^2 algorithm in both execution time and data quality by comparing with three algorithms KACA, OKA and Bottom-UP. In their own papers, these three algorithms show that they have advantages of data quality and execution time. All algorithms are implemented using VB.Net and executed on a Core (MT) 2 Duo CPU 2.0 GHz with 1 GB physical memory and Windows XP OS. The database Adult has 6 numerical attributes and 8 categorical attributes. It leaves 30162 records after removing the records with missing values. In our experiments, we retain only six-attribute set {*age*, *gender*, *marital*, *country*, *race*, *edu*} considered as quasi-identifier attribute set. With Distortion metric, all six attributes are treated as categorical attribute and WHD in (4) uses $w_{i,i\text{-}1}=1/(i\text{-}1)$, it means that $\beta=1$. With IL and Uncertainty metrics, *age* attribute is treated as a numerical attribute and five remaining attributes are treated as categorical attributes. Table 1 describes the features of these six attributes.

Table 1. Features of Quasi-identifier Attributes

Attribute	Type	# of Values	Height
Age	Numeric	74	4
Gender	Categorical	2	2
Marital	Categorical	7	3
Country	Categorical	41	3
Race	Categorical	5	2
Edu	Categorical	16	4

In (12), (13) and (14) we set $w_i=1$ for all attributes. The achieved result is the average of three times executing the algorithms with each k.

Fig. 3 shows comparisons between eM^2 and KACA on Distortion, CAVG metrics and execution time. With Distortion, eM^2 gets approximately with KACA. But CAVG and execution time of eM^2 is better than those of KACA.

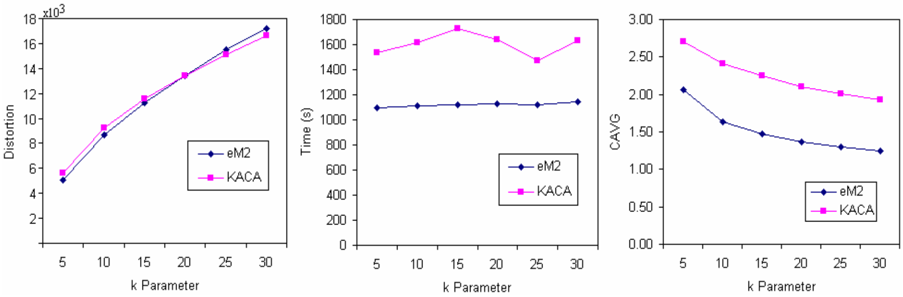


Fig. 3. Compare between eM^2 and KACA

Fig. 4 shows comparisons between eM^2 and Bottom-Up on Uncertainty, CAVG metrics and execution time. With Uncertainty and execution time, eM^2 gets much better than OKA. And with CAVG, eM^2 gets better than Bottom-Up but there are not much differences.

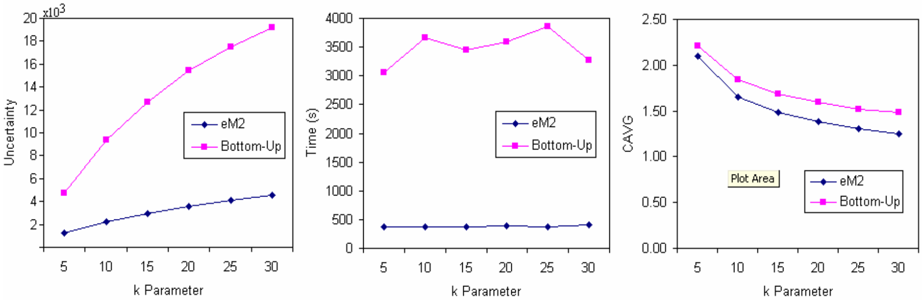


Fig. 4. Compare between eM^2 and Bottom-Up

Fig. 5 shows comparisons between eM^2 and OKA on IL, CAVG metrics and execution time. With IL, eM^2 gets much better than OKA. Execution time of OKA is long (4010 seconds at k=5), but then it quickly reduces when k increases (482 seconds at k=30). With CAVG, OKA has a difference with three remaining algorithms; CAVG of eM^2, KACA and Bottom-Up reduces when k increases, but that of OKA increases when k increases.

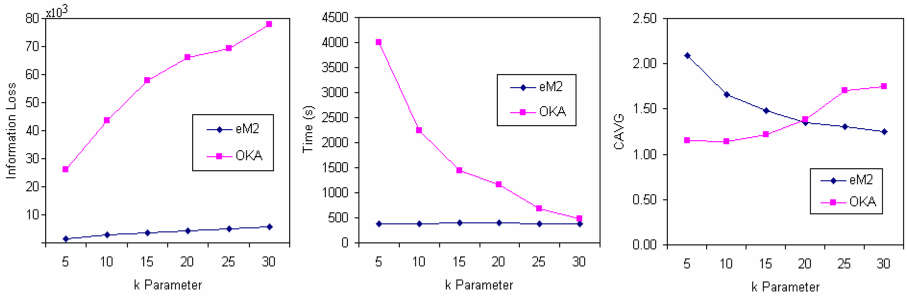


Fig. 5. Compare between eM^2 and OKA

Execution time of eM^2 when using Distortion is higher than that of eM^2 when using Uncertainty and IL though with the same eM^2 algorithm. It means that, computation complexity of Distortion metric is higher than that of Uncertainty and IL metrics.

6 Conclusions

In this paper, our main contribution includes twofold: (i) Introduce an adapted Member Migration (MM) technique, proposed recently by our previous research [4], that includes the advantages of the Generalization techniques in the k-anonymity model and has its own unique characteristics for ensuring k-anonymity and mitigating information loss in general-purpose datasets; (ii) Propose the eM^2 algorithm, based on the MM technique, that has advantages of data quality and execution time over existing state-of-the-art techniques while obtaining k-anonymity. By proposing adapted formulas of metrics (cf. subsection 3.2) and carrying out intensive experiments, we have shown that the MM technique and eM^2 algorithm is completely suitable for traditional approach into privacy preservation.

Besides the k-anonymity model, there is a variety of its variants proposed to preserve data out of individual re-identification such as: l-Diversity [15], t-Closeness [9], (α, k)-Anonymity [10]. Extending the eM^2 algorithm to these models will be of our great interests in the future. Furthermore, developing or varying our algorithm to deal with other problems in the area of privacy preservation will be also among our future research activities.

Acknowledgment

The work is a next development step from [4] which was contributed by helpful opinions of Dr. Vo Thi Ngoc Chau, Dr. Tran Van Hoai (CSE/HCMUT/VNUHCM) and Dr. Nguyen Duc Cuong (IU/VNUHCM). Also, we would like to thank all people in ASIS Lab that help us accomplish this work.

References

1. Samarati, P.: Protecting Respondent's Privacy in Microdata Release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
2. Sweeney, L.: Achieving k-Anonymity Privacy Protection using Generalization and Suppression. International Journal on Uncertain. Fuzz 10(6), 571–588 (2002)
3. Sweeney, L.: K-anonymity A Model for Protecting Privacy. International Journal on Uncertain. Fuzz 10(5), 557–570 (2002)
4. Phuong, H.V.Q., Khanh, D.T., Chau, V.T.N.: A Privacy Preserving Algorithm that Maintains Association Rules. In: Proc. of International Conference on Advanced Computing and Application (ACOMP), HCM City, Vietnam, pp. 180–189 (March 2010)
5. Aggarwal, C.C., Yu, P.S.: Privacy-Preserving Data Mining Models and Algorithms. Springer, Heidelberg (2008)
6. Li, J.Y., Wong, R.C.W., et al.: Anonymisation by Local Recoding in Data with Attribute Hierarchical Taxonomies. IEEE Transactions on Knowledge and Data Engineering 20(9), 1187–1194 (2008)
7. Lin, J.-L., Wei, M.C.: An Effcient Clustering Method for k-Anonymization. In: Proc. of the International Workshop on Privacy and Anonymity in the Information Society (2008)
8. Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A.: Utility-Based Anonymization Using Local Recoding. In: SIGKDD, pp. 785–790 (2006)
9. Li, N., Li, T., et al.: t-Closeness: Privacy beyond k-Anonymity and l-Diversity. In: Proc. of the 23rd IEEE International Conference on Data Engineering, pp. 106–115 (2007)

10. Jian-min, H., Hui-qun, Y., Juan, Y., Ting-ting, C.: A Complete (α,k)-Anonymity Model for Sensitive Values Individuation Preservation. In: ISECS, pp. 318–323. IEEE, Los Alamitos (2008)
11. Ye, Y., Deng, Q., et al.: BSGI: An Effective Algorithm towards Stronger l-Diversity. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 19–32. Springer, Heidelberg (2008)
12. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian Multidimensional k-Anonymity. In: Proc. of the 22nd IEEE International Conference on Data Engineering (2006)
13. U.C. Irvin Machine Learning Repository, `http://archive.ics.uci.edu/ml/` (accessed 2009)
14. Juan, Y., Zanzhu, X., et al.: TopDown-KACA: An Efficient Local-Recoding Algorithm for k-Anonymity. In: Proc. of the IEEE Int. Conference on Granular Computing, pp. 727–732 (2009)
15. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: Proc. of the 22nd IEEE Int. Conference on Data Engineering, ICDE 2006 (2006)
16. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: PODS 2004 (2004)

Appendix A: Risk Function $F_R(m)=2k-m$

The reason for choosing $F_R(m) = C - m$ ($C = 2k$, $0 < m < k$, $C \in \mathrm{N}$) is for the satisfaction of theorem 1. In the proof of theorem 1, we can see that case 1 does not depend on *C.* So we only consider case 2 (g_i is a k-unsafe group and g_j is k-safe). We have $Risk_{before} = C$-$|g_i|$. Because we have two k-unsafe groups after the MM operation, it is obvious that the migrant direction is $g_i \leftarrow g_j$ with the number of tuples is l and satisfies two conditions: $2 \le |g_i|+l \le k$-1 and $1 \le |g_j|$-$l \le k$-1. So we have $3 \le |g_i|+|g_j| \le 2k$ - 2 (*). Besides $Risk_{after} = 2C$-$|g_i|$-$|g_j| < Risk_{before} = C$-$|g_i|$, that means $|g_j|>C \equiv |g_j| \ge C$ +1. So we have $|g_i|+|g_j| \ge C$+2 (**). From (*) and (**) we have $C+2 \le 2k - 2 \equiv C \le 2k - 4$ (*'). If (*') is true, it means that theorem 1 is false. Therefore, for theorem 1 to be true, (*') must be false. In other words, C must satisfy $C \ge 2k - 3$. Finally, we choose C=2k and $F_R(m) = 2k - m$ ($0 < m < k$).

Constrained Anonymization of Production Data: A Constraint Satisfaction Problem Approach

Ran Yahalom[1], Erez Shmueli[1,2], and Tomer Zrihen[1,2]

[1] Deutsche Telekom Laboratories
[2] Department of Information Systems Engineering,
Ben-Gurion University P.O.B. 653, Beer Sheva 84105, Israel

Abstract. The use of production data which contains sensitive information in application testing requires that the production data be anonymized first. The task of anonymizing production data becomes difficult since it usually consists of constraints which must also be satisfied in the anonymized data. We propose a novel approach to anonymize constrained production data based on the concept of constraint satisfaction problems. Due to the generality of the constraint satisfaction framework, our approach can support a wide variety of mandatory integrity constraints as well as constraints which ensure the similarity of the anonymized data to the production data. Our approach decomposes the constrained anonymization problem into independent sub-problems which can be represented and solved as constraint satisfaction problems (CSPs). Since production databases may contain many records that are associated by vertical constraints, the resulting CSPs may become very large. Such CSPs are further decomposed into dependant sub-problems that are solved iteratively by applying local modifications to the production data. Simulations on synthetic production databases demonstrate the feasibility of our method.

1 Introduction

Testing is a cardinal stage in the life-cycle of every information system. It cannot be performed without data which is usually stored in databases. Clearly, the simplest way to provide this data to a test environment is to copy it from the production environment. However, production data often contains sensitive information which should not be exposed in a non-privileged test environment (such as an external, sometimes even foreign, testing group). Thus, the production data must be anonymized before it is copied to the test environment so that sensitive information is masked from the end-user. However, the task of anonymizing production data is non-trivial since it must also preserve certain characteristics (rules) of the original production data for the anonymized data to be useful for testing. Failing to do so may compromise the testing process. We refer to this problem as the *Constrained Anonymization* problem.

Available anonymization tools (e.g. [12,10,11,13]) provide a means of enforcing common rules, usually a set of predefined integrity rules (such as identity and reference rules)[6] or simple statistical aggregate rules (such as maintaining the average value of some field). However, application-specific rules are enforced via anonymization routines that are developed ad hoc by the user. Although these tools offer powerful scripting

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 41–53, 2010.
© Springer-Verlag Berlin Heidelberg 2010

capabilities which can be used with such routines, a framework that allows translating a set of application-specific constraints into a set of anonymizing routines without user intervention is clearly missing. Such a framework would drastically decrease the amount of implementation effort required by the user.

In this paper we propose a novel approach for anonymizing constrained production data based on the framework of constraint satisfaction problems (CSPs). Following the formal definition in [14], a CSP is defined by a set of variables, $\{x_1,x_2,\ldots,x_n\}$ and a set of constraints, $\{c_1,c_2,\ldots,c_m\}$. Each variable x_i has a nonempty domain D_i of possible values. Each constraint c_i involves some subset of the variables and specifies the allowable combinations of values for that subset. A solution to the CSP is an assignment $\{x_1 = v_1,x_2 = v_2,\ldots,x_n = v_n\}$ of values to all variables that does not violate any constraint (known as a complete and consistent assignment). Given the sensitive information and rules in the production data, we represent and solve the anonymization problem as a CSP. Harnessing the power of the well established CSP framework allows our approach to cope with a wide variety of anonymization constraints that are supported by this framework.

Two other well known methods for achieving privacy are data encryption and data generalization (which is commonly used in Privacy Preserving Data Publishing[15]). However these methods are inappropriate for solving the constrained annonymization problem. Data encryption cannot be used because encrypted data will not necessarily conform to the rules defined on the production data and once it is decrypted, the original production data is exposed. Likewise, when data is generalized, it may violate even the most basic rules such as value set or identity integrity rules[6].

The remainder of this paper is organized as follows. Section 2 gives a formal definition of the constrained anonymization problem. Section 3 presents the details of our CSP-based approach, including a demonstration on a toy example. In Section 4, we evaluate the performance of our method and demonstrate its feasibility. Section 5 reviews the related work. Finally, we summarize our results in Section 6 and describe future research directions.

2 Problem Statement

In this section we formally define the problem of *Constrained Anonymization* of production data:

Definition 1. *Sensitive Field (SF): a field in PD whose values must not be disclosed.*

Definition 2. *Rule (R): A relation/condition defined over a set of fields in PD, which specifies the legal values that cells in these fields can have. In this paper we focus on two types of rules: Integrity and Similarity. An integrity rule defines a condition which must be met for the data to be unimpaired, complete, sound and compliant. An excellent discussion on integrity rules can be found in [6]. For example, an identity rule on field f that requires all values to be different. A similarity rule is aimed at making* *the anonymized data similar to the production data. For example, a rule that requires the average of field f in the anonymized data to be the same as in the production data.*

Definition 3. *Constrained Anonymization: Given a production database PD, a set of sensitive fields SFs and a set of rules Rs, the objective is to derive the test database, denoted by TD, from PD so that all fields in SFs are anonymized (i.e., their original values in the production database are unknown) and all rules in Rs are enforced.*

We assume that the user provides *SI* and *Rs*. Extracting *SI* and *Rs* automatically from the production data is not within the scope of this paper.

3 CSP Approach for Constrained Anonymization

We now propose a method for solving the constrained anonymization problem by reformulating it as a CSP. In our setting, the user provides the following three inputs: (1) The production database *PD*, (2) The set of sensitive fields SFs and (3) a set of rules *Rs*.

Our method first duplicates the test database, denoted by TD, from *PD*. Then our method decomposes *Rs* into independent rule subsets *RSs*. Each rule subset *RS* that contains at least one sensitive field corresponds to a CSP. First, the set of CSP variables is defined and then *RS* is translated into CSP constraints defined over these variables. This CSP, denoted by CSP_{RS}, is then further decomposed into separate sub-CSPs, denoted by $CSPS_{RS}$. Finally, $CSP_i \in CSPS_{RS}$ is solved and the solutions are stored in TD. The union of these solutions is a solution to CSP_{RS} and the union of the solutions to all CSP_{RS} is a solution to the whole anonymization problem. The complete method is described in Algorithm 1 and its different stages are outlined in Fig. 1. In the following subsections we describe the main stages of our method in detail. Subsection 3.5 illustrates the whole process by a toy example.

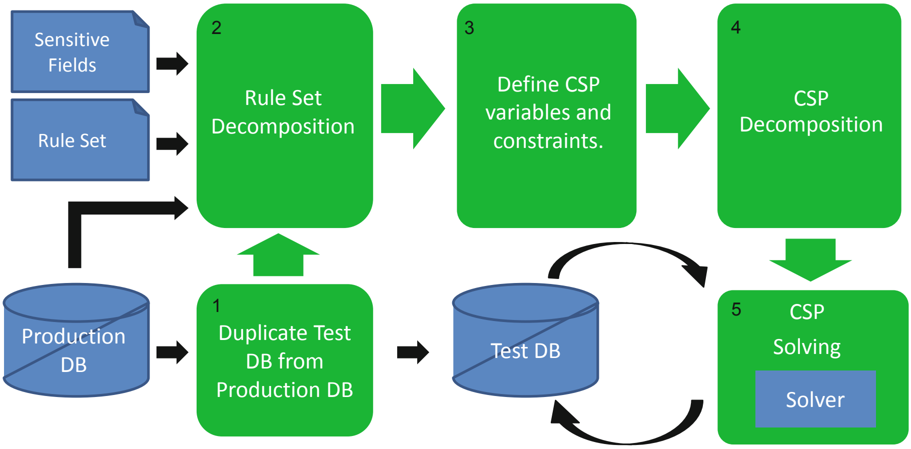


Fig. 1. The different stages of the proposed method

Algorithm 1. constrainedAnonymizationOfProductionData(*PD*,*SFs*,*Rs*)

Input

PD: the production database *PD*.

SFs: the set of sensitive fields.

Rs: the set of rules.

Output

TD: the test database.

```
$TD \leftarrow PD$;
$RSs \leftarrow$ decomposeRuleSet($Rs$);
for all ($RS \in RSs \wedge SF \cap F_{RS} \neq \emptyset$)
    $X_{CSP_{RS}}, C_{CSP_{RS}}, CSPs \leftarrow \emptyset$;
    $X_{CSP_{RS}} \leftarrow$ defineCSPvariables($RS$,$SF$,$TD$);
    for all ($r \in RS$)
        $C_r \leftarrow$ defineCSPconstraints($r$,$X_{CSP_{RS}}$);
        $C_{CSP_{RS}} \leftarrow C_{CSP_{RS}} \cup C_r$;
    end for
    $CSP_{RS} \leftarrow < X_{CSP_{RS}}, C_{CSP_{RS}} >$;
    $C_{vertical}, CSPS_{RS} \leftarrow$ decomposeCSP($CSP_{RS}$);
    if ($C_{vertical} = \emptyset$)
        for all ($CSP_i \in CSPS_{RS}$)
            solve($CSP_i$,$TD$);
        end for
    else
        solveByLocalModification($CSPS_{RS}$, $C_{vertical}$,$TD$);
    endif
end for
return $TD$;
```

3.1 Decomposing the Rule Set

Rules are defined over the fields of *PD* and either represent relationships within a field or between different fields. It is possible to decompose *Rs* into disjoint rule subsets *RSs* such that fields of any two rules in the same subset have to be anonymized together (because they transitively constrain each other through the rules of the subset) and fields of rules in different subsets can be anonymized independently. This decomposition can be efficiently done by finding the connectivity components of the rules set graph, G_{Rs}, in which each rule is represented by a vertex and two vertices are connected if the corresponding rules involve a common field. This rules set decomposition is carried out in Algorithm 1 by the decomposeRuleSet procedure.

3.2 Defining the CSPs

Some, but not necessarily all of the rule subsets resulting from the previous stage are translated into CSPs. Denoting F_{RS} as the set of fields which are associated with rules in *RS*, when $SF \cap F_{RS} \neq \emptyset$, *RS* is translated into CSP_{RS}. Otherwise, there is no need to translate *RS* into CSP and the TD values for fields in F_{RS} are untouched. Moreover, for any field $f \notin \bigcup_{RS \in RSs} F_{RS}$, if $f \in SF$, TD values for field f are randomly drawn from the appropriate domain, otherwise they are left untouched. In words: F_{RS} which does not contain any sensitive field is simply copied from *PD*; fields who do not have any rule defined over them and are sensitive are simply generated; fields who do not have any rule defined over them and are non-sensitive are simply copied from *PD*.

Defining the CSP Variables. A CSP variable is defined for each cell in F_{RS}. We denote x_{ij}^T as the variable that corresponds to the cell in record i, field j and table T. The domains of x_{ij}^T automatically follow from the data type of the values in the j'th field of T and, optionally, from other schema definitions (such as value bounds). The above is conducted in Algorithm 1 by the defineCSPvariables procedure which returns the set of these variables, denoted by $X_{CSP_{RS}} = \{x_{ij}^T\}$.

Defining the CSP Constraints. Whereas rules are defined over fields, the CSP constraints are defined over variables that correspond to individual cells. Thus, after $X_{CSP_{RS}}$ is defined, each $r \in RS$ is translated into the corresponding CSP constraints (conducted in Algorithm 1 by the defineCSPconstraints procedure). For example, consider a rule requiring that a field f_j is unique. This rule will be translated into a single n-ary $allDifferent$[2] constraint: $allDifferent(x_{0j}^T, x_{1j}^T ..., x_{(n-1)j}^T)$, where n is the number of records in table T. Another example can be a rule that requires the value of field f_j to be larger than the value of field f_k. This rule will be translated into n binary *gt* [2] constraints: $\{gt(x_{0j}^T, x_{0k}^T), gt(x_{1j}^T, x_{1k}^T), ..., gt(x_{(n-1)j}^T, x_{(n-1)k}^T)\}$.

Due to the tabular structure of the production DB, the CSP constraints may be of four possible types:

1. Horizontal: constraints which contain at least one subset of different variables $\{x_{ij}^T\}$ that have the same i. Constraints for which all variables have the same i are called strictly horizontal, e.g. the above *gt* constraint.
2. Vertical: constraints which contain at least one subset of different variables $\{x_{ij}^T\}$ that have the same j. Constraints for which all variables have the same j are called strictly vertical, e.g. the above $allDifferent$.
3. Mixed: constraints that are both vertical and horizontal.
4. Cross-table: constraints involving variables from different tables. These constraints may also be any of the other three constraint types.

3.3 Decomposing CSP_{RS}

Once the variables and constraints of CSP_{RS} have been defined, we need to solve it with a CSP solver[1]. In many cases, CSP_{RS} will contain a large amount of variables. As a very common example, consider a CSP resulting solely from the $allDifferent$ constraint previously mentioned and its n associated variables. Solving such a CSP is

[1] There are many available CSP packages (both commercial and not) from which the solver can be chosen. Deciding which package to use must be carefully considered because not all CSP packages support the required variable domains and/or constraints. Thus, the choice of CSP package can affect the extendability of the method.

impractical in many cases due to the overwhelming amount of variables. In general, it is reasonable to assume that for any solver s, there exists some manageable number of variables k_s (which may vary between different solvers) for which any CSP with more than k_s variables cannot be handled. Thus, our method further decomposes CSP_{RS} into a set of separate sub-CSPs, denoted by $CSPS_{RS}$, where the number of variables in each sub-CSP does not exceed k_s (Algorithm 2). This decomposition is based on the standard CSP decomposition into separate sub-problems according to its constraint graph[14]. We begin by finding the connectivity components of the constraint graph associated with CSP_{RS}. Then, constraints from $C_{CSP_{RS}}$ whose variables are part of a connectivity component of size $> k_s$ are removed from $C_{CSP_{RS}}$. If we assume that the total number of fields in PD is $\leq k_s$, only vertical constraints will be removed. The removed constraints are stored in $C_{vertical}$ and the standard decomposition is applied on CSP_{RS} (with the non-vertical set of constraints). This ensures that each $CSP_i \in CSPS_{RS}$ will involve less than k_s variables. We then iteratively merge any two sub-CSPs whose combined number of variables does not exceed k_s[2].

3.4 Solving $CSPS_{RS}$

If $C_{vertical}$ is empty, all $CSP_i \in CSPS_{RS}$ can be solved independently, even simultaneously on different machines. That is, each CSP_i is formulated in terms of the CSP solver, solved and the results are stored in TD (this is done by the *solve* procedure used in Algorithm 1). However, if $C_{vertical}$ is not empty, the sub-CSPs in $CSPS_{RS}$ are dependant and must be solved as one CSP. Fortunately, in the context of Constrained Anonymization, we can modify the sub-CSPs so that the resulting sub-CSPs are still dependant but can be solved separately in a sequential manner. A key aspect of our scenario is the fact that there always exists at least one known solution to any sub-CSP corresponding to the relevant values in PD. Our method takes advantage of this solution by repeatedly attempting to make local modifications to it. We refer to this as the local modifications, or *LM* heuristic which is described in Algorithm 3. More specifically, for each $CSP_i \in CSPS_{RS}$, we derive and solve a new CSP, denoted by CSP_i^*, for which $X_{CSP_i^*} = X_{CSP_i}$ and $C_{CSP_i^*} = C_{CSP_i} \cup C_{vertical}$, where any variable from $C_{vertical}$ not included in $X_{CSP_i^*}$ is replaced by its corresponding value from TD. Since CSP_i^* has no more than k_s variables and at least one solution, it is necessarily tractable. If CSP_i^* has exactly one solution, it will necessarily be the set of values currently in TD. Otherwise, the solution will constitute a local modification that differs from TD to a varying extent (by up to k_s different values).

3.5 A Toy Example

We now demonstrate our method on a toy production database as illustrated in Fig. 2. We denote the EMPLOYEE table by T^E and the BUDGET table by T^B. We assume that all fields in the production DB except *age* and *percent_fulltime* are sensitive and that Rs only consists of the following integrity rules:

[2] It has been our experience that, when possible, solving fewer problems with a larger number of variables outperforms solving more problems with a smaller number of variables.

Algorithm 2. decomposeCSP(X,C)

Input

X: variable set of the CSP being decomposed.

C: constraint set of the CSP being decomposed.

Output

$CSPS_{RS} = \{CSP_i | CSP_i =< X_i, C_i >\}$: the set of sub-problems for which $|X_{CSP_i}| \leq k_s, \forall i$.

$C_{vertical}$: the set of vertical constraints from C that link the sub-problems in $CSPS_{RS}$.

```
$C_{vertical}, CSPS_{RS} \leftarrow \emptyset$;
for all ($c \in C$)
    if ($c$ is vertical and $X_c$ are part of a connectivity component of size $> k_s$)
        $C_{vertical} \leftarrow C_{vertical} \cup \{c\}$;
    end if
end for
while ($C \neq \emptyset$)
    remove constraint $c$ from $C$;
    for all ($x \in X_c$)
        if ($\exists CSP_i \in CSPs$ such that $x \in X_i$)
            $C_i \leftarrow \{c\} \cup C_i$;
            $X_i \leftarrow X_i \cup \{x_j | x_j \in X_c\}$;
        else
            $CSP_i \leftarrow< \{x_j | x_j \in X_c\}, \{c\} >$;
            $CSPS_{RS} \leftarrow CSPS_{RS} \cup \{CSP_i\}$;
        end if
    end for
end while
while ($\exists CSP_i, CSP_j \in CSPS_{RS} : |X_{CSP_i}| + |X_{CSP_j}| \leq k_s$)
    $CSP_i \leftarrow< (X_{CSP_i} \cup X_{CSP_j}), (C_{CSP_i} \cup C_{CSP_j}) >$;
end while
return $CSPS_{RS}$,$C_{vertical}$;
```

1. The *id* field of T^B is a unique primary key.
2. The *id* field of T^E is a unique primary key.
3. The *id* field of T^E must be a valid cellular phone number that conforms to the following regular expression pattern: "05[0247][0-9][0-9][0-9][0-9][0-9][0-9]".
4. The *hire_date* must be a valid date between the 1/1/2007 and the 12/31/2010.
5. The *end_date* must be a valid date between the 1/1/2007 and the 12/31/2010.
6. The *hire_date* must precede the *end_date*.
7. The following equation must hold: $\frac{fulltime\_salary \cdot percent\_fulltime}{100} = monthly\_pay$.
8. The *budget_id* field is a foreign key which references the primary key in T^B.
9. For any budget b, the sum of *monthly_pay* for all employees funded by b must be at most the value of the *value* field corresponding to b.

Note that the resulting anonymized test data may greatly differ from the original production data because no similarity rules are defined. Also note that since this toy example

Algorithm 3. solveByLocalModification($CSPS_{RS}$,$C_{vertical}$,TD)

Input

$CSPS_{RS} = \{CSP_i | CSP_i =< X_i, C_i >\}$: a set of CSP sub-problems (generated by the decomposition stage) for which $|X_{CSP_i}| \leq k_s, \forall i$.

$C_{vertical}$: the set of vertical constraints that link the sub-problems in $CSPS_{RS}$.

TD: the test database.

```
for all ($CSP_i \in CSPS_{RS}$)
    $X_{CSP_i^*} \leftarrow X_{CSP_i}$;
    $C_{verticalTemp} \leftarrow C_{vertical}$;
    for all ($\{x | x \in X_c \wedge c \in C_{verticalTemp} \wedge x \notin X_{CSP_i^*}\}$)
        replace $x$ with its corresponding cell value in $TD$;
    end for
    $C_{CSP_i^*} \leftarrow C_{CSP_i} \cup C_{verticalTemp}$;
    $CSP_i^* \leftarrow < X_{CSP_i^*}, C_{CSP_i^*} >$;
    solve($CSP_i^*$);
end for
```

	id	*value*
0	0	15717
1	1	14287
i / *j*	**0**	**1**

(a) The BUDGET table.

	id	*age*	*hire_date*	*end_date*	*fulltime_salary*	*percent_fulltime*	*monthly_pay*	*budget_id*
0	546410616	30	20100130	20101031	18000	59	10620	0
1	521972696	22	20100605	20100915	15250	86	13115	1
2	508207463	25	20101002	20101026	5700	72	4104	0
i / *j*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

(b) The EMPLOYEE table.

Fig. 2. A production database containing two tables that store company records of employees and budgets by which their salaries are funded. The record (denoted by i) and field (denoted by j) indices of the table cells are also indicated.

is very small, we neither apply the rule set decomposition nor the *LM* approach. The resulting CSP variables are:

1. $\{x^B_{ij} | i = 0,1 \wedge j = 0,1\}$.
2. $\{x^E_{ij} | i = 0,1,2 \wedge j = 0,2,3,4,5,6,7\}$.

and the CSP constraints are:

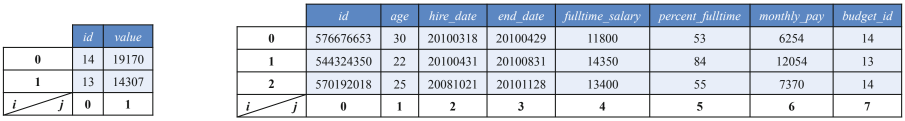
1. $allDifferent(x^B_{00}, x^B_{10})$.
2. $allDifferent(x^E_{00}, x^E_{10}, x^E_{20})$.
3. $\{regular(regexp_1, x^E_{i0}) | i = 0,1,2\}$ where $regexp_1$ = “05[0247][0-9][0-9][0-9][0-9][0-9][0-9]”.
4. $\{regular(regexp_2, x^E_{ij}) | i = 0,1,2 \wedge j = 2,3\}$ where $regexp_2$ = “20(0[7-9]|10)(0[1-9]|1[0-2])(0[1-9]|1[0-9]|2[0-9]|30|31)”.

5. $\{x^E_{i2} < x^E_{i3} | i = 0,1,2\}$.
6. $\{\frac{x^E_{i4} \cdot x^E_{i5}}{100} = x^E_{i6} | i = 0,1,2\}$.
7. $\{x^E_{i7} = x^B_{00} | i = 0,2\}$and $x^E_{17} = x^B_{10}$.
8. $x^E_{06} + x^E_{26} \leq x^B_{01}$ and $x^E_{16} \leq x^B_{11}$.

The *CSP* sub-problems after CSP decomposition are:

1. $CSP_1 =< \{x^B_{00}, x^B_{10}, x^E_{07}, x^E_{17}, x^E_{27}\}, \{allDifferent(x^B_{00}, x^B_{10}), x^E_{07} = x^B_{00}, x^E_{17} = x^B_{10}, x^E_{27} = x^B_{00}\} >$.
2. $CSP_2 =< \{x^E_{00}, x^E_{10}, x^E_{20}\}, \{allDifferent(x^E_{00}, x^E_{10}, x^E_{20}), regular(regexp_1, x^E_{00}), regular(regexp_1, x^E_{10}), regular(regexp_1, x^E_{20})\} >$.
3. $CSP_3 =< \{x^E_{02}, x^E_{03}\}, \{x^E_{02} < x^E_{03}, regular(regexp_2, x^E_{02}), regular(regexp_2, x^E_{03})\} >$.
4. $CSP_4 =< \{x^E_{12}, x^E_{13}\}, \{x^E_{12} < x^E_{13}, regular(regexp_2, x^E_{12}), regular(regexp_2, x^E_{13})\} >$.
5. $CSP_5 =< \{x^E_{22}, x^E_{23}\}, \{x^E_{22} < x^E_{23}, regular(regexp_2, x^E_{22}), regular(regexp_2, x^E_{23})\} >$.
6. $CSP_6 =< \{x^E_{04}, x^E_{05}, x^E_{06}, x^E_{24}, x^E_{25}, x^E_{26}, x^B_{01}\}, \{\frac{x^E_{04} \cdot x^E_{05}}{100} = x^E_{06}, \frac{x^E_{24} \cdot x^E_{25}}{100} = x^E_{26}, x^E_{06} + x^E_{26} \leq x^B_{01}\} >$.
7. $CSP_7 =< \{x^E_{14}, x^E_{15}, x^E_{16}, x^B_{11}\}, \{\frac{x^E_{14} \cdot x^E_{15}}{100} = x^E_{16}, x^E_{16} \leq x^B_{11}\} >$.

Using the CSP solver provided in the CHOCO package [1], each sub-problem is solved and the results are inserted into the test DB as illustrated in Fig. 3. A close inspection will show that all values of the test DB indeed satisfy all *Rs*.



	id	*value*
0	14	19170
1	13	14307
i / *j*	**0**	**1**

(a) The anonymized BUDGET table.

	id	*age*	*hire_date*	*end_date*	*fulltime_salary*	*percent_fulltime*	*monthly_pay*	*budget_id*
0	576676653	30	20100318	20100429	11800	53	6254	14
1	544324350	22	20100431	20100831	14350	84	12054	13
2	570192018	25	20081021	20101128	13400	55	7370	14
i / *j*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

(b) The anonymized EMPLOYEE table.

Fig. 3. The test DB resulting from anonymizing the production DB of Fig. 2

4 Experimental Results

To demonstrate the feasibility of our method, we simulated the anonymization of production databases, based on the toy example of section 3.5. The production databases were generated by a variation of our anonymization method which relies on the *IG* heuristic described in Appendix A. The production databases that were generated differ by the number of records they contain in the *EMPLOYEE* (n) tables (the number of records in the *BUDGET* table was always $n/10$). Each production database, was anonymized using the *LM* heuristic and without applying any heuristic (*NH*).

All experiments were conducted on an Intel Core 2 Duo CPU 2.4 GHz personal computer with 3 GB of RAM, running Windows 7 Enterprise and MySQL 5.136 with default settings. For the solving stage, we used the CHOCO CSP solver [1], version 2.1.1 with $k_s = 100$ (which was determined in preliminary experiments).

Table 1 compares the time required for anonymization of each production database, using *LM* and *NH*. Since *Rs* includes two *allDifferent* constraints, it led to large CSP_{RS}. Even for relatively small sized n (i.e., 5,000), *NH* was not able to complete its execution (denoted by *NA* in the table). Thus, we could evaluate *NH* only for the experiments involving relatively few records (less than 5,000). Regarding *LM*, even though our example contained several rules of different types and despite the fact that CSP is NP-hard in general, the solving process was completed in a reasonable time.

Table 1. Execution time (in seconds) required to anonymize the different production DBs in each experiment

Heuristic \ n	100	1,000	5,000	10,000	100,000
NH	2	35	NA	NA	NA
LM	2	13	70	141	2179

5 Related Work

As opposed to encryption and generalization, which derive the test values by modifying the corresponding production values, our CSP-based approach falls within the category of data generation methods. Data generation methods derive the test value of each cell in the database with complete disregard to the original production value of that cell. A number of data generation methods have been published, each focusing on aspects such as data dependencies or fast generation of a large amount of data.

In [9], the authors introduce the data dependency graph and describe a modified topological sort used to determine the order in which data is generated. However, it is unclear how this method can handle: (1) cycles in the graph; (2) rules that involve several fields in which no field is explicitly represented as a formula of the others, for example, $X^2 + Y^2 = 1$. In [4], the authors present a C-like Data Generation Language (DGL) which allows the generation of data using the definition of its data types, rows, iterators and distributions. However, dependencies in data (such as foreign keys) must be explicitly handled by the user. In [8] the authors suggest a parallel synthetic data generator (PSDG) designed to generate "industrial sized" data sets quickly using cluster computing. PSDG depends on an XML based synthetic data description language (SDDL) which codifies the manner in which generated data can be described and constrained. However, PSDG suffers from the same limitations of [9]. In [3], the authors suggest a system to generate query aware data. The system is given a query and produces data that is tailored to the specific query. However it is not very useful for general purpose generation of test data for ad hoc testing. In [7], a hardware based technique to quickly generate large databases on multi-processor systems is described. The authors focus on low-level implementation details (such as process spawning and table partitioning strategies) to generate data using a dedicated hardware architecture. In [5], the authors focus on generating data for performance tests. More specifically, they try to generate data which preserves the query optimizer characteristics of the original data. The generation process must preserve three data characteristics: consistence, monotony and dependence between columns. To summarize, as dependencies and rules defined over fields become more complicated, none of the existing methods is general enough to handle them.

6 Summary and Future Work

We have presented a novel, CSP-based approach for anonymizing production data which involves complex constraints. Due to the power of CSP, our method can support a wide variety of general constraints that are frequently encountered in production data. Such constraints include not only mandatory integrity constraints but also constraints that can be defined in order to make the test data similar to the production data.

After formulating the Constrained Anonymization problem as a CSP, we decompose it into separate and smaller sub-CSPs. Often, the resulting sub-CSPs are independent and therefore simpler to solve. Nonetheless, sub-CSPs which contain vertical constraints will remain dependent and intractable due to the overwhelming amount of variables. In this case, our method modifies the sub-CSPs so that they are still dependant but can be solved separately in a sequential manner. This is possible due to the fact that there always exists at least one known solution to any sub-CSP which can be taken from the production database.

We have demonstrated the feasibility of our method by simulating the anonymization of synthetic production DBs (see Table 1). Our simulations emphasize the necessity of the proposed heuristics when the production DB contains a typically large number of records.

In future work we intend to: (1) Develop an automated method for identifying *Rs* (both integrity rules and similarity rules), perhaps by analyzing the DB schema and/or applying various data-mining solutions to the production data; (2) Formally define a measure of production data anonymization which will allow us to evaluate the quality of the anonymization algorithm; (3) Improve the performance of our method by solving independent sub-CSPs in parallel on distributed machines; (4) Evaluate our method on a real (or a benchmark) database.

References

1. Choco solver (2010), `http://choco.emn.fr`
2. Beldiceanu, N., Carlsson, M., Rampon, J.X.: Global constraint catalog (2005)

3. Binnig, C., Kossmann, D., Lo, E., Özsu, M.T.: Qagen: generating query-aware test databases. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 341–352. ACM, New York (2007)
4. Bruno, N., Chaudhuri, S.: Flexible database generators. In: VLDB 2005: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 1097–1107. VLDB Endowment (2005)
5. Castellanos, M., Zhang, B., Jimenez, I., Ruiz, P., Durazo, M., Dayal, U., Jow, L.: Data desensitization of customer data for use in optimizer performance experiments. In: Proceedings of the 26th IEEE International Conference on Data Engineering (2010)
6. Duncan, K., Wells, D.: A Rule-Based Data Cleansing. Journal of Data Warehousing 4(3), 146–159 (1999)
7. Gray, J., Sundaresan, P., Englert, S., Baclawski, K., Weinberger, P.J.: Quickly generating billion-record synthetic databases. ACM SIGMOD Record 23(2), 252 (1994)
8. Hoag, J.E., Thompson, C.W.: A parallel general-purpose synthetic data generator. SIGMOD Rec. 36(1), 19–24 (2007)
9. Houkjar, K., Torp, K., Wind, R.: Simple and realistic data generation. In: Proceedings of the 32nd international conference on Very large data bases, p. 1246. VLDB Endowment (2006)
10. Camouflage Software Inc.: Camouflage transformers. Data Sheet (2009), `http://www.datamasking.com`
11. Camouflage Software Inc.: Enterprise-wide data masking with the camouflage translation matrix. Data Sheet (2009), `http://www.datamasking.com`
12. Camouflage Software Inc.: Secure analytics - maximizing data quality & minimizing risk for banking and insurance firms. White Paper (2009), `http://www.datamasking.com`
13. Grid-Tools GridTools Ltd.: Simple data masking. Data Sheet (2009), `http://www.grid-tools.com`
14. Russell, S.J., Norvig, P., Canny, J.F., Malik, J., Edwards, D.D.: Artificial intelligence: a modern approach. Prentice Hall, Englewood Cliffs (1995)
15. Wang, K., Chen, R., Yu, P.S.: Privacy-Preserving Data Publishing: A Survey on Recent Developments (2010)

A The Incremental Generation (*IG*) Heuristic

Consider a setting in which we would like to utilize our CSP approach to generate a constrained database. In this case the *LM* heuristic cannot be used since it relies on modifying an existing solution. Thus, we need a different heuristic to handle $CSPS_{RS}$ (resulting from the decomposition of CSP_{RS}) when $C_{vertical} \neq \emptyset$. For some special cases of $C_{vertical}$, we can apply an iterative process in which we solve $CSP_i \in CSPS_{RS}$ based on previously solved $CSP_j, \forall j < i$. More specifically, in the *i*'th iteration, we solve a new CSP, denoted by CSP_i^*, for which $X_{CSP_i^*} = X_{CSP_i}$ and $C_{CSP_i^*} = C_{CSP_i} \cup C_{vertical}$. For any variable belonging to a constraint $c \in C_{vertical}$ not included in $X_{CSP_i^*}$, if $x \in X_{CSP_j^*}$ and $j < i$, it is replaced by its solution in CSP_j^*, otherwise it is removed from X_c. This *Incremental Generation* (*IG*) approach is described in Algorithm 4. Note that when $C_{vertical}$ contains only one $allDifferent$ constraint (like in our evaluation setting), the *IG* heuristic will find a solution to $CSPS_{RS}$ and $C_{vertical}$, if and only if, a solution exists to CSP_{RS}.

Algorithm 4. solveByIncrementalGeneration($CSPS_{RS}$,$C_{vertical}$,TD)

Input

$CSPS_{RS} = \{CSP_i | CSP_i =< X_i, C_i >\}$: a set of CSP sub-problems (generated by the decomposition stage) for which $|X_{CSP_i}| \leq k_s, \forall i$.

$C_{vertical}$: the set of vertical constraints that link the sub-problems in $CSPS_{RS}$.

TD: the empty test database.

```
for all ($CSP_i \in CSPS_{RS}$)
    $X_{CSP_i^*} \leftarrow X_{CSP_i}$;
    $C_{verticalTemp} \leftarrow C_{vertical}$;
    for all ($\{x | x \in X_c \wedge c \in C_{verticalTemp} \wedge x \notin X_{CSP_i^*}\}$)
        if (the cell corresponding to $x$ in $TD$ is not empty)
            replace $x$ with its corresponding cell value in $TD$;
        else
            $X_c \leftarrow X_c \setminus \{x\}$;
        end if
    end for
    $C_{CSP_i^*} \leftarrow C_{CSP_i} \cup C_{verticalTemp}$;
    $CSP_i^* \leftarrow < X_{CSP_i^*}, C_{CSP_i^*} >$;
    solve($CSP_i^*$);
end for
```

Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems

Giampaolo Armellin[1], Dario Betti[1], Fabio Casati[2], Annamaria Chiasera[1,2], Gloria Martinez[2], and Jovan Stevovic[1,2]

[1] GPI SpA, Via Ragazzi del '99, 13,
Trento, Italy
{garmellin,dbetti,achiasera,jstevovic}@gpi.it
[2] Information Engineering and Computer Science Department,
University of Trento, Italy
casati@disi.unitn.eu, glomarin@gmail.com

Abstract. Processes in healthcare and socio-assistive domains typically span multiple institutions and require cooperation and information exchange among multiple IT systems. In most cases this cooperation today is handled "manually" via document exchange (by email, post, or fax) and in a point-to-point fashion. One of the reasons that makes it difficult to implement an integrated solution is that of privacy, as health information is often sensitive and there needs to be a tight control on which information is sent to who and on the purpose for which it is requested and used. In this paper we report on how we approached this problem and on the lessons learned from designing and deploying a solution for monitoring multi-organization healthcare processes in Italy. The key idea lies in combining a powerful monitoring and integration paradigm, that of event bus and publish/subscribe systems on top of service-oriented architectures, with a simple but flexible privacy mechanism based on publication of event summaries and then on explicit requests for details by all interested parties. This approach was the first to overcome the privacy limitations defined by the laws while allowing publish/subscribe event-based integration.

Keywords: interoperabilty, SOA, EDA, privacy enforcement.

1 Introduction

Recently we are assisting to a strong commitment of the public administrations toward e-government projects focused mainly on the dematerialization of the administrative processes, to monitor, control and trace the clinical and assistive processes with a fine-grained control on the access and dissemination of sensitive information. Such processes involve both private and public organizations from the social and healthcare domain providing a variety of services, from food delivery to house cleaning, telecare and nursing outside the hospitals. Typically these processes span multiple institutions and to support their integration or, as a minimum, to monitor their execution, some degree of integration among the IT systems of these institutions is necessary.

Application Integration poses many well-known problems in terms of bridging the different protocols and data models of the various systems involved. In this paper we

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 54–69, 2010.
© Springer-Verlag Berlin Heidelberg 2010

study these problems from a privacy perspective. Specifically, our goal here is to report on the problems, solutions, and lessons learned from the development and deployment of a large integration project called CSS, aimed at monitoring healthcare and social processes across the different government and healthcare institutions in the region of Trentino, Italy. Trentino was used as a pilot region in Italy, and the results will next be applied at the national level.

The project allows the **cooperation** of applications from different agencies (both public and private) to provide high quality clinical and socio-assistive services to the citizens. The institutions to be integrated range from small civic centers to the regional government. As such, they are very high in number and very heterogeneous in nature. Furthermore, institutions progressively join the integrated CSS process monitoring ecosystem, so that an additional challenge lies in how to facilitate the addition of new institutions. From a data privacy perspective, these kinds of scenarios present tough challenges, such as:

- how to make it simple for all the various data sources to define the privacy constraints and enforce them. Simplicity here is the key as otherwise the complexity of the information system and in some cases also of the law make the problem already hard to tackle, and unless the privacy model remains really simple it is very hard to understand which information is protected and to what extent.
- how to achieve patient/citizen empowerment by supporting consent collection at data source level (opt-in, opt-out options to share the events and their content)
- how allow monitoring and tracing of the access request (who did the request and why/for which purpose?)
- how to support the testability and auditability of privacy requirements: owners of data sources often require that the privacy rules they are asked to define can be tested and audited so that they can be relieved of the responsibility of privacy breaches.

In the paper we show how we addressed these challenges, the lessons we learned, and the extent to which they can be generalized. In a nutshell, the solution we adopted is based on the following observations and decisions:

- First, we integrate systems based on events. Performing process analysis via a traditional data warehousing approach is not feasible as it would be too complex to dive into each of the information sources because of their number and diversity. Instead, we ask each source to model and distribute only the events which are relevant from the perspective of the process to be monitored. In this way the sources can focus only on getting the information needed for those events.
- Second, we realized that each data source owner wants to define their own privacy rules (even if they have to obey the same law). Hence, there cannot be a single institution acting as centralized data collector (e.g. the one that manages the CSS platform) that can define the privacy rules. However, defining the privacy rules on the source information system is complex and requires technical expertise that we cannot assume the data source owner (e.g. administrative and medical staff) has. We address this problem by defining privacy rules on events that, as we will see, not

only makes specifications relatively simple, they also avoid the danger of having over-constrained privacy specifications.

- Third, in many cases consumers do not need all the details that are generated by a data producer. We want to be able to define and control in a fine-grained manner which data is delivered to consumers. Privacy rules are dependent on which institutions is consuming the data. One of the problem in this case is who will decide which data can be transferred from the data sources to third party entities. We address this via a fine-grained, purpose-based, two-phase access control mechanism where each source initially exposes (publishes to the consumer) events that are only partially specified, just enough for the consumer to understand if they need more detail. Then, each interested consumer must explicitly request the details to the source (along with a purpose statement), via the CSS platform. This protocol allows sources to specify different visibility rules based on explicitly stated purposes, which makes the approach compliant with privacy regulations.

We have experienced that these ingredients taken together make the system easy to use and also legal from a privacy perspective, as well as easy to audit and test. In the following we describe the scenario in more detail and then present the solution that CSS provides.

2 The Scenario

Figure 1 shows the main actors involved in the social-health scenario and the flows of communication among them that today occurs mainly on paper. In this scenario, a patient needs healthcare and socio-assistive assistance (socio-health care services). The patient interacts with healthcare and social service providers (socio-health care

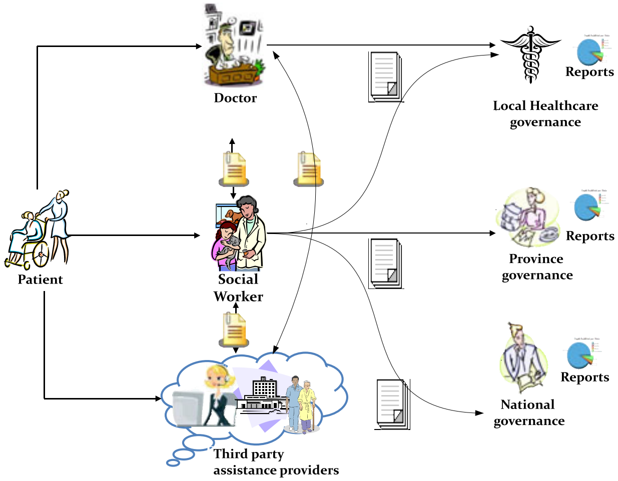


Fig. 1. Social and Health local scenario

service providers) such as doctors, social workers, or third parties service providers. The providers communicate mainly via documents or mail and, in some cases, by e-mail. Most of the times the patients themselves should bring their documents from office to office.

Typically each service provider has to provide data at different level of granularity (detailed vs aggregated data) to the governing body (province or ministry of health and finance) for accountability and reimbursement purposes. The governing body also uses the data to assess the efficiency of the services being delivered.

In this scenario is easy to have unintentional privacy breaches, as the data owners (doctors, social workers and third party assistance providers) that collect the data from the patients do not have any fine-grained control on the data they exchange or send to the governing body. Very often, either they make the data inaccessible (over-constraining approach) or they release more data than required to the others to complete their job in conflict with the principle of minimal usage [12].

Furthermore, as there is no central controller of the data access requests there is no way to trace how data is used by whom and for what purpose and to be able to answer to auditing inquiry by the privacy guarantor or the data subject herself.

3 State of the Art

In this section we will briefly analyze the state of the art related to business process interoperability, data interoperability and privacy management. We focus on the health-care field but the considerations we did are valid also in more general contexts. Service Oriented Architecture (SOA) [1] through web services are emerging as the standard for interoperability in intra-domain situations. One of the major problems of the SOA pattern is the point-to-point synchronous interaction that is established between involved actors. Event Driven Architectures (EDA) decouples service providers and consumers through asynchronous messaging and solves the point-to-point communication problem of SOA. In our project is adopted an Event Driven SOA approach [15] in which involved entities exchange the data through Web Service invocation and the Data Controller implements a Publish/Subscribe [10] event-driven model that allows many entities to subscribe to the same type of event.

One of the peculiarity of the Health domain is that it is characterized by extremely sensitive data and one of the main constrains in developing eHealth systems (EHR, PHR etc) is that the data should remain under the responsibility of the owners (i.e. producers). One possible solution, proposed by consortium like IHE - Integrating the Healthcare Enterprise [11] and applied in various eHealth projects [4] [18] [17], proposes a central registry with searching and data management functionalities over meta data that links and describes the data generated by producers. Various implementations of registries like UDDI, ebXML, XML/EDI [22] are available. The most appropriate in terms of flexibility, interoperability and adoption for the Health domain [9], that is adopted in our project is ebXML by OASIS [3].

The problem of privacy and security management in multi-domain environment where multiple heterogeneous entities are involved like in our scenario has a growing importance. The main problems are how to express and how to apply privacy constraint in a multi-domain environment. One possible approach of dealing with privacy is by defining constraints on the data at the moment the data is released to a third party [5]. Current privacy policy languages like P3P [19] and access control languages like IBM's EPAL or OASIS' XACML [2] allow to express privacy requirements in terms of the authorized purposes for the use of the data. In the SOA context various XML standards [23] that are more suitable in a distributed and inter-applications communication have been defined. The eXtensible Access Control Markup Language (XACML) [16] is a standardized security-policy language that allows an enterprise to manage the enforcement of all the elements of its security policy in all the components of its information systems.

XACML is mainly used for document-level access control that is too limited to support a fine-grained security enforcement. In some extensions like [6], XACML is used mainly to evaluate runtime requests on entire documents on an allow/deny model. In [14] is proposed a slightly different approach in which a query rewriting technique is used to provide to users with different access rights different views on data. In [8] is proposed a fine-grained solution that allows the definition and enforcement of access restrictions directly on the structure and content of the documents. For each request the system provides a specific XML response with the associated Document Type Definition (DTD). We apply the same idea but using XML schema (XSD) [21] instead of DTD because of more compliance with a Web Service based architecture. Furthermore our approach does not imply modification of Web Services interfaces and messages schema for each requests in order to provide a simpler solution for the involved entities that has heterogeneous level of ICT expertise.

However, all languages summarized before are not intuitive enough to be used by a privacy expert to express even simple constraints. They require a translation step to make the privacy constraints enforceable on the data schema. To overcome this problem we proposed a web application that provides an intuitive step-by-step policy configuration process. The definition process considers single events and it does not require any knowledge of underlying sources DB schema.

4 Event-Based Architecture

The CSS platform derives from a in-depth study of the healthcare and socio-assistive domains conducted in synergy with the Social welfare Department, the Health Care Agency, the two major municipalities and local companies providing telecare and elderly services in the Trentino region. In particular, we analyzed, together with the domain experts, the clinical and assistive processes to be monitored that involves all the partners to: identify the organizational and technological constraints of the IT systems in use, capture the business processes executed and the bits of data they produce and exchange with each other. This process-oriented analysis approach relieves us from the internal complexity of the single information sources as it focuses only on the 'visible' effects of the business processes captured by data events.

Events are the atomic pieces of information exchanged between data producers and data consumers according to event-based architectural pattern [15]. The infrastructure driving events from the sources to destinations is SOA based implemented with web services on top of an enterprise service bus that allows for event distribution across all interested parties.

A data event signals a change of state in a source system that is of interest to the other parties (e.g. the completion of a clinical exam by an hospital is of interest to the family doctor of the patient) and should be traced and notified to them. The composition of data events on the same person produced by different sources gives her social and health profile the caregivers needs to take care of her.

An event is described by two types of messages: notification and detailed message (in the rest of the paper we will use message and event interchangeably). Together they fully characterize the occurrence of an event but at a different levels of detail and sensitiveness:

- the notification message contains only the data necessary to identify a person (who), a description of the event occurred (what), the date and time of occurrence (when) and the source of the event (where). It contains the identifying information of a person but not sensitive information.
- the detail message contains all the data associated to an event generated during the assistance process (e.g. the result of a clinical trial or the report of a psychological analysis) that may be particularly sensitive and for this reason is maintained at the sources.

It is like if the profile of a person is represented by a sequence of “snapshots” (the events) and each snapshot has a short description of meta-data that explains where, when and by whom the “photo” was taken and what’s the picture about (i.e., notification message); the picture is the detail (the detail message) and you can see part of it only if the owner of the picture gives you the permission. A consumer will ask for the detail only if necessary based on the short information in the notification.

As shown in Figure 2 the central rooting node of the CSS platform is represented by the *data controller* that maintains an index of the events (*events index*, implemented according to the ebXML [3] standard) as it stores all the notification messages published by the producers and notifies them automatically to the interested parties that has previously subscribed to the corresponding classes of event. The detail messages are maintained only in the producer’s system as owner and responsible body for that sensitive information.

This dichotomy is what makes the architecture compliant to the national privacy regulations [13], [12] which prohibits the duplication of sensitive information outside the control of the data owner.

In the next section we will explain in more details how the data producer could define the privacy policies that regulates the use of the events and in particular the distribution of the notifications and the access to the details. Notifications are sent only to authorized consumers that can ask more details for specific purposes. This allows a fine-grained access control and allows the data source to hide part of the details to certain consumers depending on their functional role in the organization (e.g. social welfare department or radiology division) and purposes of use. The data controller is in charge of applying

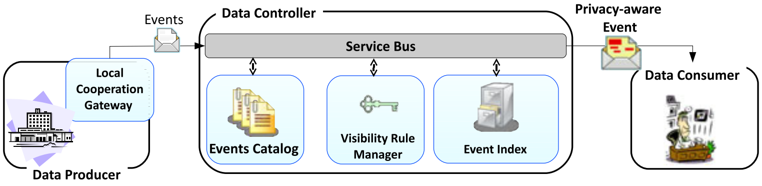


Fig. 2. General event-based architecture for the cooperation of healthcare and socio-assistive systems

the privacy policies to retrieve only what the consumer is authorized to see from the producer. It also offers the following services and functionalities:

- support both data producers and data consumers in joining the CSS platform and in particular: the data producer declares the classes of events it will generate in the event catalog and defines the privacy policies for their use by means of the visibility rule manager; the data consumer subscribes to the classes of events it is interested in;
- receive and store the notification messages and deliver them to the subscribers by means of a service bus (in the current prototype we customized the open source ESB *ServiceMix* [20]). The identifying information of the person specified in the notification is stored in encrypted form to comply with the privacy regulations;
- resolve request for details from the data consumer by enforcing the privacy policies and retrieving from the source the required and accessible information;
- resolve events index inquiry;
- maintains logs of the access request for auditing purposes;

The data controller acts as a mediator and broker between data sources and consumers and is the guarantor for the correct application of the privacy policy for retrieving the details and exploring the notifications. A data consumers can query the events index to get the list of notifications it is authorized to see without necessarily subscribe to the corresponding notification messages.

The decoupling between notification messages and detail messages is not only 'structural', as they are carried in different XML messages, but also temporal: typically a data

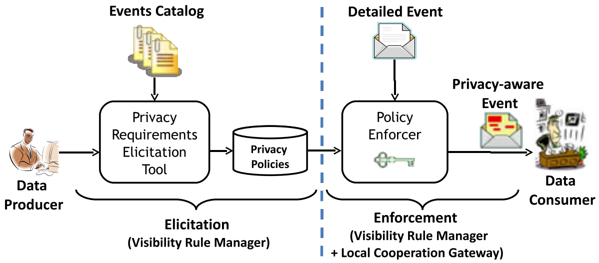


Fig. 3. Privacy Constraints elicitation and enforcement approach

consumer gets from the events index (either by automatic notification or by querying the index) an event and only at a later time it asks for the corresponding details. Requests for details may arrive to the data controller even months after the publication of the notification. This requires the data producer the capability to retrieve at any time the details associated to a past notification. These functionalities are encapsulated in the *local cooperation gateway* provided as part of the CSS platform to further facilitate the connection with the existing source systems. This module persists each detail message notified so that they can be retrieved even when the source systems are un-accessible.

5 Event-Based Privacy Constraints

The participation of an entity to the architecture (as data producer or data consumer) is conditioned to the definition of precise contractual agreements with the data controller. The contract between a data source and the data controller constraints how the data could be accessed by a third party and in particular:

- which data consumers could receive notifications (routing rules)
- how many details the data consumer could obtain from a request for details (details rules)

The data controller is not able to define such rules as it does not know which part of the event detail is really sensitive and which instead are its safe usages. On the other hand for the data source the definition of privacy rules that can be directly enforced in the system (e.g. in XACML [23]) is a complex and tedious task as it has to do it for each class of event details and requires technical expertise the typical privacy expert does not have.

To facilitate the data sources in this task we support the whole lifecycle of an event from the definition of the privacy policies (routing and detail rules) to their enforcement to resolve details requests.

In particular, we provide a GUI for the intuitive definition of the privacy policies on each class of events (Privacy Requirements Elicitation Tool) that produces policies that are directly enforceable in the system, a module that matches a detail request with the corresponding privacy policy (*Policy Enforcer*), and a module to be installed at the sources for the enforcement of the detail rules of a privacy policy in case a request is authorized (*Local Cooperation Gateway*).

The data producer declares the ability to generate a certain type of event (the *Event Details*). The structure of the event is specified by an XSD that is ‘installed’ in an *event catalog* module. The event catalog, as the structure of its events, is visible to any candidate data consumer that has previously signed a contract with the data controller to join to the cooperation architecture. In order to subscribe to a class of event (e.g. a *blood test*) or to access to its data content, the data consumer (e.g. a *family doctor*) should have the authorization by the data producer. If there is not already a privacy policy defined for that particular data consumer the data producer (that in that case could be the hospital) is notified of the pending access request and it is guided by the Privacy Requirements Elicitation Tool to define a privacy policy. Such a privacy policy defines if the *family doctor* has access to the event *blood test* and for which purpose (e.g. for healthcare treatment provisioning) and which part of the event he/she can access (e.g. the results regarding an AIDS test should be obfuscated).

It is important to note that the whole privacy policy elicitation process for the definition and storage of the privacy policies is centralized in the data controller but the resolution of a detail request is in part delegated to the source. More specifically, it is up to the data controller to find a privacy policy that matches a detail request (*policy matching phase*) and to apply the policy but it is let to the data source to return to the data controller a detailed event that contains only the data allowed by the policy (*policy enforcement*).

This approach has the following advantages:

- once the data sources have defined the privacy policies they do not need to keep track of the data consumers and data usage purposes as this is done by the data controller in charge of them. The data controller acts as guarantor and as certificated repository of the privacy policies
- it is never the case that data not accessible by a certain data consumer leaves the data producer
- the architecture is robust and flexible to changes in the representation of the privacy policies as the policy enforcement at the data controller is decoupled from the event details retrieval at the data producer
- it does not require any expertise to the data producers to define the privacy policies.

We assume that the partners are trusted parties and so we do not deal in this work with identity management. However, we plan to include as future extension of the infrastructure identity management mechanisms that are currently under development at the national level [7] for the identification of the specific users accessing the information, to validate their credentials and roles and to manage changes and revocation of authorizations in a policy. The security management mechanisms is also defined at the national level (e.g. PdD [7]) for message encryption and identity management and will be adopted when our solution will pass from a testing phase to the final release.

In the next two sections we describe more in details how the privacy policy elicitation and enforcement phases are performed.

5.1 Elicitation

As explain above the data consumer defines a privacy policy for each type of event (i.e. Event Details). We are using XACML to model internally to the Policy Enforcer module the privacy policies. According to the XACML notation [16], a *policy* is a set of *rules* with *obligations* where a rule specifies which *actions* a certain *subject* can perform on a specific *resource*. The *obligations* specifies which operations of the triggered policy should be executed at enforcing time (e.g. to obfuscate part of the resource). In our architecture an action correspond to a *purpose* of use (e.g. healthcare treatment, statistical analysis, administration).

A subject is an *actor* reflecting the particular hierarchical structure of the organization. For example, an actor could be a top level organization (e.g. '*Hospital S. Maria*') or a specific department inside it (e.g. '*Laboratory*', '*Dermatology*').

We consider an *event details* as a list of fields $e = \{f_1, \dots f_k\}$.

Definition 1. *Let* $E(D_i) = \{D_i.e_1, \ldots, D_i.e_n\}$ *be the classes of event details generated by the data producer* D_i *for* $i = 1, \ldots N$ *and let* $M_N(D_i.e_j)$ *and* $M_D(D_i.e_j)$ *be respectively the notification and detailed messages that are instances of the event* e_j *generated by* D_i*.*

We define *Events Catalog*, $E = \cup_{D_1,\ldots D_N} E_{D_i}$. Basically, the event catalog contains all the types of event details that the data producers could generate. Furthermore, we call *Events Index* the set of all the notification messages generated by the data producers $D_1, \ldots, D_N$, that is $I = \cup_{i=1}^{N} M_N D_i.e_j$.

For each type of event details and type of usage the data producer D defines a privacy policy.

Definition 2. *Let* $E(D) = \{D.e_1, \ldots, D.e_n\}$ *be the set of events a data source* D *could produce. We define* $P_D = \{p_1, \ldots, p_n\}$ *the set of privacy policies defined by* D *where* $p_i = \{A, e_j, S, F\}$ *such that:*

- A *is an actor that can ask for an event details*
- $e_j \in E$ *is a type of event details*
- S *is a set of purposes*
- F *is a set of fields where* $F \subseteq e_j$

Intuitively, a privacy policy indicates which fields F of an event details of type e_j could be accessed by actor A, for the purposes S. For example, the privacy policy $p = \{$ *National Governance, autonomy test, statistical analysis,* $\langle age, sex, autonomy\_score\rangle\}$ allows the statistical department of the *National Governance* to access to *age*, *sex* and *autonomy_score* for the event details of type *autonomy test* to perform statistical analysis on the needs of elderly people.

We apply the *deny-by default* approach so that unless permitted by some privacy policy an Event Details cannot be accessed by any subject. With this rule semantics the obligations specify which part of the Event Details is accessible by a certain subject for some purposes. Notice also that a subject can issue only read requests for an event type.

5.2 Enforcement

Definition 3. *Given a privacy policy* $p = \{A, e_j, S, F\}$ *and an event request* $r = \{A_r, \tau_e, S_r\}$ *we say that p is a* matching policy *for r if* $e_j = \tau_e \wedge A_r = A \wedge S_r \in S$.

Intuitively a policy matches a certain request if it refers to the same type of event details and actor and if the requested purpose of usage is allowed by the policy.

Definition 4. *Given the privacy policy* $p = \{A, e_j, S, F\}$ *and the event instance* e *of type* τ_e *we say that* e *is* privacy safe *for* p *wrt to the request* $r = \{A_r, \tau_e, S_r\}$*, i.e.* $e \models_r p$*, if* p *is a matching policy for* r *and* $\nexists f \in \tau_e$ *such that* $(e[f]$ *is not empty* $\wedge f \notin F)$ *where* $e[f]$ *is the value of* f *in* e*.*

Intuitively an event satisfies a privacy policy if it does not expose any field that is not allowed by the policy.

If an event instance e is privacy safe wrt to a request r for all the policies in a set P we write $e \models_r P$ meaning that $e \models_r p_i, \forall p_i \in P$.

Algorithm 1: GETEVENTDETAILS$(R) \mapsto e$

```
Require: $R = \{a, \tau_e, eID, s\} \neq \emptyset$
    $P$ set of policies defined by the data consumers
Ensure: $e \models_r P$
src_eID ⇐ $retrieveEventProducerId(eID)$
$\langle A, e_j, S, F \rangle \Leftarrow matchingPolicy(R)$
if $evaluate(\langle A, e_j, S, F \rangle, R) \equiv permit$ then
    return $getResponse(src\_eID, F)$
end if
return $deny$
```

Algorithm 2: GETRESPONSE$(src_eID, F) \mapsto e$

```
Require: $src\_eID \neq NULL, F \neq \emptyset$
Ensure: $e \models_r P$
$d \Leftarrow getEventDetails(src\_eID)$
return $parse(d, F)$
```

Privacy policies comes into play in two distinct moments of the events life-cycle and in particular at subscription time and at access time (*request for details* and *event index inquiry*).

In order to subscribe to a class of notification events the data consumer should be authorized by the data producer that means there should be a privacy policy regulating the access to the corresponding event details for that particular data consumer. If such a privacy policy is not defined then, according with the deny by default semantics, the subscription request is rejected. The inquiry of the event index is managed in the same way as also in this case the data consumer is asking for notifications.

The request for details resolution is more articulated and we will describe it more in depth with a focus on the specific architectural components involved.

A request for details requires to specify the type and identifier of the event to be obtained from the source. This information is contained in the notification message that is a pre-requisite to issue the request for details and grant that only the data consumer notified by a data producer can access the details. The notification is obtained either automatically by means of the pub/sub service offered by the infrastructure or by direct inquiry of the event index.

Figure 4 shows the internal components of the Policy Enforcer module in the data controller which are in charge of receive the request for details from the data consumer, retrieve the matching privacy policy associated to the Event Type and Event ID specified in the request, apply and evaluate the policy against the request and finally return a response with the result of the authorization decision. The result is an event details with values only for the fields authorized by the matching policy.

The components which constitute the Policy Enforcer are based on the XACML Specification.

Algorithm 1 ($getEventDetails(R)$) shows the actions performed by the policy enforcer to resolve an authorization request R issued by a data consumer a to access to the event with identifier eID and type τ_e for purpose s. The main steps are described below:

1. The authorization request is received by the Policy Enforcement Point (PEP). Through the Policy Information Point (PIP) it retrieves the corresponding local event ID (src_eID) valid in the data producer of the event. This mapping step is necessary as the event identifier distributed in the notification messages (eID) is a global artificial identifier generated by the data controller to identify the events independently from their data producers.

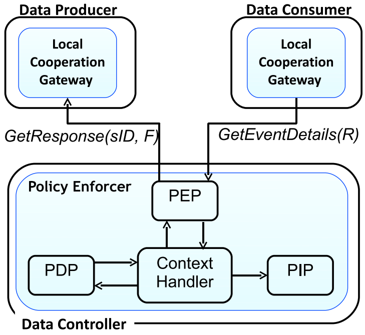
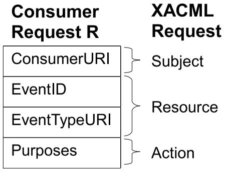


Fig. 4. Detail request resolution and privacy policy enforcement



Consumer Request R	XACML Request
ConsumerURI	Subject
EventID	Resource
EventTypeURI	
Purposes	Action

Fig. 5. Mapping in XACML request notation

2. The PEP sends the request to the Policy Development Point (PDP). The PDP retrieves the matching policy associated to the data producer, the data consumer and the resource: $\langle A, e_j, S, F\rangle$.
3. The PDP evaluates the matching policy and sends the result to the PEP. If there is no matching policy for the request or the evaluation fails the response will be *deny* and an *Access Denied message* is sent to the data consumer.
4. If the matching policy successfully evaluates the request (*permit decision*), the PEP asks the allowed part of the event details (F) to the data producer (i.e. the owner of the resource). Basically the producer applies the obligations in the policy.

Algorithm 2 shows the actions performed by the data producer in the $getResponse(src\_eID, F)$ method and in particular:

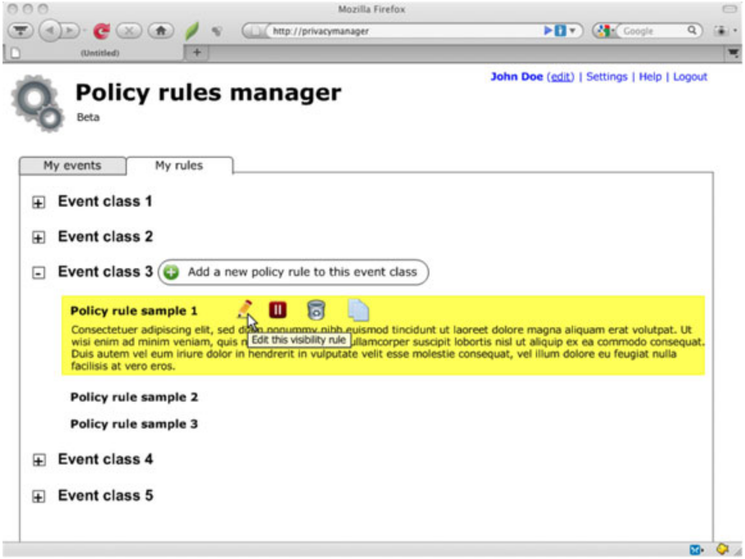
1. retrieves the Event Details from the internal events repository at the Local Cooperation Gateway;
2. parses the Event Details to filter out the values of the fields that are not allowed and produces the Privacy-Aware Event to be sent to the data consumer.

Notice that only the data accessible to the data consumer leaves the data producer. Fields that are not authorized are left empty. The data controller is a trusted entity which performs the application of policies, can trace the request of access and does the message routing between data producers and data consumers.

The architecture of the policy enforcer reflects XACML but the way we interact with the data producer and data consumer is independent from the underlying notation and enforcement strategy. As shown in Figure 5 the request for details of the data consumer is mapped to an XACML request by the policy enforcer. Similarly, the $getResponse$ method does not depend on the policy but only on the fields to be included in the event details.

6 Privacy Requirements Elicitation Tool

Figure 6 shows the Dashboard of the Privacy Rules Manager the data owner will use to define the privacy policies. She can define one or more privacy policy rule for each type


Mozilla Firefox
http://privacymanager
Google
(Untitled)
John Doe (edit) | Settings | Help | Logout
My events
My rules
Add a new policy rule to this event class
Edit this visibility rule

Policy rules manager

Beta

⊞ Event class 1

⊞ Event class 2

⊟ Event class 3

Policy rule sample 1

Consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis n ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros.

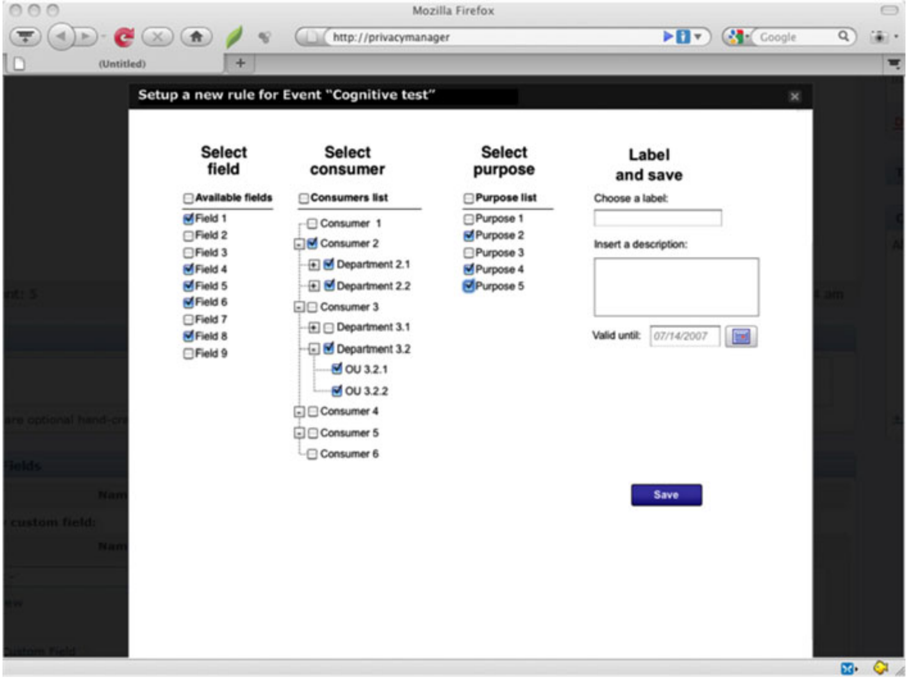
Policy rule sample 2

Policy rule sample 3

⊞ Event class 4

⊞ Event class 5

Fig. 6. Dashboard of the Privacy Rules Manager



Setup a new rule for Event "Cognitive test"

Select field

□ Available fields
- ☑ Field 1
- □ Field 2
- □ Field 3
- ☑ Field 4
- ☑ Field 5
- ☑ Field 6
- □ Field 7
- ☑ Field 8
- □ Field 9

Select consumer

□ Consumers list
- □ Consumer 1
- ☑ Consumer 2
  - ☑ Department 2.1
  - ☑ Department 2.2
- □ Consumer 3
  - □ Department 3.1
  - ☑ Department 3.2
    - ☑ OU 3.2.1
    - ☑ OU 3.2.2
- □ Consumer 4
- □ Consumer 5
- □ Consumer 6

Select purpose

□ Purpose list
- □ Purpose 1
- ☑ Purpose 2
- □ Purpose 3
- ☑ Purpose 4
- ☑ Purpose 5

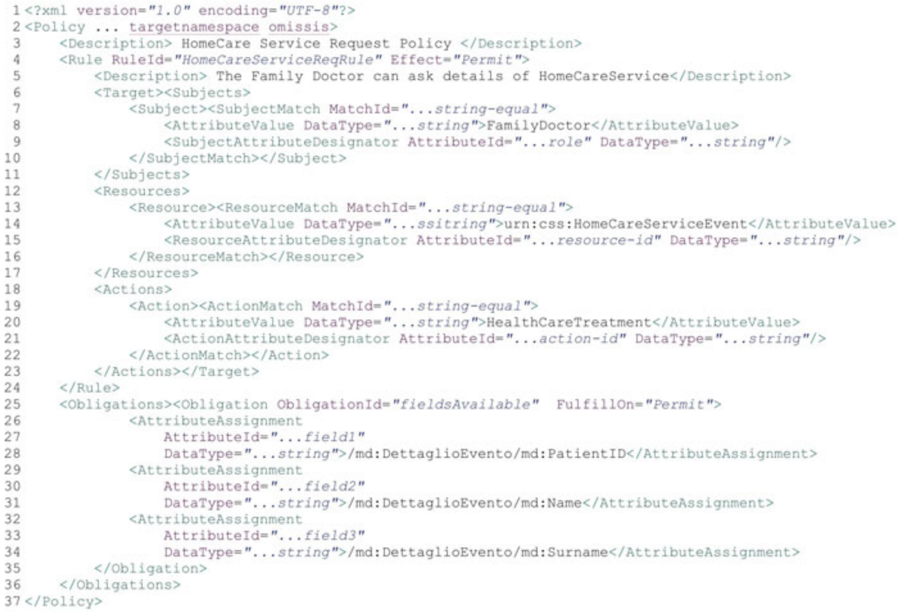
Label and save

Choose a label:

Insert a description:

Valid until: 07/14/2007

Fig. 7. Privacy Policy definition tool



```
 1<?xml version="1.0" encoding="UTF-8"?>
 2<Policy ... targetnamespace omissis>
 3    <Description> HomeCare Service Request Policy </Description>
 4    <Rule RuleId="HomeCareServiceReqRule" Effect="Permit">
 5        <Description> The Family Doctor can ask details of HomeCareService</Description>
 6        <Target><Subjects>
 7            <Subject><SubjectMatch MatchId="...string-equal">
 8                <AttributeValue DataType="...string">FamilyDoctor</AttributeValue>
 9                <SubjectAttributeDesignator AttributeId="...role" DataType="...string"/>
10            </SubjectMatch></Subject>
11        </Subjects>
12        <Resources>
13            <Resource><ResourceMatch MatchId="...string-equal">
14                <AttributeValue DataType="...ssitring">urn:css:HomeCareServiceEvent</AttributeValue>
15                <ResourceAttributeDesignator AttributeId="...resource-id" DataType="...string"/>
16            </ResourceMatch></Resource>
17        </Resources>
18        <Actions>
19            <Action><ActionMatch MatchId="...string-equal">
20                <AttributeValue DataType="...string">HealthCareTreatment</AttributeValue>
21                <ActionAttributeDesignator AttributeId="...action-id" DataType="...string"/>
22            </ActionMatch></Action>
23        </Actions></Target>
24    </Rule>
25    <Obligations><Obligation ObligationId="fieldsAvailable"  FulfillOn="Permit">
26            <AttributeAssignment
27                AttributeId="...field1"
28                DataType="...string">/md:DettaglioEvento/md:PatientID</AttributeAssignment>
29            <AttributeAssignment
30                AttributeId="...field2"
31                DataType="...string">/md:DettaglioEvento/md:Name</AttributeAssignment>
32            <AttributeAssignment
33                AttributeId="...field3"
34                DataType="...string">/md:DettaglioEvento/md:Surname</AttributeAssignment>
35        </Obligation>
36    </Obligations>
37</Policy>
```

Fig. 8. Example of Privacy Policy

of event. Figure 7 shows the UI for the definition of an instance of privacy policy. The user can select (i) one or more items from the list of fields in the event details type, (ii) whom (i.e. ore or more OUs as consumers) and (iii) the admissible purposes. Privacy rules are saved with a name and a description. Optionally the user can specify a validity date that limits the application of the rule for a certain time window. This option is particularly useful when private companies are involved in the care process and should access to the events of their customers only for the duration of their contract.
Some of the advantages of the UI are listed below:

- it is very intuitive to use as it does not require any knowledge of XACML but it asks to the user to define a policy in terms of actor, type of event to protect and admissible purposes of use;
- it automatically generates and store in a policy repository the privacy policy in XACML format.

In Figure 8 we provide an example e of a privacy policy that allows a user with role family doctor (lines 7-10) to access the event of type HomeCareServiceEvent (lines 13-16) for HealthCareTreatment purpose (line 20). In particular, only the fields PatientId, Name and Surname of the details are accessible (line 25-36).

7 Conclusions and Future Work

In this paper we proposed and implemented a solution that respects national standards both in application cooperation and in privacy. The approach allows us to couple the benefits of a pub/sub event-based system (decoupling of publishers and subscribers) with a privacy approach that is compliant with the privacy laws typically adopted in managing healthcare information, as validated with privacy compliance experts. The system provides services for policy definition and application but does not dig into the data that travels from sources to destinations and improves the control on data exchanges and consumption by validating and logging all data requests and exchanges. The informatization of data flows permits also to avoid privacy violations caused by manual importing into systems (operators does not input anymore data into systems). The privacy requirements elicitation tool is very intuitive and can be used by privacy experts that does not have sufficient knowledge on underlying DB schema.

We already develop all the components of the infrastructure (routing, storage and privacy aware details retrieval of the events) and tested it with sample data given by the data providers. The Web Interfaces for the definition of the privacy rules are still under development to take better into account the requirements of accessibility of the partners as shown in the mockup implementation of the User Interface in Figure 6 and 7.

The system can be used also directly by the citizens to specify and control their consent on data exchanges. This possibility will acquire more importance considering that the CSS is the backbone for the implementation of a Personalized Health Records (PHR) in Trentino.

Acknowledgment

This research work comes from our experience in the project CSS founded by the Autonomous Province of Trento. A special thank to Dr. Andrea Nicolini, project manager of the ICAR[1] project.

References

[1] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architecture and Applications. Springer, Heidelberg (2004)

[2] Anderson, A.H.: A comparison of two privacy policy languages: EPAL and XACML. In: SWS 2006: Proceedings of the 3rd ACM workshop on Secure web services, pp. 53–60. ACM, New York (2006)

[3] Breininger, K., McRae, M.: ebxml registry tc v3.0. Technical report, OASIS (2005)

[4] Canada health infoway, http://www.infoway-inforoute.ca/

[5] Chiasera, A., Casati, F., Florian, D., Velegrakis, Y.: Engineering privacy requirements in business intelligence applications. In: Jonker, W., Petković, M. (eds.) SDM 2008. LNCS, vol. 5159, pp. 219–228. Springer, Heidelberg (2008)

[6] Chou, S.-C., Huang, C.-H.: An extended xacml model to ensure secure information access for web services. J. Syst. Softw. 83(1), 77–84 (2010)

[1] http://www.progettoicar.it/

[7] CISIS. Inf-3: Sistema federato di autenticazione, `http://tinyurl.com/27yo92v`

[8] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for xml documents. ACM Trans. Inf. Syst. Secur. 5(2), 169–202 (2002)

[9] Dogac, A., Laleci, G.B., Kabak, Y., Unal, S., Heard, S., Beale, T., Elkin, P.L., Najmi, F., Mattocks, C., Webber, D., Kernberg, M.: Exploiting ebxml registry semantic constructs for handling archetype metadata in healthcare informatics. Int. J. Metadata Semant. Ontologies 1(1), 21–36 (2006)

[10] Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)

[11] IHE: IHE - integrating the healthcare enterprise xds profile, `http://www.ihe.net/profiles/`

[12] Personal data protection code. Italian privacy guarantor, Legislative Decree no. 196 dated 30 June 2003 (2003)

[13] Guidelines on the electronic health record and the health file. Italian privacy guarantor, Italy's Official Journal 71 dated 26 March 2009 (2009)

[14] Luo, B., Lee, D., Lee, W.-C., Liu, P.: Qfilter: fine-grained run-time xml access control via nfa-based query rewriting. In: CIKM 2004: Proceedings of the thirteenth ACM international conference on Information and knowledge management, pp. 543–552. ACM, New York (2004)

[15] Michelson, B.M.: Event-driven architecture overview event-driven soa is just part of the eda. Patricia Seybold Group (2006)

[16] Moses, T.: Extensible access control markup language tc v2.0 (xacml). Technical report, OASIS (2005)

[17] NHS-UK: Nhs connecting for health, `http://www.connectingforhealth.nhs.uk/`

[18] NICTIZ-AORTA: AORTA the dutch national infrastructure

[19] Schunter, M., Wenning, R. (eds.): The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Group Note (November 2006)

[20] ServiceMix, A.: Apache servicemix, `http://servicemix.apache.org/`

[21] W3C. Xmlschema (2001), `http://www.w3.org/2001/XMLSchema`

[22] Webber, D., Dutton, A.: Understanding ebxml, uddi, xml/edi. Technical report, XML Global Technologies Inc. (2000)

[23] Yagüe, M.: Survey on xml-based policy languages for open environments. Journal of Information Assurance and Security, 11–20 (2006)

Joining Privately on Outsourced Data

Bogdan Carbunar[1] and Radu Sion[2,⋆]

[1] Applied Research Center, Motorola Labs, Schaumburg, IL, USA
[2] Computer Science, Stony Brook University, Stony Brook, NY, USA

Abstract. In an outsourced database framework, clients place data management with specialized service providers. Of essential concern in such frameworks is data privacy. Potential clients are reluctant to outsource sensitive data to a foreign party without strong privacy assurances beyond policy "fine–prints". In this paper we introduce a mechanism for executing general binary JOIN operations (for predicates that satisfy certain properties) in an outsourced relational database framework with full computational privacy and low overheads – a first, to the best of our knowledge. We illustrate via a set of relevant instances of JOIN predicates, including: range and equality (e.g., for geographical data), Hamming distance (e.g., for DNA matching) and semantics (i.e., in health-care scenarios – mapping antibiotics to bacteria). We experimentally evaluate the main overhead components and show they are reasonable. For example, the initial client computation overhead for 100000 data items is around 5 minutes. Moreover, our privacy mechanisms can sustain theoretical throughputs of over 30 million predicate evaluations per second, even for an un-optimized OpenSSL based implementation.

1 Introduction

Outsourcing the "database as a service" [24] emerged as an affordable data management model for parties ("data owners") with limited abilities to host and support large in-house data centers of potentially significant resource footprint. In this model a *client* outsources its data management to a *database service provider* which provides online access mechanisms for querying and managing the hosted data sets.

Because most of the data management and query execution load is incurred by the service provider and not by the client, this is intuitively advantageous and significantly more affordable for parties with less experience, resources or trained man-power. Compared with e.g., a small company, with likely a minimal expertize in data management, such a database service provider intuitively has the advantage of expertize consolidation. More-over it is likely to be able to offer the service much cheaper, with increased service availability and uptime guarantees.

[⋆] Sion is supported by the National Science Foundation through awards CCF 0937833, CNS 0845192, IIS 0803197, and by CA Technologies, Xerox, IBM and Microsoft.

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 70–86, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Significant security issues are associated with such “outsourced database” frameworks, including communication-layer security and data confidentiality. Confidentiality alone can be achieved by encrypting the outsourced content. Once encrypted however, the data cannot be easily processed by the server. This limits the applicability of outsourcing, as the type of processing primitives available will be reduced dramatically.

Thus, it is important to provide mechanisms for server-side data processing that allow both confidentiality and a sufficient level of query expressibility. This is particularly relevant in relational settings. Recently, protocols for equijoin and range queries have been proposed [34,35,33].

Here we go one step further and provide low overhead solutions for *general* binary JOIN predicates that satisfy certain properties: for any value in the considered data domain, the *number* of corresponding “matching” pair values (for which the predicate holds) is upper bound. We call these finite match predicates (FMPs).

Such predicates are extremely common and useful, including any discrete data scenarios, such as ranges, inventory and company asset data sets, forensics, genome and DNA data (e.g., fuzzy and exact Hamming distances), and healthcare databases (e.g., bacteria to antibiotics matches). Moreover, at the expense of additional client-side processing (pruning of false positives) other predicate types (multi-argument, continuous data) can be accommodated.

While on somewhat orthogonal dimensions, it might be worth noting that other important challenges are to be considered in the framework of database outsourcing. Transport layer security is important as eavesdropping of data access primitives is unacceptable. This can be achieved by deploying existing traditional network security protocols such as IPSec/SSL. Moreover, query correctness issues such as authentication and completeness are important and have been previously considered.

The main contributions of this paper include: (i) the proposal and definition of the problem of private joins for generalized query predicates, (ii) a solution for FMPs, (iii) its analysis, (iv) a proof-of-concept implementation and (v) the experimental evaluation thereof.

The paper is structured as follows. Section 2 introduces the main system, data and adversary models. Section 3 overviews, details and analyzes our solution. Section 4 discusses predicate instance examples and the handling thereof. Section 5 introduces our proof-of-concept implementation and provides an experimental analysis thereof. Section 6 surveys related work and Section 7 concludes.

2 Model

We choose to keep the data outsourcing model concise yet representative. Sensitive data is placed by a client on a database server situated at the site and under the control of a *database service provider*. Later, the client can access the outsourced data through an online query interface exposed by the server. Network layer confidentiality is assured by mechanisms such as SSL/IPSec. This corresponds to a *unified client model* [14,33]. Clients would like to allow the server to process data queries while maintaining data confidentiality. For this purpose, they will encrypt data before outsourcing. As encrypted data is hard to process without revealing it, to allow for more expressive server-side data processing, clients will also pre-process data according to a set of supported (join) predicates. They will then outsource additional associated metadata to aid the server in processing tasks. This metadata, however, will still be "locked" until such processing tasks are requested by the client.

Later, to allow server – side data processing, the client will provide certain "unlocking" information for the metadata associated with the accessed items. The server will perform *exactly* the considered query (and nothing more) without finding out any additional information.

It is important for the outsourced metadata not to reveal any information about the original data. Additionally, the computation, storage and network transfer overheads should maintain the cost advantages of outsourcing, e.g., execution times should not increase significantly. We consider a relational model, where we consider the outsourced data as a set of t data columns (e.g., relational attributes), D stored on the server. Let n be the average number of values stored in a column and b be the number of bits in the representation of a value. Naturally, we allow relations to contain variable number of tuples. We use this notation for analysis purposes only.

Finite Match Predicates (FMPs). In this paper we consider binary predicates $p : \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{B} = \{true, false\}$ for which the "match sets" $P(x) := \{y|p(x, y) = true\}$ can be computed efficiently and their size (taken over all encountered values of x) upper bound. In other words, given a certain value x in the considered data domain, its "matching" values are easily determined and their number is upper bound. We call these predicates *finite match* predicates (FMPs). We call the number of matching values *maximum match size* (MMS). For instance, consider the following discrete time – range join query that joins arrivals with departures within the same 30 mins interval (e.g., in a train station):

```
SELECT * FROM arrivals,departures
WHERE ABS(arrivals.time - departures.time) > 30
```

In this example, the FMP has an MMS of 60.

Privacy Requirements. In the considered adversarial setting, the following privacy requirements are of concern.

Initial Confidentiality. The server should not be able to evaluate inter-column (join) predicates on the *initially* received data without ("un-lock") permission from the client.

Predicate Safety. The server should not be able to evaluate predicates on "unlocked" data. This also implies that no additional information should be leaked in the process of predicate evaluation. For instance, allowing the evaluation of predicate $p(x, y) := (|x - y| < 100)$, should not reveal $|x - y|$.

We stress that here we do not provide confidentiality of predicates, but rather just of the underlying target data. We also note that we do not consider here the ability of the server to use out of band information and general knowledge about the data sets to infer what the underlying data and the query results look like. In fact we envision a more formal definition in which privacy guarantees do not allow any leaks to the server beyond exactly such inferences that the curious server may do on its own based on outside information.

Performance Constraints. The main performance constraint we are interested in is *maintaining the applicability of outsourcing*. In particular, if a considered query load is more efficient (than client processing) in the unsecured data outsourcing model – then it should still be more efficient in the secured version. We believe this constraint is essential, as it is important to identify solutions that validate in real life. There exist a large number of apparently more elegant cryptographic primitives that could be deployed that would fail this constraint. In particular, experimental results indicate that *predicate evaluations on the server should not involve any expensive (large modulus) modular arithmetic such as exponentiation or multiplication*. We resisted the (largely impractical) trend (found in existing research) to use homomorphisms in server side operations, which would have simplified the mechanisms in theory but would have failed in practice due to extremely poor performance, beyond usability.

supported by recent findings that show the total cost of storage management is orders of magnitude higher than the initial storage equipment acquisition costs [17].

Adversary. We consider an *honest but curious* server: given the possibility to get away undetected, it will attempt to compromise data confidentiality (e.g., in the process of query execution). The protocols in this paper are protecting mainly data *confidentiality*. The server can certainly choose to deny service by explicitly not cooperating with its clients, e.g., by not returning results or simply closing connections.

2.1 Tools

Encryption, Hashing and Random numbers. We consider ideal, collision-free hashes and strongly un-forgeable signatures. While, for reference purposes we benchmark RC4 and AES in section 5, we will not be more specific with respect to its nature and strength as it is out of scope here. We note that our solution does not depend on any specific encryption mechanism. We will denote by $E_K(v)$ the encryption of value v with key secret key K. If not specified, the key K will be implicitly secret and known only to the client. In the following, we use the notation $x \hookrightarrow_R S$ to denote x's uniformly random choice from S.

Bloom Filters. Bloom filters [8] offer a compact representation of a set of data items, allowing for fast set inclusion tests. Bloom filters are *one-way*, in that, the "contained" set items cannot be enumerated easily (unless they are drawn from a finite, small space). Succinctly, a Bloom filter can be viewed as a string of l bits, initially all set to 0. To *insert* a certain element x, the filter sets to 1 the bit values at index positions $H_1(x), H_2(x), \ldots, H_h(x)$, where $H_1, H_2, \ldots, H_h$

are a set of h crypto-hashes. Testing set inclusion for a value x is done by checking that the bits for *all* bit positions $H_1(x), H_2(x), \ldots, H_h(x)$ are set. By construction, Bloom filters feature a controllable rate of false positives (p_{fp}) for set inclusion tests. For a certain number N of inserted elements, there exists a relationship that determines the optimal number of hash functions h_o minimizing p_{fp}: $h_o = \frac{l}{N}\ln 2 \approx 0.7\frac{l}{N}$ which yields a false positive probability of $p_{fp} = \left(\frac{1}{2}\right)^{h_o} = \left(\frac{1}{2}\right)^{\frac{l}{N}\ln 2} \approx 0.62^{l/N}$ For a Bloom filter BF, we denote $BF.insert(v)$ the insertion operation and $BF.contains(v)$ the set inclusion test (returning *true* if it contains value v, $false$ otherwise).

For an excellent survey on applications on Bloom filters and their applications in a variety of network problems please see [10].

Computational Intractability Assumptions. Let $\mathbb{G}$ be a finite field of size p prime and order q and let g be a generator for $\mathbb{G}$. The Computational Diffie-Hellman assumption (CDH) [21]:

Definition 1. *(CDH Assumption) given g, g^a mod p and g^b mod p, for $a, b \in \mathbb{Z}_q$, it is computationally intractable to compute the value g^{ab} mod p.*

In the same cyclic group $\mathbb{G}$, the Discrete Logarithm assumption (DL):

Definition 2. *(DL Assumption) given $g, v \in \mathbb{G}$, it is intractable to find $r \in \mathbb{Z}_q$ such that $v = g^r$ mod p.*

3 Outsourced JOINs with Privacy

We define the arbitrary (non hard-coded to a specific application) predicate join solution to be a quadruple $(pred_{FM}, G, E, J)$, where $pred_{FM}$ is the FMP, G is a parameter generation function, E is a data pre-processing function and J denotes a joining function according to predicate $pred_{FM}$. G and E are executed by the client and the output of E is outsourced to the server. J is executed by the server on two attributes of the client's data. In this section we provide a general description of the G, E and J functions and in Section 4 we study two predicate and corresponding G, E and J function instances. In Figure 1 we summarize the symbols used in our solution.

p	prime number
N	bit size of p
$\mathbb{G}$	subgroup of $\mathbb{Z}_p$
q	order of $\mathbb{G}$
g	generator of $\mathbb{G}$
x_A, y_A	secret values for column A

Fig. 1. Symbol Table

G is a parameter generation operation executed initially by the client. Its input is N, a security parameter and t, the number of columns in the client database D. Let $p = 2p' + 1$ be a N bit long prime, such that p' is also prime. The reason for this choice is to make the CDH assumption harder. Let $\mathbb{G} = \mathbb{Z}_p$ be a group of order q, with a generator g.

$G(N, t)$. Generates an encryption key $K \hookrightarrow_R \{0,1\}^*$. For each column $A \in D$, generate two values $x_A, y_A \hookrightarrow_R \mathbb{Z}_q$, $x_A \neq y_A$. Publish p and g and keep secret the key K and the values x_A and y_A, for all columns $A \in D$.

E is executed by the client, after running G. It takes as input a column $A \in D$, the key K and the secret values x_A and y_A corresponding to column A.

$E(A, K, x_A, y_A)$. Associate with each element $a_i \in A$, $i = 1..n$ a Bloom filter denoted $BF(a_i)$, with all the bits initially set to 0. Let $P(a_i) = \{v | pred_{FM}(a_i, v) = true\}$ be the set of values that satisfy the predicate $pred_{FM}$ for element a_i. For each $a_i \in A$, encrypt a_i with the key K, producing $E_K(a_i)$ and compute an "obfuscation" of a_i, $O(a_i) = H(a_i)x_A \ mod \ q$. Then, $\forall v \in P(a_i)$, compute $e_A(v) = g^{H(v)y_A} \ mod \ p$ and insert them into a_i's Bloom filter $(BF(a_i).insert(e_A(v)))$. Finally, output the values $E_K(a_i)$, $O(a_i)$ and $BF(a_i)$. Let D_T denote the output of E for all the columns in D. The client stores D_T on the server. Hence, element $a_i \in A$ is stored on the server as $D_T(A, i) = [E_K(a_i), \ O(a_i), \ BF(a_i)]$.

We now describe the join operation, J, executed by the server. J takes as input two column names A, B, a desired predicate $pred_{FM}$ and a trapdoor value (computed and sent by the client) $r_{AB} = g^{y_A/x_B} \ mod \ p$ and outputs the result of the join of the two columns on the predicate.

$J(A, B, pred_{FM}, r_{AB})$. For each element $b_j \in B$, compute $e_A(b_j) = r_{AB}^{O(b_j)} \ mod \ p$. For each element $a_i \in A$, iff. $BF(a_i).contains(e_A(b_j))$ return the tuple $\langle E_K(a_i), E_K(b_j) \rangle$.

In real life, J will output also any additional attributes specified in the SELECT clause, but for simplicity we explicit here and in the following only the join attributes.

Properties. We now list the security properties of this solution, whose proofs will be included in an extended version of the paper.

Theorem 1. *(Correctness) The join algorithm J returns all matching tuples.*

Theorem 2. *The $(pred_{FM}, G, E, J)$ solution satisfies the* initial confidentiality *requirement outlined in Section 2.*

Theorem 3. *$(pred_{FM}, G, E, J)$ is* predicate safe.

Notes on Transitivity. Under certain circumstances the server may use our solution to perform transitive joins. That is, provided with information to join A with B and later to join B with C, it can join A and C. We make the observation that on certain FMPs any solution will allow the server to perform partial transitive joins, using the outcome of previous joins. That is, when an element $b \in B$ has matched an element $a \in A$ and an element $c \in C$, the server can infer that with a certain probability a also matches c. In conclusion, we believe the transitive join problem to be less stringent than reducing server-side storage and computation overheads.

Same-column Duplicate Leaks. In the case of duplicate values occurring in the same data column, a data distribution leak can be identified. The deterministic nature of the obfuscation step in the definition of E associates the same obfuscated values to duplicates of a value. Upon encountering two entries with the same obfuscated value, the server indeed can infer that the two entries are identical. We first note that if joins are performed on primary keys this leak does not occur. Additionally, it is likely that in many applications this is not of concern. Nevertheless, a solution can be provided, particularly suited for the case when the number of expected duplicates can be upper bound by a small value (e.g., m). The deterministic nature of $O(a_i)$ is required to enable future Bloom filter lookups in the process of predicate evaluation. However, as long as the predicate evaluation is designed with awareness of this, each duplicate can be replaced by a unique value. This can be achieved by (i) populating Bloom filters with multiple different "variants" for each value expected to occur multiple times, and (ii) replacing each duplicate instance with one of these variants instead of the actual value. These variants can be constructed for example by padding each value with different $log_2(m)$ bits.

Bloom Filter Sizes. In the case of a join, the false positive rate of Bloom filters implies that a small percentage of the resulting joined tuples do *not* match the predicate the join has been executed for. These tuples will then be pruned by the client. Thus, a trade-off between storage overhead and rate of false positives (and associated additional network traffic) emerges. For example, for a predicate MMS $= N = 60$ (e.g., in the simple query in Section 2), a desired false positive rate of no more than $p_{fp} = 0.8\%$, the equations from Section 2.1 can be used to determine one optimal setup $l = 600$ and $h = 7$.

Data Updates and Multiple Clients. In data outsourcing scenarios, it is important to handle data updates incrementally, with minimal overheads. In particular, any update should not require the client to re-parse the outsourced data sets in their entirety. The solution handles data updates naturally. For any new incoming data item, the client's pre-processing function E can be executed per-item and its results simply forwarded to the server. Additionally, in the case of a multi-threaded server, multiple clients (sharing secrets and keys) can access the same data store simultaneously.

Complex, Multi-predicate Queries. Multiple predicate evaluations can be accommodated naturally. Confidentiality can be provided for the attributes involved in binary FMPs. For example, in the following database schema, the association between patients and diseases is confidential but any other information is public and can be used in joins. To return a list of Manhattan-located patient names and their antibiotics (but not their disease) the server will access both confidential (disease) and non-confidential (name,zip-code) values.

```
SELECT patients.name,antibiotics.name
FROM patients,antibiotics
WHERE md(patients.disease,antibiotics.name)
AND patients.zipcode = 10128
```

Only the predicate $md()$ will utilize the private evaluation support. This will be achieved as discussed above, by encrypting the `patients.disease` attribute and generating metadata for the `antibiotics` relation (which contains a list of diseases that each antibiotic is recommended for).

4 Predicate Instances

To illustrate, we choose to detail two predicate instances: a simple, range join and a Hamming distance predicate requiring custom predicate-specific extensions.

4.1 Range JOIN

Consider the binary FMP $p(x, y) := (v_1 \leq (x - y) \leq v_2)$ where $x, y \in \mathbb{Z}$. An instance of this predicate is the following travel agency query, allocating buses to trips, ensuring 5 (but no more than 10) last-minute empty slots per trip:

```
SELECT buses.name,trips.name
FROM buses,trips
WHERE (buses.capacity-trips.participants) >= 5
AND (buses.capacity-trips.participants) <= 10
```

Executing such a query remotely with privacy can be achieved efficiently by deploying the solution presented in Section 3. The parameter generation algorithm, G and the join algorithm J will be the same. As above, the data encoding algorithm encodes in the Bloom filter $BF(a_i)$ of element a_i all integer values in $P(a_i) := \{y | p(a_i, y) = true\}$ namely with values $\in [x - v_2, x - v_1]$. Note that given the size of the range, n and a fixed probability of false positives, p_{fp}, we have that the optimum Bloom filter size is $l = -\frac{n \ln p_{fp}}{(\ln 2)^2}$.

4.2 Hamming JOIN

It is often important to be able to evaluate Hamming distance on remote data with privacy in un-trusted environments. This has applications in forensics, criminal investigation (e.g. fingerprints), biological DNA sequence matching, etc.

Let x and y be b bit long strings and let $0 < d < b$ be an integer value. We use $d_H(x, y)$ to denote the Hamming distance of x and y. We consider the join predicate $pred_{FM}(x, y) := (d_H(x, y) \leq d)$. An example is the following fingerprint matching query that retrieves the names and last dates of entry for all individuals with physical fingerprints (in some binary representation) close enough to the ones of suspects on the current FBI watch list:

```
SELECT watchlist.name,
       immigration.name,
       immigration.date
FROM watchlist,immigration
WHERE Hamming(watchlist.fingerprint,
              immigration.fingerprint)<5
```

A private execution of this join operation can be deployed using the solution introduced in Section 3. The implementation of the Hamming part of the predicate requires specific adjustments. In particular, in pre-processing, the client pseudo-randomly bit-wise permutes all the data elements consistently. It then splits each data element into β equal sized blocks, where β is an input parameter discussed later. Then, for each such block, it generates three data items: one item will allow later private comparisons with other blocks for equality (Hamming distance 0). The other two (a Bloom filter and a "locked" obfuscated value) will be used by the server to identify (with privacy) blocks at Hamming distance 1. In the following we describe the (d_H, G_H, E_H, J_H) solution, as an extension of the solution presented in Section 3.

The parameter generator, G_H, takes two additional parameters, β and b. b is the bit length of elements from D and β is the number of blocks into which each data element is split. We assume $\beta > d$ is constant, much smaller than the number of elements stored in a database column. Possible values for β are investigated later in this section.

G_H**(N,t,**β**,b).** Choose a value $s \hookrightarrow_R \{0,1\}^*$ and generate a secret pseudo-random permutation $\pi : \{0,1\}^b \rightarrow \{0,1\}^b$. For each data column $A \in D$ [1], compute $s_A = H(s, A)$. Use s_A to seed a pseudo-random number generator PRG. Use PRG to generate 3β secret, duplicate-free pseudo-random values $x_A(1), .., x_A(\beta)$, $y_A(1), .., y_A(\beta)$, $z_A(1), .., z_A(\beta) \hookrightarrow_R \mathbb{Z}_q$.

$E_H(A, K, x_A(k), y_A(k), z_A(k)), k = 1..\beta, A \in D$**.** For each element a_i, $i = 1..n$ of A, compute a_i's *bit-wise* permutation $\pi(a_i)$, then split $\pi(a_i)$ into β blocks of equal bit length, $a_{i1}, .., a_{i\beta}$. For each block a_{ik}, $k = 1..\beta$, generate an obfuscated value $O(a_{ik}) = H(a_{ik})x_A(k) \ mod \ q$. Then, create a_{ik}'s Bloom filter by generating all values v for which $d_H(a_{ik}, v) = 1$. That is, generate all values with Hamming distance 1 from block a_{ik}. For each value v, let $e_A^k(v) = g^{H(v)y_A(k)} \ mod \ p$. Encode $e_A^k(v)$ into a_{ik}'s Bloom filter, using operation $BF(a_{ik}).insert(e_A)^k(v))$. Compute an additional structure allowing the server to assess (with privacy) equality of the kth block of a_i with the kth blocks of other values, $Z(a_{ik}) = H(a_{ik})z_A(k) \ mod \ q$. Finally, output $[E_K(a_i)$, $O(a_{ik})$, $Z(a_{ik})$, $BF(a_{ik})]$, for all $k = 1..\beta$. Hence element a_i is stored on the server as a tuple $D_T(A, i) = [E_K(a_i), O(a_{ik}), Z(a_{ik}), BF(a_{ik})]$, similar to the solution in Section 3.

To join two columns A and B on predicate $pred_{FM}$, J_H receives the following 3β trapdoor values from the client (3 for each block) (i) $r_A(k) = g^{R_k/z_A(k)} \ mod \ p$, (ii) $r_B(k) = g^{R_k/z_B(k)} \ mod \ p$ and (iii) $r_k = g^{y_A(k)/x_B(k)} \ mod \ p$, for $k = 1..\beta$, where $R_k \hookrightarrow_R \{0,1\}^*$ (generated at the client side).

$J_H(A, B, r_A(k), r_B(k), r_k), k = 1..\beta$**.** For each element a_i from A and for each $k = 1..\beta$, compute $v(a_{ik}) = r_A(k)^{Z(a_{ik})} \ mod \ p$. For each element b_j from B and for each $k = 1..\beta$, compute $v(b_{jk}) = r_B(k)^{Z(b_{jk})} \ mod \ p$. For each element

[1] A here is the column's unique server-side name.

$b_j \in B$ and each element $a_i \in A$, set counter c to 0. For each $k = 1..\beta$, if $BF(a_{ik}).contains(r_k^{O(b_{jk})})$ then do $c = c + 1$ and $k = k + 1$. Else, if $v(a_{ik}) = v(b_{jk})$, do $k = k + 1$. Otherwise, move to the next element, a_{i+1}, from A. If at the end of the k loop, $c < d$, return $\langle E_K(a_i), E_K(b_j)\rangle$. Else, move to the next element from A, a_{i+1}.

Note that for future query purposes the client does not need to remember the values $(x_A(k)$, $y_A(k)$, $z_A(k))$ for each column A. Instead, it generates them by seeding its PRG with s_A. For this, the client only needs to store one value, s.

Theorem 4. *Any given pair of elements from A and B at Hamming distance less than or equal to d is found with probability at least* $e^{-d/\beta}(1 + \frac{d}{\beta})$.

Arbitrary Alphabets. The above solution can also be deployed for an arbitrary alphabet, that is, when the elements stored in the database D are composed of symbols from multi-bit alphabets (e.g., DNA sequences). This can be done by deploying a custom binary coding step. Let $\mathcal{A} = \{\alpha_0, .., \alpha_{u-1}\}$ be an alphabet of u symbols. In the pre-processing phase, the client represents each symbol over u bits ($u/\log u$-fold blowup in storage), such that symbol $\alpha_u = 2^i$. That is, $d_H(\alpha_i, \alpha_j)$ is 1 if $i \neq j$ and 0 otherwise. If each data item has b symbols, each of the item's blocks will have bu/β bits, and, due to the coding, pairs of elements of symbol-wise distance d will have a $2d$ bit-wise Hamming distance. Thus, after the coding phase, the above algorithm can be deployed without change. As an example, for an alphabet of 4 symbols {A,C,G,T}, the following encoding will be used {A=0001,C=0010,G=0100,T=1000}. To compare the strings ACG and ACT (alphabet distance 2), the following two binary strings will be compared instead: 000100100100 and 000100101000 (binary Hamming distance 2).

5 Experimental Results

Implementation Details. We conducted our experiments using a C/C++ implementation of the private predicate join algorithms, on 3.2GHz Intel Pentium 4 processors with 1GB of RAM running Linux. We implemented the cryptographic primitives using OpenSSL 0.9.7a. Our goal was to investigate the feasibility of the algorithms in terms of computation, communication and storage overheads, both on the client and the server side.

To understand the costs of encryption and hashing, we have evaluated several symmetric encryption and crypto-hashing algorithms. In our setup we benchmarked RC4 at just bellow `80MB/sec`, and MD5 to of up to `150MB/sec`, shown in Figure 2(a). We also benchmarked integer hashing throughput at more than 1.1 million MD5 hashes per second, showing the "startup" cost of hashing.

As recommended by the Wassenaar Arrangement [31], we set N, the size of the prime p to be 512 bits and the size of the prime q to be 160 bits. From our benchmarks, shown in Figure 2(b), we have concluded that 512-bit modular exponentiations (with 160 bit exponents) take 274usec while 512-bit modular multiplications take only 687nsec.

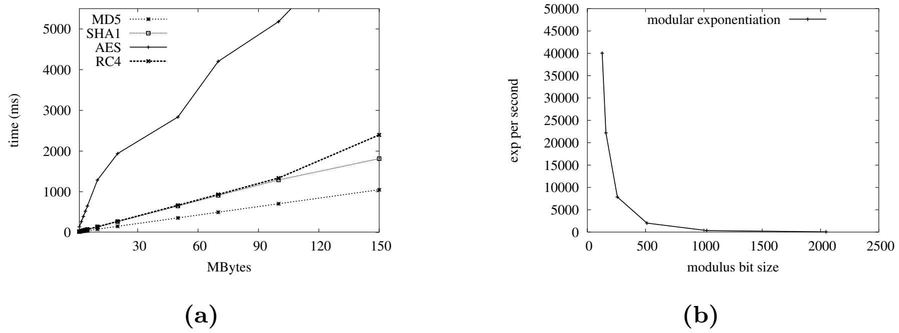
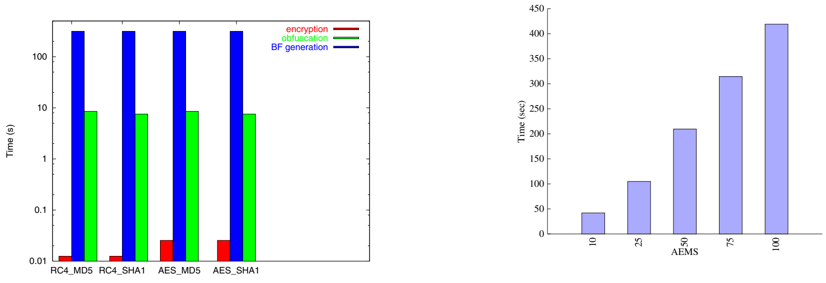


Fig. 2. Overheads of cryptographic operations

We have considered three types of applications for the private join algorithms. In a first application we used SNPs (single nucleotide polymorphisms) from a human DNA database [2]. An SNP is a variation of a DNA sequence that differs from the original sequence in a single position. The goal of a join is to identify all pairs of sequences from two columns, that differ in a single position. To achieve this, the Bloom filter of a DNA sequence contains all the sequence's SNPs. Since SNPs from [2] have 25 nucleotides, each from the set A, T, C, or G, a Bloom filter stores 75 values (MMS=75). Our second application performs fingerprint matching, that is, identifying similar pairs of fingerprints. We have used fingerprint data from [1] where each fingerprint consists of 100 features. For this application we considered only fingerprints that differ in at most one feature to be a match, thus, Bloom filters store 100 values (MMS=100). The last application identifies picture similarities, using digital images from the LabelMe [42] and Caltech 101 [16] databases. A set of images are annotated with scores for lightness, hue or colors of interest [15,19]. The Bloom filter associated with an image contains score ranges of interest, which for this application was set to 100 values around the image's score (MMS=100). To compare two images for similarity, the score of one image is searched in the Bloom filter associated with the other image.

Client Computation Overheads. We now describe our investigation of the initial client pre-processing step. Of interest were first the computation overheads involved in generating the encryption, obfuscation and Bloom filter components associated with a database of 100000 elements of 16 bytes each. We experimented with four combinations of encryption algorithms (RC4 and AES) and hashing algorithms (MD5 and SHA1), in a scenario where Bloom filters store 75 items each. Figure 3(a) depicts our results (log scale time axis). For each encryption/hash algorithm combination shown on the x axis, the left hand bar is the encryption cost, the middle bar is the Bloom filter generation cost and the right hand bar is the obfuscation cost. Our experiments show the dominance of the Bloom filter generation, a factor of 30 over the combined encryption and



(a) **(b)**

Fig. 3. Client computation overheads

obfuscation costs. The total computation cost of each implementation is roughly 320 seconds with the minimum being achieved by RC4/MD5. We further investigated the RC4/MD5 combination by increasing the MMS value from 10 to 100. Figure 3(b) shows that the pre-processing overhead increase is linear in the MMS value. The total costs range between 40 seconds (MMS=10) and 7 minutes (MMS=100). We stress that this cost is incurred by the client only once, when computing the initial data structures.

Server Computation Costs. In order to evaluate the performance of the private join algorithm we used columns of 10000 images each, collected from the LabelMe [42] and Caltech 101 [16] databases. For each image we deployed 1024-bit Bloom filters ($h = 12$ hashes) with MMS=75. The join operation returns all pairs of images that have scores within a given range of each other. In our implementation, for each element from one column we perform a 512-bit modular exponentiation with a 160 bit modulus, followed by a crypto-hash, fragment the result into 12 parts and use each part as a bit position into each of the Bloom filters associated with the elements of the other column.

As, to the best of our knowledge no other solutions exist for arbitrary private joins on encrypted data, we chose to compare our solution against a hypothetical scenario which would use the homomorphic properties of certain encryption schemes such as Paillier [38]. This comparison is motivated by recent related work (e.g., [18]) that deploy this approach to answer SUM and AVG aggregation queries on encrypted data. Moreover, we also considered the cost of solutions that would use RSA encryptions or decryptions to perform private joins.

Figure 4(a) compares our solution against (i) C_P, that performs one modular multiplication within the Paillier cryptosystem with a 1024-bit modulus, for every two elements that need to be compared, (ii) C_{enc}, that uses one 1024-bit RSA encryption for each comparison and (iii) C_{dec}, that uses one 1024-bit RSA decryption operation. The y axis represents the time in logarithmic scale. The first bar shows the performance of our FMP join algorithm. The cost is dominated by $10^8 \times 12$ verifications of Bloom filter bit values (the cost of computing 10^4 hashes and exponentiations (modulo a 512-bit prime) is under

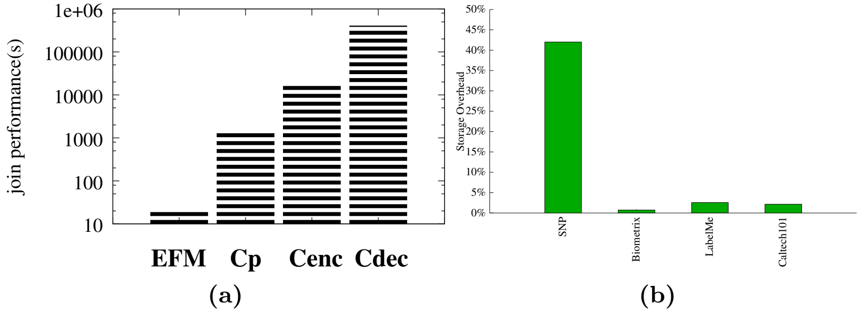


Fig. 4. (a) Join costs for columns of 10000 elements. Our solution is 2-4 orders of magnitude faster than other solutions that use 1024-bit modular operations (b) Bloom filter storage overhead as percentage of the size of the cleartext data. The overhead is 42% for SNP databases, but under 3% for fingerprint or image databases.

3.5s). With a 21.3s computation overhead, the FMP join solution performs two orders of magnitude faster than C_P (second bar) taking 1525s, three orders of magnitude faster than C_{enc} (third bar), taking 19168s and four orders of magnitude faster than C_{dec} (fourth bar), taking 408163s. One reason for the large overhead of the modular multiplications in the Paillier system (used also in [18]) is the fact that while the modulus n has 1024 bits, the multiplications are actually performed in the space $\mathbb{Z}^*_{n^2}$. That is, the active modulus has 2048 bits. Using less than 1024 bits for n is not recommended [3,31].

Storage Overhead. Since we use symmetric encryption algorithms, the size of the E values stored on the server is roughly the same as the original size of the elements – thus no significant overhead over stioring the cleartext data. The size of the O value for each element is $N = 512$ bits, which is small and data-independent. Finally, Figure 4(b) shows the overhead of the 1024 bit Bloom filters as a percentage of the size of the original data. The largest overhead is 42%, for the SNP database, due to the smaller size of SNPs. However, for image databases, the overhead is under 3% and for fingerprints is under 1%.

Transfer Overhead. We have measured the communication overhead of the initial database transfer between sites located in Chicago and New York, more than a thousand miles apart. With the bottleneck being the uplink capacity of the client, of around 3 Mbps, the overhead of transferring the Bloom filters associated with 100000 items was roughly 32 seconds.

6 Related Work

Extensive research has focused on various aspects of DBMS security and privacy, including access control and general information security issues [5,4,6,7,12,13,25,26,28,29,32,36,37,39,40]. Statistical and *Hippocratic* databases aim to address the problem of allowing aggregate queries on confidential data (stored on trusted servers) without leaks [4,5,12,13,30]. Hacigumus et al.[23] introduced a method for executing SQL queries over partly obfuscated outsourced data. The data is divided into secret partitions and queries over the original data can be rewritten in terms of the resulting partition identifiers; the server can then partly perform queries directly. The information leaked to the server is 1-out-of-s where s is the partition size. This balances a trade-off between client and server-side processing, as a function of the data segment size. At one extreme, privacy is completely compromised (small segment sizes) but client processing is minimal. At the other extreme, a high level of privacy can be attained at the expense of the client processing the queries in their entirety. Similarly, Hore et al. [27] deployed data partitioning to build "almost"-private indexes on attributes considered sensitive. An untrusted server is then able to execute "obfuscated range queries with minimal information leakage". An associated privacy-utility trade-off for the index is discussed.

Ge and Zdonik [18] have proposed the use of a secure modern homomorphic encryption scheme, to perform private SUM and AVG aggregate queries on encrypted data. Since a simple solution of encrypting only one value in an encryption block is highly inefficient, the authors propose a solution for manipulating multiple data values in large encryption blocks. Such manipulation handles complex and realistic scenarios such as predicates in queries, compression of data, overflows, and more complex numeric data types (float), etc. In Section 5 we show that the overhead of the operations used in [18] is very large, exceeding the overhead of FMP joins by three orders of magnitude.

The problem of searching on encrypted data has also been studied extensively. Song et al. [41] introduced an elegant solution that uses only simple cryptographic primitives. Chang and Mitzenmacher [11] proposed a solution where the server stores an obfuscated keyword index which is then used by the client to perform the actual searches. Golle et al. [22] provide a solution with the additional feature of allowing conjunctive keyword searches. In a similar context Boneh et al. [9] proposed the notion of "public key encryption with keyword search". They devised two solutions, one using bilinear maps and one using trapdoor permutations. While ensuring keyword secrecy, these techniques do not prevent servers from building searched keyword statistics and inferring sensitive information.

Goh [20] proposed the notion of "secure index" – a data structure associated with a file. The secure index is stored on a remote server and allows clients to privately query an item into the file. The operation can be performed only if the clients have knowledge of a particular trapdoor value. The construction of a secure index uses pseudo-random functions and Bloom filters. This solution requires knowledge of the trapdoor associated with the searched item. Thus, secure indexes are insufficient to provide private joins on outsourced data.

7 Conclusions

In this paper we introduced mechanisms for executing JOIN operations on outsourced relational data with full computational privacy and low overheads. The solution is not hard-coded for specific JOIN predicates (e.g., equijoin) but rather works for a large set of predicates satisfying certain properties. We evaluated its main overhead components experimentally and showed that we can perform more over 5 million private FMPs per second, which is between two and four orders of magnitude faster than alternatives that would use asymmetric encryption algorithms with homomorphic properties to achieve privacy.

Acknowledgments

We would like to thank our reviewers for their extended comments and suggestions which were extremely helpful in preparing the final version of this paper.

References

1. Biometrix Int., http://www.biometrix.at/
2. International HapMap Project, http://www.hapmap.org/
3. TWIRL and RSA Key Size, http://www.rsasecurity.com/rsalabs/node.asp?id=2004
4. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: Proceedings of the International Conference on Very Large Databases VLDB, pp. 143–154 (2002)
5. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the ACM SIGMOD, pp. 439–450 (2000)
6. Bertino, E., Braun, M., Castano, S., Ferrari, E., Mesiti, M.: Author-X: A Java-Based System for XML Data Protection. In: IFIP DBSec, pp. 15–26 (2000)
7. Bertino, E., Jajodia, S., Samarati, P.: A flexible authorization mechanism for relational data management systems. ACM Transactions on Information Systems 17(2) (1999)
8. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
9. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
10. Broder, A., Mitzenmacher, M., Mitzenmacher, A.B.I.M.: Network applications of bloom filters: A survey. In: Internet Mathematics, pp. 636–646 (2002)
11. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: ACNS (2005)
12. Clifton, C., Kantarcioglu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A., Suciu, D.: Privacy-preserving data integration and sharing. In: The 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pp. 19–26. ACM Press, New York (2004)
13. Clifton, C., Marks, D.: Security and privacy implications of data mining. In: Workshop on Data Mining and Knowledge Discovery Computer Sciences, Montreal, Canada, pp. 15–19. University of British Columbia (1996)
14. Devanbu, P.T., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic third-party data publication. In: IFIP Workshop on Database Security, pp. 101–112 (2000)
15. Fagin, R.: Fuzzy queries in multimedia database systems. In: Proceedings of the 17th PODS, pp. 1–10 (1998)

16. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples. In: Proceedings of IEEE Workshop on Generative-Model Based Vision (2004)
17. Gartner, Inc., Server Storage and RAID Worldwide. Technical report, Gartner Group/Dataquest (1999), http://www.gartner.com
18. Ge, T., Zdonik, S.B.: Answering aggregation queries in a secure system model. In: Very Large Databases (VLDB), pp. 519–530 (2007)
19. Gevers, T., Smeulders, A.W.M.: PicToSeek: Combining Color and Shape Invariant Features for Image Retrieval. IEEE Trans. on Image Processing 9(1), 102–119 (2000)
20. Goh, E.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003), http://eprint.iacr.org/2003/216/
21. Goldreich, O.: Foundations of Cryptography I. Cambridge University Press, Cambridge (2001)
22. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
23. Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 216–227. ACM Press, New York (2002)
24. Hacigumus, H., Iyer, B.R., Mehrotra, S.: Providing database as a service. In: IEEE International Conference on Data Engineering (ICDE) (2002)
25. Hale, J., Threet, J., Shenoi, S.: A framework for high assurance security of distributed objects (1997)
26. Hildebrandt, E., Saake, G.: User Authentication in Multidatabase Systems. In: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, Vienna, Austria, (August 26-28, 1998)
27. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of VLDB (2004)
28. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorizations. IEEE S&P, 31–42 (1997)
29. Jajodia, S., Samarati, P., Subrahmanian, V.S., Bertino, E.: A unified framework for enforcing multiple access control policies. In: SIGMOD (1997)
30. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWitt, D.J.: Limiting disclosure in hippocratic databases. In: Proceedings of VLDB, pp. 108–119 (2004)
31. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology 14(4), 255–293 (2001)
32. Li, Feigenbaum, Grosof: A logic-based knowledge representation for authorization with delegation. In: PCSFW: Proceedings of the 12th CSFW (1999)
33. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. In: ISOC Symposium on Network and Distributed Systems Security NDSS (2004)
34. Mykletun, E., Narasimha, M., Tsudik, G.: Signature Bouquets: Immutability for Aggregated/Condensed Signatures. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 160–176. Springer, Heidelberg (2004)
35. Narasimha, M., Tsudik, G.: DSAC: integrity for outsourced databases with signature aggregation and chaining. Technical report (2005)

36. Nyanchama, M., Osborn, S.L.: Access rights administration in role-based security systems. In: Proceedings of the IFIP DBSec, pp. 37–56 (1994)
37. Osborn, S.L.: Database security integration using role-based access control. In: Proceedings of the IFIP DBSec, pp. 245–258 (2000)
38. 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)
39. Rasikan, D., Son, S.H., Mukkamala, R.: Supporting security requirements in multilevel real-time databases (1995), http://citeseer.nj.nec.com/david95supporting.html
40. Sandhu, R.S.: On five definitions of data integrity. In: Proceedings of the IFIP DBSec, pp. 257–267 (1993)
41. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. Proceedings of the IEEE S&P (2000)
42. Russell, B., Freeman, W.T.: LabelMe: the open annotation tool., http://labelme.csail.mit.edu/

Computationally Efficient Searchable Symmetric Encryption

Peter van Liesdonk[1], Saeed Sedghi[2], Jeroen Doumen[2],
Pieter Hartel[2], and Willem Jonker[2]

[1] Technical University of Eindhoven
[2] University of Twente

Abstract. Searchable encryption is a technique that allows a client to store documents on a server in encrypted form. Stored documents can be retrieved selectively while revealing as little information as possible to the server. In the symmetric searchable encryption domain, the storage and the retrieval are performed by the same client. Most conventional searchable encryption schemes suffer from two disadvantages. First, searching the stored documents takes time linear in the size of the database, and/or uses heavy arithmetic operations. Secondly, the existing schemes do not consider adaptive attackers; a search-query will reveal information even about documents stored in the future. If they do consider this, it is at a significant cost to the performance of updates. In this paper we propose a novel symmetric searchable encryption scheme that offers searching at constant time in the number of unique keywords stored on the server. We present two variants of the basic scheme which differ in the efficiency of search and storage. We show how each scheme could be used in a personal health record system.

1 Introduction

Searchable encryption is a technique that provides functionalities to retrieve encrypted documents from a (honest but curious) server selectively while revealing as little information as possible to the server. In case the retrieving and storing encrypted documents is performed by the same client, the searchable encryption is in the symmetric setting. Searchable encryption has many applications, particularly where client privacy is a main concern such as in E-mail servers [4] (public setting), keeping medical information of a client [18] (public and symmetric setting), storing private videos and photos, and backup applications [17] (symmetric setting).

Our work is motivated by the development of personal health care systems. Nowadays, keeping medical records is shifting from paper-based systems to digital record systems. Personal health record (PHR) systems which are initiated and maintained by an individual, are examples of digital record systems. An example of a PHR system is Google Health which offers a client the ability to store her medical records on Googles servers, and allows a general practitioner (GP) to get access to the medical records of her patients. Unlike paper-based systems,

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 87–100, 2010.
© Springer-Verlag Berlin Heidelberg 2010

where the privacy is mainly protected by chaos (since it is almost impossible to locate an individuals record from a multitude of providers) the PHR server might get information about the medical record of the individual after storing or retrieving the record. One way to protect the privacy of the clients is to use a searchable encryption scheme such that i) the medical records are stored in encrypted form, ii) the key used to encrypt the record is kept secret from the server, and iii) the record can be retrieved efficiently and securely.

We call this privacy enhanced PHR, which uses searchable encryption, PHR$^+$. Typical usage scenarios are i) a GP who uses it to retrieve the record of each patient before a visit and who updates the record afterwards, ii) a traveler who uses PHR$^+$ to get access to her medical record anywhere she prefers. In these examples, the reason that PHR$^+$ is used instead of PHR is that using PHR$^+$ the client can store the medical to any honest but curious server (e.g. Google server). Hence, trusting the server is not needed and the client can store the medical records more freely.

We will focus on a setting where a single symmetric key is used for encryption and searching. In more complicated usage scenarios this key can be distributed by means of existing key distribution or access control schemes.

Problem. Existing searchable encryption schemes offer a search algorithm which takes time linear in the number of the documents stored. There are some schemes which allow for a more efficient search, but updating the database is inefficient. Therefore, the problem is to have a searchable encryption scheme that allow efficient search and update.

Contribution. In this paper we propose a novel searchable symmetric encryption scheme that offers efficient searching of the documents stored on the server. Our scheme supports searching time logarithmic in the number of the unique keywords stored on the server, and the client can alter the content of the documents stored while the server learns as little as possible about the alteration. We propose two variants of the scheme proposed which differ in the efficiency of the search and the update operation.

In comparison to [10,6], our scheme deals with adaptive security while having a much lower computational complexity. In comparison with [9], our scheme achieve approximately the same. However, where that scheme the update very impractical it is possible in our scheme; we do not give a security proof however.

The rest of the paper is organized as follows: Section 2 describes the related work in this field. In section 3 we describe the problem with the conventional searchable encryption schemes. In section 4 we describe the background and the security definitions. Our approach and the two variants of the approach are presented in section 5. The appropriate applications of the schemes is described in section 6 and the conclusion is followed in section 7.

2 Related Work

In theory, the classical work of Goldreich and Ostrovsky [12] on oblivious RAMs can resolve the problem of doing private searches on remote encrypted data.

Their scheme is asymptotically efficient and nearly optimal, but does not appear to be efficient in practice as very large constants are hidden in a big-O notation.

Of related interest are private queries on remote public data, or Private Information Retrieval (PIR). This can be achieved efficiently and with perfect security [8] or with computational security [7] when two or more non-colluding servers are used. A computationally secure solution for only a single server is proposed by [14], though it is heavy in both communication and computation.

In [17], Song et al. the question for efficient *keyword* searches was raised. In that paper they propose a scheme that separately encrypts every word of a document independently. This approach has a number of disadvantages. First, it is incompatible with existing file encryption methods. Second, it cannot deal with compressed or binary data. Finally, as the authors themselves acknowledge, their scheme is not secure against statistical analysis across encrypted data. It also lacks a theoretically sound proof.

Goh [10] introduced the formal IND-CKA (Indistinguishability against chosen keyword attacks) and IND2-CKA adversary models. He gives a new approach based on Bloom filters, which hides the amount of keywords used. Chang and Mitzenmacher [6] introduce a simulation-based security definition that is intended to be stronger than IND2-CKA

The first result for an asymmetric setting (multi-user) is *Public-key Encryption with keyword Search* (PEKS) based on identity-based encryption [4]. It uses a adversary model similar to Goh's, but require the use of computationally intensive pairings. This work was extended by [2] to use multiple keywords and to remove the need for secure channels. They also raised the issue of so-called *adaptive* adversaries, where storage occurs after search queries, without giving a solution. Abdalla et. al. [1] perform a more formal analysis of the relation between anonymous IBE and PEKS and discuss the consistency of such schemes.

Curtmola et. al. [9] use a tree-based approach of searchable encryption that takes care of adaptive adversaries. Their scheme is efficient, applicable in both symmetric and asymmetric settings. To prove the scheme secure against a stronger security definition *Adaptive indistinguishability security for SSE.* Unfortunately this tree-based approach also makes updating the index very expensive, making it only suitable for one-time construction of the database.

Of independent interest is work by Bellare et al. [3] and some related papers, which uses deterministic symmetric encryption to achieve a very efficient scheme, with a very weak security model. Finally, [5] give a symmetric scheme using Bloom filters and PIR, that provably leaks no information. However, huge communication and computational costs, makes it only of theoretical interest.

In this paper we propose a novel scheme which supports keyword-based searchable encryption. In contrast with the Curtmola et al. scheme, our scheme enables the client to update the storage efficiently each time required. We give a security proof for the case that the database is built once and searches are made afterwards. While the scheme also covers the case of adaptive updating of the database, this gives rise to very complicated security models. There are a lot of attacks (mostly statistical) that are following not so much from our scheme, but from the scenario and this also applies to other existing schemes.

3 Description of the Problem

Assume that a client wishes to store n documents on a server where each document $D_i = (M_i, W_i), i = 1, ..., n$ is a tuple consisting of a data item M_i and an associated metadata item W_i. The metadata item $W_i = \{w_1, w_2 ...\}$ is actually a set of keywords appended to M_i. The objectives of the client for searchable encrypted storage on the server are as follows:

1. The documents are stored on the server in such a way that the confidentiality of the data items $(M_i), i = 1, .., n$ and the associated metadata items $(W_i), i = 1, .., n$ is preserved.
2. The client queries for a keyword w in order to retrieve all data items M_i where $w \in W_i$ in a secure and efficient way. Here, the security means that the server learns no information about the content of the metadata items when a search is performed except the metadata items retrieved with the query.

According to the client objectives, conventional searchable symmetric encryption schemes for each document $D = (M, W)$ proceed in four phases:

`Keygen`(s)**:** Given a security parameter s, output a private key $K = (k_M, k_W)$ where $k_M \in \{0,1\}^s$ and $k_W \in \{0,1\}^s$.

`Document-Storage`(D)**:** Given the private key K, the document $D = (M, W)$ is transformed to a suitable format for storage on the server using the following sub-algorithms:

- `Data-Storage`(M, k_M): Given as input the data item M and the private key k_M, output encrypted data $\mathcal{E}_{k_M}(M)$.
- `Metadata-Storage`(W, k_W): Given as input the set of associated keywords W and the private key k_W, transform W to a searchable representation S_W.

`Trapdoor`(w, k_W)**:** Given a keyword w and the private key k_W, output a trapdoor T_w.

`Search`(T_w, S_W)**:** Given the searchable representation S_W and the trapdoor T_w, output 1 if $w \in W$.

Having described the construction of conventional searchable encryption schemes, it is evident that the `search` algorithm requires $O(n)$ time, where n is the total number of the documents stored on the database. The reason is that, given a trapdoor T_w, the server has to invoke the `search` function for all the searchable representations stored on the server to check the output of the `search` function.

Although the SWP scheme [17] informally, and the SSE scheme [9] formally addresses the problem by transforming each unique keyword to a searchable representation rather than each metadata item, updating the database is totally inefficient in these schemes since the client needs to alter the whole database for each update. In the rest of the paper we address this problem by proposing an approach which provides functionalities for an efficient search, while efficiently updating the database is still possible for the client.

4 Background and Definitions

Notation Throughout the paper we use the following notation. We use $x \leftarrow_R S$ to denote that x is uniformly drawn from the set S. For a randomized algorithm $\mathcal{A}$, we use $x \leftarrow \mathcal{A}(.)$ to denote the random variable x representing the output of the algorithm. We use $||$ to denote string concatenation.

Pseudo-random function. A pseudo-random function $f(.)$ which is by definition computationally indistinguishable from a truly random function, transforms each element x of the set $\mathcal{X}$ to an output $y \in \mathcal{Y}$ with a secret key $k_f \in \mathcal{K}$ such that the output is not predictable. We say that a pseudo-random function $f : \mathcal{X} \times \mathcal{K} \longrightarrow \mathcal{Y}$, is a (t, q, ε_f) secure pseudo-random function if for every oracle algorithm A making at most q oracle queries and with running time at most t:

$$|Pr[A^{f(.,k_f)} = 1 | k_f \leftarrow \mathcal{K}] - Pr[A^g = 1 | g \leftarrow \{F : \mathcal{X} \rightarrow \mathcal{Y}\}]| < \varepsilon_f$$

Pseudo-random generator. A pseudo-random generator $G(.)$ outputs strings that are computationally indistinguishable from random strings. A pseudo-random generator $G : \mathcal{X} \rightarrow \mathcal{Y}$ is (t, ε_G) secure if for every algorithm A with running time at most t:

$$|Pr[A(G(x)) = 1 | x \leftarrow \mathcal{X}] - Pr[A(y) = 0 | y \leftarrow \mathcal{Y}]| < \varepsilon_G$$

Pseudo random permutation, (block cipher). We say that $\mathcal{E} : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{X}$ is a pseudo-random permutation if every oracle algorithm A making at most q queries and win running time at most t has advantage:

$$|Pr[A^{\mathcal{E}_{k_\mathcal{E}}, \mathcal{E}^{-1}_{k_\mathcal{E}}} = 1] - Pr[A^{\pi, \pi^{-1}} = 1]| < \varepsilon_\mathcal{E}$$

where π is a random permutation selected uniformly from the set of all bijections on $\mathcal{X}$, and where the probabilities are taken over the choice of $\mathcal{K}$ and π.

4.1 Security Definitions

In this paper we use the security definitions introduced in [9]. Security for searchable encryption is intuitively characterized as the requirement that no information beyond the outcome of a search is leaked. However, aside from [11] and the theoretical result of [5], there are no practical schemes that satisfy this characterization; all current practical schemes leak the user's *search pattern* in addition. We take leakage of the access pattern into account by following the simulation-based security definition from [9]. For this definition we need three auxiliary notions: the *history*, which defines the user's input to the scheme; the server's *view*, or everything he sees during the protocols; and the *trace*, which defines the information we allow to leak.

Note that this definition from [9] only considers adaptive search queries, but not adaptive storage or update queries. This means that it only covers the scenario where an initial database is made and searched, but not the scenario with adaptive additions and updates. This would really complicate the security-model, mostly because the choice that the server has to directly return matching documents. From this fact the server learns a lot of information, which would have to be included in the security model.

An interaction between the client and the server will be determined by a document collection and a set of words that the client wishes to search for (and that we wish to hide from the adversary); an instantiation of such an interaction is called a *history.*

Definition 1 (History). *A* history h_q*,is an interaction between a client and a server over q queries, consisting of a collection of documents* $\mathcal{D}$ *and the keywords* w_i *used for* q *consecutive search queries. The* partial history h_q^t *of a given history* $h_q = \big(\mathcal{D}, w_1, \ldots, w_q\big)$*, is the sequence* $h_q^t = \big(\mathcal{D}, w_1, \ldots, w_t\big)$*, where* $t \leq q$*.*

The server's view consists of all the information it can gather during a protocol run. This includes the encrypted documents, their associated document identifiers (ID), the set of searchable representations S which are stored on the server, and all the trapdoors T_{w_i} used for the search queries.

Definition 2 (View). *Let* $\mathcal{D}$ *be a collection of* n *documents and let* $h_q = \big(\mathcal{D}, w_1, \ldots, w_q\big)$ *be a history over* q *queries. An adversaries* view *of* h_q *under secret key* k *is defined as*

$$V_k(h_q) = \big(ID(M_1), \ldots, ID(M_n), \mathcal{E}_{k_M}(M_1), \ldots, \mathcal{E}_{k_M}(M_n), S, T_{w_1}, \ldots, T_{w_q}\big).$$

The partial view $V_k^t(h_q)$ *of a history* h_q *under secret key* k *is the sequence*

$$V_k^t(h_q) = \big(ID(M_1), \ldots, ID(M_n), \mathcal{E}_{k_M}(M_1), \ldots, \mathcal{E}_{k_M}(M_n), S, T_{w_1}, \ldots, T_{w_t}\big).$$

The trace can be considered as all the information that the server is allowed to learn, i.e. information that we allow to leak. This information includes the IDs and length of the encrypted documents (The number of keyword appear in each metadata item), the documents were returned on each search query and the user's search pattern. A user's search pattern Π_q can be thought of as a symmetric binary matrix where $(\Pi_q)_{i,j} = 1$ iff. $w_i = w_j$. Additionally, we include $|\mathcal{W}_\mathcal{D}|$, the total amount of keywords used in all documents together. See Section 5.5 on how to hide the amount of keywords.

Definition 3 (Trace). *Let* $\mathcal{D}$ *be a collection of* n *documents and let* $h_q = \big(\mathcal{D}, w_1, \ldots, w_q\big)$ *be a history over* q *queries. The* trace *of* h_q *is the sequence*

$$\mathrm{Tr}(h_q) = \Bigg(ID(M_1), \ldots, ID(M_n), |M_1|, \ldots, |M_n|, |\mathcal{W}_\mathcal{D}|, \mathcal{D}(w_1), \ldots, \mathcal{D}(w_n), \underset{q}{\Pi}\Bigg).$$

Now we are ready for the security definition for semantic security, where we use a simulation-based approach, like [9,13]. In this definition we assume that the client initially stores an amount of documents and afterwards does an arbitrary amount of search queries. Intuitively, it says that given all the information the server is allowed to learn (Trace), he learns nothing from the information he receives (View) about the user's input (History) that he could not have generated on his own. Note that this security definition does not take updates into account.

Definition 4 (Adaptive Semantic Security for SSE). *A SSE scheme is* adaptively semantically secure *if for all* $q \in \mathbb{N}$ *and for all (non-uniform) probabilistic polynomial-time adversaries* $\mathcal{A}$*, then there exists a (non-uniform) probabilistic polynomial-time algorithm (the simulator)* $\mathcal{S}$ *such that for all traces* Tr_q *of length* q*, and for all polynomially sampleable distributions* $\langle_q = \{h_q : \mathrm{Tr}(h_q) = \mathrm{Tr}_q\}$ *(i.e. the set of histories with trace* Tr_q*), all functions* $f : \{0,1\}^m \rightarrow \{0,1\}^{l(m)}$ *(where* $m = |h_q|$ *and* $l(m) = poly(m)$*, all* $0 \leq t \leq q$ *and all polynomials* p *and sufficiently large* κ*:*

$$\left|\Pr\left[\mathcal{A}\left(V_k^t(h_q)\right) = f(h_q^t)\right] - \Pr\left[\mathcal{S}\left(\mathrm{Tr}(h_q^t)\right) = f(h_q^t)\right]\right| < \frac{1}{p(s)}$$

where $h_q \xleftarrow{R} \langle_q$*,* $k \leftarrow keygen(s)$*, and the probabilities are taken over* $\langle_q$ *and the internal coins of keygen,* $\mathcal{A}$*,* $\mathcal{S}$ *and the underlying Storage algorithm.*

5 Efficiently Searchable Encryption Schemes

Consider a set of documents $\mathcal{D} = \{D_1, ..., D_n\}$, a set of metadata items $\mathcal{W} = \{W_1, ..., W_n\}$, and a set of data items $\mathcal{M} = \{M_1, ..., M_n\}$, where each document $D_i = (M_i, W_i)$ consists of a data item M_i and a metadata item (a set of associated keywords) W_i. Let ID_i be the document identifier (ID) associated to document D_i (and W_i and M_i as well). Let $I_w = \{ID_i | w \in W_i\}$ be a collection of IDs showing in which metadata items a keyword w occurs.

In this section, we first present our approach which supports computationally efficient searching on symmetrically encrypted documents. The main idea of our approach is transforming each unique keyword w to a searchable representation S_W, in a way that the client can keep track of the metadata items in which w occurs $\{W_i | w \in W_i\}$ via a trapdoor T_w. Our approach also allows the client to update the searchable representation of w efficiently each time needed. The construction of the searchable representation for each unique keyword w is:

$$S_W = (f_{k_f}(w), m(I_w), R(w)).$$

Here f_{k_f} is a pseudo-random function that identifies the searchable representation of w, k_f is the private key k_W or part of k_W, $m(.)$ is a masking function and $R(w)$ is a function that keeps some information for unmasking I_w. Each time the client wants to retrieve the encrypted data items whose metadata items contain a keyword w, a trapdoor $T_w = (f_{k_f}(w), R'(w))$ is sent to the server. Given the trapdoor T_w, the server first searches the searchable representations for $f_{k_f}(w)$. In case $f_{k_f}(w)$ occurs, the associated masked $m(I_w)$ is unmasked using $R(w)$ and $R'(w)$. The server then sends the encrypted data items whose IDs occur in I_w to the client. This idea allows faster search compared to many other searchable encryption schemes since the time taken for the search is logarithmic in the number of unique keywords stored on the server. Note that since our scheme reveals the search pattern, the document identifiers that occur in the set I_w are hidden until a search occurs.

Depending on how masking is performed we present two schemes which are the two variants of our approach. The first scheme is more efficient than the second one in terms of computation for the search, but the search and the storage should be performed interactively between the client and the server. The second scheme performs searching and storing the database non-interactively.

5.1 Scheme 1: Interactive Search and Storage

Let I_w be the set of IDs showing in which metadata items keyword w occurs and S_w be the searchable representation of keyword w, which have been already stored on the server. In this scheme I_w is represented as an array of bits where each bit is zero unless its position equals to at least one of the keywords that occur in the metadata items. In this scheme the construction of the searchable representation is:

$$S_w = (f_{k_f}(w), I_w \oplus G(r), \mathcal{E}_{k_\mathcal{E}}(r))$$

Our basic scheme comprises of the following algorithms:

`Keygen(`s`)` : Given a security parameter s, outputs a master key $K = (k_W, k_M)$.

`Document-Storage(`$\mathcal{D}, K$`)` : Given a set of documents $\mathcal{D}$, and the private key K, transform $\mathcal{D}$ to an appropriate format for storage on the server using the following sub algorithms:

- `Data-Storage`$(\mathcal{M}, k_M)$: Given the data items $\mathcal{M}$ and the private key k_M, transform each data item $M_i \in \mathcal{M}$ to encrypted form $< \mathcal{E}_{k_M}(M_i), ID_i >_{i=1,..,n}$.
- `Metadata-Storage`$(\mathcal{W}, k_W)$: Let $k_W = (k_f, k_\mathcal{E})$, where $k_f \in \{0,1\}^s, k_\mathcal{E} \in \{0,1\}^s$. Given a set of metadata items $\mathcal{W} = \{W_1, ..., W_m\}$, and the key for each unique keyword $w \in \mathcal{W}$, this algorithm has to updates the list of the indexes of searchable representation S_w (in case S_w is not stored on the server, we assume that $I_w = 0$ such that the updated searchable representation S'_w contain indexes of the metadata items in which w occur. This algorithm is performed interactively between the client and the server.
  1. Client: Build $U_w = \{ID_i | w \in W_i, W_i \in \mathcal{W}\}$ which is the indexes of the metadata items in which w occurs. Send $f_{k_f}(w)$ to the server.
  2. Server: Search the first components of the searchable representations for $f_{k_f}(w)$. Send the $\mathcal{E}_{k_\mathcal{E}}(r)$ associated with $f_{k_f}(w)$ to the client.
  3. Client: Given $\mathcal{E}_{k_\mathcal{E}}(r)$, decrypt $r = \mathcal{E}^{-1}_{k_\mathcal{E}}(\mathcal{E}_{k_\mathcal{E}}(r))$. Pick a new random value r', and send the server $C = (f_{k_f}(w), U_w \oplus G(r) \oplus G(r'), \mathcal{E}_{k_\mathcal{E}}(r'))$.

4. Server: Let $C = (C_1, C_2, C_3)$. Let $S_w = (S_1, S_2, S_3)$. The updated searchable representation is $S'_w = (f_{k_f}(w), U_w \oplus I_w \oplus G(r'), \mathcal{E}_{k_\mathcal{E}}(r'))$. Here the updated list of index for w, $I'_w = I_w \oplus U_w$, contains the IDs showing where w occurs after the storage is performed.

`Trapdoor`(w, k_W): Given a keyword w and the private key $k_W = (k_f, k_\mathcal{E})$, output a trapdoor $T_w = f_{k_f}(w)$.

`Search`(T_w, S): This algorithm is performed interactively between the server and the client.

1. Server: Given a trapdoor T_w, search the first component of the searchable representations for T_w. Let $S_w = (S_1, S_2, S_3)$. Send S_3 to the client.
2. Client: Send back $\mathcal{E}^{-1}_{k_\mathcal{E}}(S_3)$ to the server, c) compute $S_2 \oplus G(\mathcal{E}^{-1}_{k_\mathcal{E}}(S_3))$ to obtain I_w, d) send the encrypted data items whose IDs occur in I_w to the client.

5.2 Security for Scheme 1

Theorem 1. *The scheme described in Section 5.1 is secure in the sense of Adaptive Semantic Security for SSE in definition 4.*

Proof. Let $q \in \mathbb{N}$, and let $\mathcal{A}$ be a probabilistic polynomial-time adversary. We will show the existence of a probabilistic polynomial-time algorithm $\mathcal{S}$ (Simulator) as in definition 4. Let

$$\mathrm{Tr}_q = \left(\mathrm{id}(M_1), \ldots, \mathrm{id}(M_n), |M_1|, \ldots, |M_n|, |W_\mathcal{D}|, \mathcal{D}(w_1), \ldots, \mathcal{D}(w_q), \underset{q}{\Pi}\right) \quad (1)$$

be the trace of an execution after q search queries and let H_q be a history consisting of q search queries such that $\mathrm{Tr}(H_q) = \mathrm{Tr}_q$. Algorithm $\mathcal{S}$ works as follows:

$\mathcal{S}$ chooses n random values $R_1, \ldots, R_n$ such that $|R_i| = |M_i|$ for all $i = 1, \ldots, n$. He constructs a simulated index $\bar{S}$ by making a table consisting of entries (A_i, B_i, C_i) with random A_i, B_i and C_i, for $i = 1, \ldots, |W_\mathcal{D}|$. Next, $\mathcal{S}$ simulates the trapdoor for query t, $(1 \leq t \leq q)$ in sequence. If $(\Pi_q)_{jt} = 1$ for some $j < t$ set $T_t = T_j$. Otherwise choose a j in $1 \leq j \leq |W_\mathcal{D}|$ such that for all i, $1 \leq i < t$, $A_j \neq T_i$ set $T_j = A_j$. $\mathcal{S}$ then constructs for all t a simulated view

$$\bar{V}^t_K(h_q) = \big(\mathrm{id}(D_1), \ldots, \mathrm{id}(D_n), R_1, \ldots, R_n, \bar{S}, T_1, \ldots, T_t\big), \quad (2)$$

and eventually outputs $\mathcal{A}(\bar{V}^t_k)$.

We now claim that $\bar{V}^t_k$ is indistinguishable from $V^t_k(h_q)$ and thus that the output of $\mathcal{A}$ on $V^t_k(h_q)$ is indistinguishable from the output of $\mathcal{S}$ on input $\mathrm{Tr}(h_q)$. Therefore we first state that: the $\mathrm{id}(M_i)$ in $V^t_k(h_q)$ and $\bar{V}^t_k(h_q)$ are identical, thus indistinguishable; $\mathcal{E}_k$ is a pseudorandom permutation, thus $\mathcal{E}_k(M_i)$ and R_i are distinguishable with negligible probability; f_{k_f} is a pseudorandom function, thus $t_i = f_{k_f}(w_i)$ and T_i are distinguishable with negligible probability. Also the relations between the elements are correct by construction.

What is left is to show that $\bar{S}$ is indistinguishable from S, i.e. that the tuples (A_i, B_i, C_i) are indistinguishable from tuples $(f_{k_f}(w_i), I_{w_i} \oplus G(r_i), \mathcal{E}_{k_\mathcal{E}}(r_i))$. First note again that $f_{k_f}(w_i)$ is indistinguishable from the random A_i since f_{k_f} is a pseudorandom function. Given I_{w_i} and the fact that G is a pseudorandom generator there exists an s_i such that $I_{w_i} \oplus G(s_i) = B_i$. Given that $\mathcal{E}$ is an IND-CPA permutation C_i is indistinguishable from $\mathcal{E}(s_i)$.

Since $\bar{V}_k^t$ is indistinguishable from $V_k^t(h_q)$, the output of $\mathcal{A}$ will also be indistinguishable. This completes the proof.

5.3 Scheme 2: Non-interactive Search and Storage

Although scheme 1 is efficient in terms of computation for searching searchable representations, there are two disadvantages with the scheme: i) the `Metadata-Storage` and the `search` algorithms are performed interactively with the server, ii) the `Metadata-Storage` algorithm requires a large bandwidth since the size of U_w should be equal to the size of I_w (which is large for the large scale databases).

Here we present scheme 2 which efficiently addresses the disadvantages described above with the cost of more computation for the search. The key idea to perform the storage and the search non-interactively is to deploy a hash chain. A hash chain of length N ,

$$H^N(a) = \underbrace{H(H(...H(a)...))}_{N}$$

is constructed by applying repeatedly a hash function $H(.)$ to an initial seed value a [15]. Since only the client knows the seed, she is able to traverse the chain forward and backward, while the server is able to traverse the chain forward only.

In contrast with scheme 1, where the IDs showing in which metadata items a keyword w occurs are stored in one array of bits I_w, in scheme 2 the list of the IDs are stored in the masked form individually each time the `metadata-storage` algorithm is invoked. This approach diminishes the bandwidth required for storing the metadata items.

In this scheme the masking function $m(.)$ is actually the pseudorandom permutation function function $\mathcal{E}$. Let $I_i(w)$ be the set of IDs showing in which metadata item w occurs, which have been added to S_W in the ith updating S_W occurs. The searchable representation of w after updating i times is:

$$S_{old}(w) = (f_{k_f}(w), \mathcal{E}_{k_1(w)}(I_1(w)), H'(k_1(w)), ..., \mathcal{E}_{k_i(w)}(I_i(w)), H'(k_i(w))),$$

where $k_j(w) = H^{N-ctr}(w||k)$. Here ctr is a counter which is incremented each time the `Metadata-Storage` function is invoked (for any keyword). The counter ctr is stored ion the client side. This construction encrypts each list $I_j(w)$ with a unique secret key $k_j(w)$ such that the server is able to compute the encryption key of the previously added lists $k_{j-1}(w), ..., k_1(w)$ given the encryption key of the latest storage $k_j(w)$, by traversing the hash chain forward. Since the server cannot traverse the chain backward, it is not possible for the server to compute $k_{j+1}(w)$ given $k_j(w)$.

Scheme 2 comprises of the following algorithms:

`Keygen`(s)**:** Given a security parameter s, output a master key $K = (k_W, k_M)$.

`Document-Storage`$(\mathcal{D}, K)$: Given a set of documents $\mathcal{D}$, and the private key K, transform $\mathcal{D}$ to an appropriate format for storage on the server using the following sub algorithms:

- `Data-Storage`$(\mathcal{M}, k_M)$: Given the data items $\mathcal{M}$ and the private key k_M, transform each data item $M_i \in \mathcal{M}$ to encrypted form $< E_{k_M}(M_i), ID_i >_{i=1,..,n}$.
- `Metadata-Storage`$(\mathcal{W}, k_W, ctr)$: Let $k_W = (k_f, k)$, where $k_f, k \in \{0,1\}^s$. Let N be the length of the hash-chain. Assume that $S_{old}(w)$ has been already i times updated. For each unique keyword $w \in \mathcal{W}$ construct $I_{i+1}(w) = \{ID_j | w \in W_j, W_j \in \mathcal{W}\}$ and then perform the following; i) increment the counter $ctr = ctr + 1$, ii) compute the secret key $k_{i+1}(k) = H^{N-ctr}(w||k)$, iii) encrypt $\mathcal{E}_{k_{i+1}(w)}(I_{i+1}(w))$, iv) sends the tuple

$$(f_{k_f}(w), \mathcal{E}_{k_{i+1}(w)}(I_{i+1}(w)), H'(k_{i+1}(w)))$$

  to the server, who adds the received tuple to $S_{old}(w)$. Figure illustrates the `metadata-storage` algorithm in scheme 2.

`Trapdoor`(w, k_W, ctr)**:** Given a keyword w and the private key k_W, output a trapdoor $T_w = (f_{k_f}(w), H^{N-ctr}(w||k))$.

`Search`(S, T_w)**:** Let $T_w = (T_1, T_2)$. Assume that S_W has been updated for i times. Given the trapdoor T_w and the searchable representations S, search S for T_1. If T_1 occurs, compute $H'(T_2)$, if $H'(T_2) = H'(k_i(w))$, decrypt I_i using T_2, otherwise keep computing $T_2 = H(T_2)$ until $H'(T_2) = H'(k_i(w))$. After $k_i(w)$ is computed, repeats the same procedure to compute the previously added list of document identifiers. Having decrypted $I_i(w), ..., I_1(w)$ send the documents whose IDs occur in the lists.

Optimization. 1. Each time the server decrypts each list $I_j(w)$ after a search, the list is kept in plaintext, such that for later searches, the server has to decrypt only the list of the document identifiers that have been added to S_W since the last search. This modification will decrease the computation for the Search algorithm.

Optimization. 2. Schemes 2 suffers from a limitation that the maximum number of times the storage can be updated is limited. The limitation comes from the finite length of the pseudo-random chain used in the scheme. In other words, after the counter ctr reaches the value of N, where N is the length of the chain, the chain cannot be used. At this point the pseudo-random chain is said to be exhausted and the whole process should be repeated again with a different seed to re-initialize the chain. One way to decrease the exhaustion rate is that the counter ctr is only incremented in case a search has occurred since the latest update. The reason is that without performing the search, the server does not know anything about the key $k_i(w)$ used in the last update. Hence, the exact $k_i(w)$ used for the last time that update occurred can be used for the current update.

5.4 Security for Scheme 1

Theorem 2. *The scheme described in Section 5.3 is one way secure for SSE in definition 4.*

In scheme 2, the trace and the simulation view will be the same as Eq.1 and Eq.2 respectively. Without loss of generality let$\mathcal{S}_W$=($f_{k_f}(w), \mathcal{E}_{k_1(w)}(I_1(w)), H'(k_1(w)))$. Then, $f_{k_f}(w)$ is indistinguishable from random. However, since the key k_1 is constructed by the one way hash function $H(.)$, the key is one way secure. Therefore $\mathcal{E}_{k_1(w)}(I_1(w))$ will be one way secure.

5.5 Security of Updates

In the security proof in Section 5.4 we did not consider the security of updates. In fact there is information leakage in this case, specifically the amount of keywords in each update and information on which keyword are in common over several updates. For our extended scheme in Section 5.3 we did not discuss security at all. There the security is similar to that of 5.1, but the improvements does not make sense when updates are not considered. However, there are several tricks to minimize this information leakage:

Batched updates. Updating a single document reveals the amount of keywords used for that document. However, our scheme allows us to update many documents at once. In that case the update only reveals information about the aggregated keywords over all updated documents. In this way the information leakage goes asymptotically towards zero bits if the amount of simultaneously updated documents increases.

Fake updates. The `Metadata-Storage` algorithm allows us to update the searchable representation of a keywords without actually changing the indexed documents, similar in idea to the technique in [2] to hide the amount of keywords. This allows the client to always do an update with an identical amount of keywords, or even to update all keywords at once.

6 Application

Having described the schemes we proposed, we revisit the two scenarios from the introduction to show how each exploits the advantages of the schemes. The first scheme is appropriate for the traveller who uses PHR$^+$ to store his medical record such that the record can be retrieved selectively anywhere. As an example, a journalist using PHR$^+$ to check the validation of a vaccination. In this case, since the client (journalist) uses a broadband internet connection, the time delay due to the second round of communication for the search is not a problem. The second scheme is appropriate for instance for a GP who uses PHR$^+$ to store the record of a patient, and who retrieves the record of each patient before or during a visit. The GP also updates the record of the patient afterwards.

7 Conclusion

We propose a novel searchable symmetric encryption scheme which supports the searching time logarithmic in the number of the unique keywords stored on the server while it is updatable. We propose two variants of the approach which differ in the efficiency of the search and the update. We now present a general assessment of the two schemes proposed. The first scheme is after each update more efficient in terms of computation, but requires two rounds of communication between the server and the client for each search. The second scheme enables the client to update the stored documents with a minimum bandwidth and high efficiency. This scheme also makes possible to undo update for the client. However, the update and the search should be interleaved and the maximum number of times the documents stored can be updated is limited. Table 1 summarizes the features of the schemes proposed.

Table 1. Summary of the features of the schemes proposed. In this table u is the number of unique keywords, N is the length of the hash chain, and l is the average number of encrypted document identifiers added to S_W since the last search.

	Variants of the Basic Scheme	
Features	Scheme 1	Scheme 2
Communication overhead	Two rounds	One round
Searching Computation	$log(u)$	$log(u) + \frac{N}{2}l$
Condition on Update	Occurs rarely	Interleaved with search

Acknowledgment

This research is supported by the SEDAN project, funded by the Sentinels program of the Technology Foundation STW under project number EIT.7630.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. J. Cryptology 21(3), 350–391 (2008)

2. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. Cryptology ePrint Archive, Report 2005/191 (2005), `http://eprint.iacr.org/`
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes [16], pp. 535–552
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
5. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public key encryption that allows PIR queries. In: Menezes [16], pp. 50–67
6. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
7. Chor, B., Gilboa, N.: Computationally private information retrieval (extended abstract). In: STOC, pp. 304–313 (1997)
8. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
9. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 79–88. ACM, New York (2006)
10. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003), `http://eprint.iacr.org/`
11. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)
12. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM 43(3), 431–473 (1996)
13. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
14. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
15. Lamport, L.: Password authentification with insecure communication. Commun. ACM 24(11), 770–772 (1981)
16. Menezes, A. (ed.): CRYPTO 2007. LNCS, vol. 4622. Springer, Heidelberg (2007)
17. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
18. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.U.: Privacy preserving error resilient dna searching through oblivious automata. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 519–528. ACM, New York (2007)

Towards the Secure Modelling of OLAP Users' Behaviour

Carlos Blanco[1], Eduardo Fernández-Medina[2], Juan Trujillo[3], and Jan Jurjens[4]

[1] Dep. Of Mathematics, Statistical and Computation. Facultad de Ciencias
University of Cantabria. Av. De los Castros s/n. 39071. Santander. Spain
Carlos.Blanco@unican.es
[2] Dep. of Information Technologies and Systems. Escuela Superior de Informática.
GSyA Research Group. University of Castilla-La Mancha.
Paseo de la Universidad, 4. 13071. Ciudad Real. Spain
Eduardo.Fdezmedina@uclm.es
[3] Dep. of Information Languages and Systems. Facultad de Informática.
LUCENTIA Research Group. University of Alicante. San Vicente s/n. 03690.
Alicante. Spain
jtrujillo@dlsi.ua.es
[4] Germany TU Dortmund & Fraunhofer ISST. Germany
jan.jurjens@cs.tu-dortmund.de

Abstract. Information Security is a crucial aspect for organizations, and must be considered during the development of Information Systems. The data in Data Warehouses (DWs) are highly sensitive since they manage historical information which is used to make strategic decisions, and security constraints should therefore be included in DW modelling within its structural aspects. However, another dynamic security component is also related to the sequences of OLAP (On-Line Analytical Processing) operations, and could be used to access (or infer) unauthorized information. This paper complements the modelling of DWs with state models, which permit the modelling of these dynamic situations in which sensitive information could be inferred. That is, it models queries that include security issues, and controls that their evolution through the application of OLAP operations always leads to authorized states. Finally, our proposal has been applied to a healthcare case study in which a DW manages admissions information with various security constraints.

Keywords: Data Warehouses, OLAP, Users Behaviour, Query Evolution, State Models, Security, Inference, Healthcare.

1 Introduction

DWs organize enterprises' historical information, which originates in heterogeneous data sources, for the decision-making process. It is widely accepted that the information in DWs is organized on the basis of multidimensional modelling in which facts represent the interesting measures of a business process to be analyzed (e.g. "sales", "deliveries", etc.) and related dimensions classify this information by the subjects that

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 101–112, 2010.
© Springer-Verlag Berlin Heidelberg 2010

represent the context in which a fact is analysed (e.g. “product”, “customer”, “time”, etc.) [1, 2].

Information security is a critical aspect which must be considered during the entire Information Systems development process [3-8]. Security requirements can therefore be identified from the early stages of the development process and taken into consideration in design decisions, thus ensuring their perfect fit in a better quality solution.

Since DWs manage highly sensitive information which supports the decision making process and which also usually includes personal data protected by legal regulations, security and privacy are absolutely vital and should be considered in all layers and operations of the DW from the beginning, as strong requirements to their final implementation in DBMS (Data Bases Management Systems) or OLAP (On-Line Analytical Processing) tools [9].

Several proposals for DW modelling through a consideration of their specific structural characteristics (facts, dimensions, bases, hierarchies, etc.) exist, but only some of them include security aspects in their modelling [10-12]. However, these contributions deal with the security problem in a static manner in which a set of security constraints basically establish what information will be shown to or hidden from the user, depending on his/her security profile.

Our previous work has been focused on the static modelling of DWs by defining several models that have been improved with security capabilities. This has been aligned with a model driven architecture, thus permitting modelling at the business (CIM), conceptual (PIM) and logical (PSM) levels, and the automatic generation of final implementation into DBMS or OLAP tools by applying transformation rules. Our complete proposal was applied in a healthcare case study in which the “admissions” to a hospital were analyzed by studying patients and diagnosis, and considering several security constraints defined over sensitive information [11, 13-17].

Nevertheless, DW confidentiality could also be compromised by sequences of OLAP operations (such us drill-down or roll-up) which could be carried out by a malicious user in order to access or to infer unauthorized information, and these security constraints cannot be defined in a static model because they depend on the evolution of the queries.

Since the modelling of all possible queries is not financially viable, our proposal is focused on sensitive queries (sensitive joints of information) and their evolution. In a first stage, sensitive queries are detected and included in the static model by using a new kind of security rule called a “Joint Rule” which specifies the security privileges needed to combine this information.

In a second stage, the sensitive queries are then modelled by using state models. Possible evolutions of the query are represented as states which are reached by applying certain OLAP operations. The designer establishes security constraints to decide which states can be reached and what information should be shown or hidden depending on the user’s security privileges and the previously visited states, thus avoiding the discovery of unauthorized information.

This paper is organized as follows: Section 2 presents our proposal for the modelling of secure OLAP systems by focusing on the secure modelling of OLAP users’ behaviour through the use of state diagrams; Section 3 provides a case study; and finally, Section 4 shows our conclusions and future work.

2 Secure Modelling of OLAP Systems

Figure 1 shows an overview of our proposal to model secure OLAP systems. In a first stage, DW static aspects are modelled by using a UML profile [18]. Sensitive queries and their evolution through the application of OLAP operations are then modelled by using state models which fulfil the static model. And finally, the user's session is also controlled in order to avoid inferences between different queries.

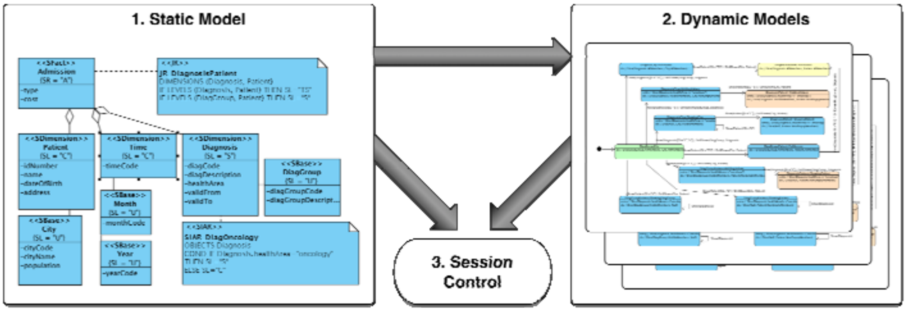


Fig. 1. Security Modelling Proposal for OLAP Systems

2.1 Static Modelling

A UML profile which is specifically created for DWs [18] allows the DW to be modelled at a conceptual abstraction level. On the one hand, structural aspects can be defined (such as fact, dimension and base classes, measures, attributes, hierarchies, etc.) and on the other hand, it is complemented with security constraints by using an Access Control and Audit (ACA) model [15]. This ACA model uses a classification consisting of three points of view: security levels (the users' clearance level), compartments (horizontal classification) and roles (hierarchical structure). It also permits security information (security levels, compartments and roles) to be associated for each element in the model (fact, dimension, etc.), and provides a set of security rules with which to define sensitive information, authorizations and auditing information.

Sensitive information assignment rules (SIAR) and authorization rules (AUR) are responsible for hiding certain information depending on the security profile (security level, compartment and role) of the user who attempts to access it.

- These rules are solved in design time, that is, the elements (facts, dimensions, bases, attributes, etc.) that have to be hidden in each user profile are already known. For instance, the "Diagnosis" dimension will be hidden from users with a security level that is lower than "Secret" (see Figure 3).
- In other situations, these rules are more complex and include conditions which have to be evaluated in execution time in order to hide the instances which do not satisfy them. For example, instances of "Diagnosis" with a health area attribute that is equal to "oncology" will require a security level of "Top Secret" (Figure 3).

Nevertheless, although the security privileges needed to access each element separately (e.g. each dimension) have been defined in the static model, the combination of several information elements is usually more sensitive than when they are accessed separately and an additional specification is required for these more restrictive security privileges (for example, in order to access “Patients” in relation to their “Diagnosis”).

This paper includes a new kind of security rules, denominated as Joint Rules, to establish the security privileges needed to access certain combinations of information. Joint Rules (JR) are associated with a fact class in the model, and must be defined according to the grammar shown in Table 1. This kind of rules is composed of several fields: a set of involved dimensions, which are represented by the keyword “DIMENSIONS”, followed by a list of dimension class names and the specification of the security privileges needed to access different combinations of multidimensional objects. This is defined by using the keyword “IF” followed by a list of multidimensional objects (the dimension or base classes which represent different aggregation levels), the keyword “THEN” and the security information required, which is composed of security levels, roles and compartments. An example of a joint rule called “JR_DiagnosisPatient” is shown in the following section (Figure 3).

Table 1. JR Syntax

JR := INVDIMENSIONS DEFLEVEL+
INVDIMENSIONS := “DIMENSIONS“ SDimensionClassName+
DEFLEVEL := “IF (” OBJECT+ “) THEN” SECINF
OBJECT := SDimensionClassName | SBaseClassName
SECINF := SLEVEL? SROLES? SCOMPARTMENTS?
SLEVEL := “SL=” securityLevel
SROLES := “SR=” userRole+
SCOMPARTMENTS := “SC=” userCompartment+

2.2 Dynamic Modelling

In this section, dynamic models (state models) are proposed in order to enrich the aforementioned static models by including security aspects which are related to queries and their evolution through the application of OLAP operations and cannot, therefore, be modelled in a static manner.

Queries involve the combination of several dimensions at certain granularity levels, and this combination tends to be more sensitive than the separate accessing of data. These sensitive combinations are detected and modelled in the static model by using the new kind of security rules - joint rules - proposed in this paper, but their evolution through the application of OLAP operations should also be considered in order to ensure that confidentiality is not compromised. Thus, once sensitive queries have been specified by using joint rules, there is a dynamic modelling stage in which each sensitive query is modelled by using an extension of state models for secure OLAP systems.

Figure 2 shows an overview of our modelling proposal. States represent evolutions of a query obtained after applying certain OLAP operations that change the aggregation level (drill down and roll up).

Each state defines which pieces of information will be provided to the user. Firstly, when a state is reached, an entry action executes a slice operation with a set of restrictions which hide the unauthorized information. A dice operation then selects the multidimensional elements (measures, dimensions, bases, etc.) and members which should be shown according to the user's privileges.

The actions which can be achieved by users to allow them to navigate towards the hierarchies involved (drill down and roll up operations) establish connections between states and define the security guard conditions (security privileges) needed to reach them. For instance, the example shown in Figure 2 considers a hierarchy "A-B". If a user in "State 1" wants to show "B", that a user's security privileges will be checked ("securityGuard"), and if they are satisfied, a drill down operation will be achieved. "State 2" is then reached and the specified slice and dice operations are carried out.

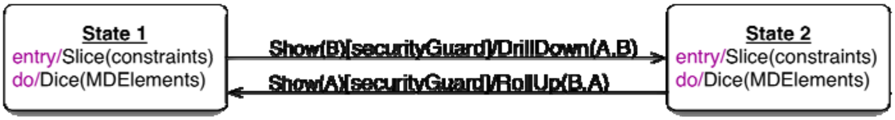


Fig. 2. Secure state model overview

The starting point of the diagram leads to the less sensitive state that defines a query involving the dimensions specified in the joint rule grouped by the less restrictive aggregation level. Since the user can finish his/her session in any time, a specific end point has not been established.

In this approach, the combination of several multidimensional elements in a certain aggregation level signifies that it is necessary to create several states with different visibility privileges. These states are created by considering: (1) the security rules defined in the static model, including the new kind of security rules (joint rules); and (2) the designer's decisions which, according to the previously visited states, establish what information should be shown to guarantee confidentiality.

The designer defines the slices and dices for each state by using multidimensional expressions (MDX). For instance, a condition with which to hide oncology diagnosis included in a SIAR can be expressed in a slice operation as "diagnosis.healthArea<>"oncology"". There are some interesting expressions for the security point of view which allow certain dimension members to be shown: all members,

Table 2. MDX security expressions defined

Expression	Description
x.AllMembers	shows all the instances of a dimension or base "x"
x.VisibleMembers	after applying the specified visibility restrictions (slices), only shows the visible instances of a dimension or base "x"
x.NonEmptyMembers	after applying the specified visibility restrictions (slices), only shows the instances of a dimension or base "x" which contain data since, in some situations, showing empty members could be used to infer information
Null	hides all the instances

visible members, non-empty members or none of them. Some of them can be easily expressed in MDX, such as all members (".allMembers"), but others, such as non-empty members, require more complex expressions. Since the use of these expressions will be very usual in the secure dynamic modelling, we have defined a set of them (see Table 2).

2.3 Session Control

In the previous stages, the security rules solved in design time and in execution time (evaluating conditions) have been defined in the static models, and sensitive combinations of information have also been detected (by using joint rules) and modelled (by using state models) in order to control the evolution of queries by applying OLAP operations.

Since users could achieve a sequence of different queries, users' sessions should also be controlled in order to detect the possibility of an information inference. For example, although the combination of "Patients" and "Diagnosis" was established as being sensitive, an attacker could query "Patients" with "Time" and then "Diagnosis" with "Time", and thus infer unauthorized information by crossing data. This problem could be modelled by using this proposal, i.e., by using additional joint rules and static models, but we believe that it is of greater interest to control inferences by analyzing users' sessions.

Each session is composed of several events which are different queries:

$$Session_n = <e_1, e_2, \ldots e_n>$$

This session control stage checks each event and uses a stack to store the multidimensional elements which have been shown. It then uses the joint rules defined in the static model to discover what the sensitive combinations (sensitive joints) are, and analyses the stack to find the possibility of an inference.

The administrator is informed if the possibility of an inference has been detected and he/she decides what is the best action to take: to introduce noise in the data; to deny certain queries; to reduce the privileges of certain users; etc.

3 Case Study

In previous works, our secure data warehouses modelling proposal was applied in a healthcare case study by using static models in which both the structural and security aspects of data warehouses could be defined [11, 13-17]. The secure state models presented in this paper complement our proposal by dealing with the dynamic security problems that could not be modelled with a static approach. This case study shows how our dynamic proposal is applied to a healthcare system for a DW that analyzes hospital admissions, and manages information concerning diagnosis, patients and time.

Firstly, Figure 3 shows the conceptual model for our case study, defined by using the UML profile presented in [18]. This example includes a secure fact class "Admission" with two measures: "type" and "cost". Admissions are classified by using three dimension classes: “Diagnosis”, “Patient” and “Time”, which are related to several base classes, thus creating three classification hierarchies with different aggregation levels (“Diagnosis-DiagnosisGroup”, “Patient-City” and “Time-Month-Year”).

The security configuration used is a sequence of security levels (Top Secret “TS”, Secret “S”, Confidential “C” and Undefined “U”) and a hierarchy of security roles (a root role Employee “E” which has two sub-roles: Health “H” and Admin “A”).

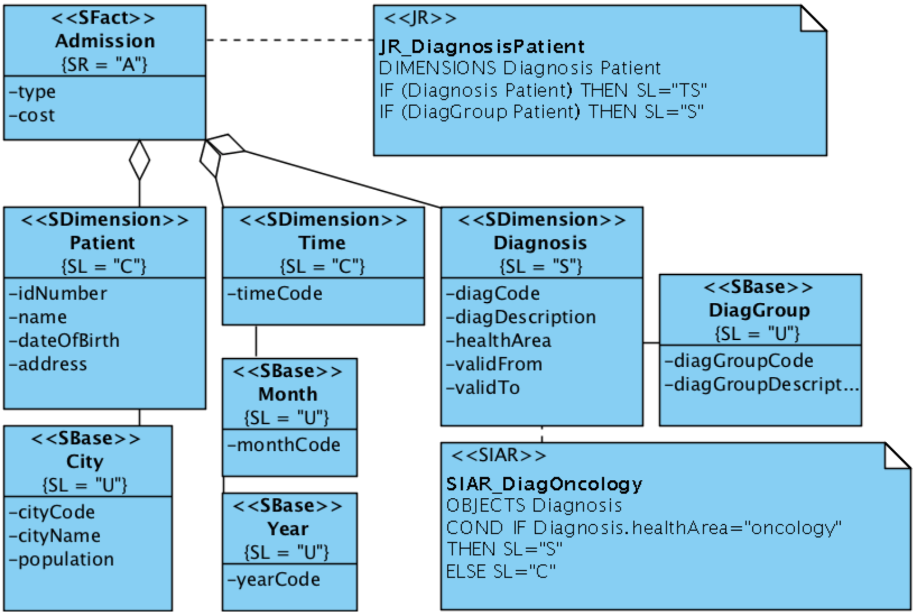


Fig. 3. Conceptual model

Various sensitive information assignment rules (SIAR) have also been defined. Some of these are directly represented by using stereotypes, such as the “Diagnosis” dimension, which uses the stereotype “SL=C” to establish the security level necessary to access the information.

On the other hand, a more complex SIAR called “SIAR_DiagOncology” assigns a higher security level (“Secret”) to “Diagnosis” if the health area is oncology. This has been defined by using an OCL expression in a note associated with the “Diagnosis” dimension class.

An example of the new kind of security rule proposed in this paper, Joint Rule (JR), has been also defined. The rule “JR_DiagnosisPatient” indicates that the querying of the “Diagnosis” and “Patient” dimensions together is more sensitive and requires more restrictive security privileges. Since there are different aggregation levels in the hierarchies involved, the security privileges needed for each sensitive combination are specified separately in the rule. In this example, the combination of “Diagnosis” and “Patient” levels requires a “Top Secret” security level, whereas the combination of “DiagnosisGroup” and “Patient” levels requires “Secret”.

Although security constraints have been considered in the design of the static model, if this model is not complemented with dynamic models, confidentiality could be compromised when users launch a query and apply a sequence of OLAP operations. For instance, a user could query "cost" information relating cities ("City" base class) with diagnosis groups ("DiagGroup" base class) and obtain a sum of costs. He/she could then apply a drill down operation over the diagnosis groups and obtain a different sum of costs because his/her security privileges do not allow all the data to be shown. In this example this user could infer inference by subtracting data.

The secure state diagrams proposed model a specific sensitive query (a sensitive joint of information) and control its evolution by establishing what information will be provided depending on the user's privileges and his/her previous history of actions. In this example, a joint rule called "JR_DiagPatient" indicates that the combinations of "Diagnosis" and "Patient" dimensions are more sensitive and should be modelled with an additional state diagram in order to avoid further information inferences.

Figure 4 shows a state model proposed to control the evolution of queries that combine "Patient" and "Diagnosis" dimensions (or their base classes). Users who wish to query these two dimensions together will always begin by observing the less restrictive state that combines the lower levels of both hierarchies ("DiagGroup" and "City"). They can then query more specific information by applying drill down operations, and they will be able to move to different states depending on the security constraints defined in the static model, the states previously visited and the security decisions made by the designer. In order to decrease the complexity of the model for a better understanding, roll up operations relating states from the most restrictive state ("DiagnosisPatient-FullAccess") towards the less restrictive state ("DiagGroupCity") have not been included.

For example, users who are shown the starting state "DiagGroupCity" might wish to view information concerning diagnosis by achieving a drill down operation over "DiagGroup". In this case, three different states have been defined in order to lead each kind of user towards a secure state in which confidentiality is not compromised:

- Users with a security level of "Top Secret" go to the "DiagnosisCity-FullAccess" state which provides them full access to the queried information.
- Users with a "Secret" security level go to "DiagnosisCity-HidingValues" which applies a slice restricting values from the "oncology" diagnosis (this constraint was defined in the static model as an SIAR) and then execute a dice operation showing all diagnosis members and only the non empty city members.
- Finally, users with a lower security level ("Confidential" or "Undefined") go to a "DiagnosisCity-ShowingCities" state in which the same slice is applied but in this case, the designer has decided to hide the diagnosis and only show visible city members. Although these users can access city information (a security level of "You" is required), if the designer decides to show all the city members, users could infer information from empty cities which might appear because all their patients have an "oncology" diagnosis.

The users are then split according to their security privileges and they are now shown diagnosis related to cities through the application of different restrictions depending on their privileges and the designer's decisions.

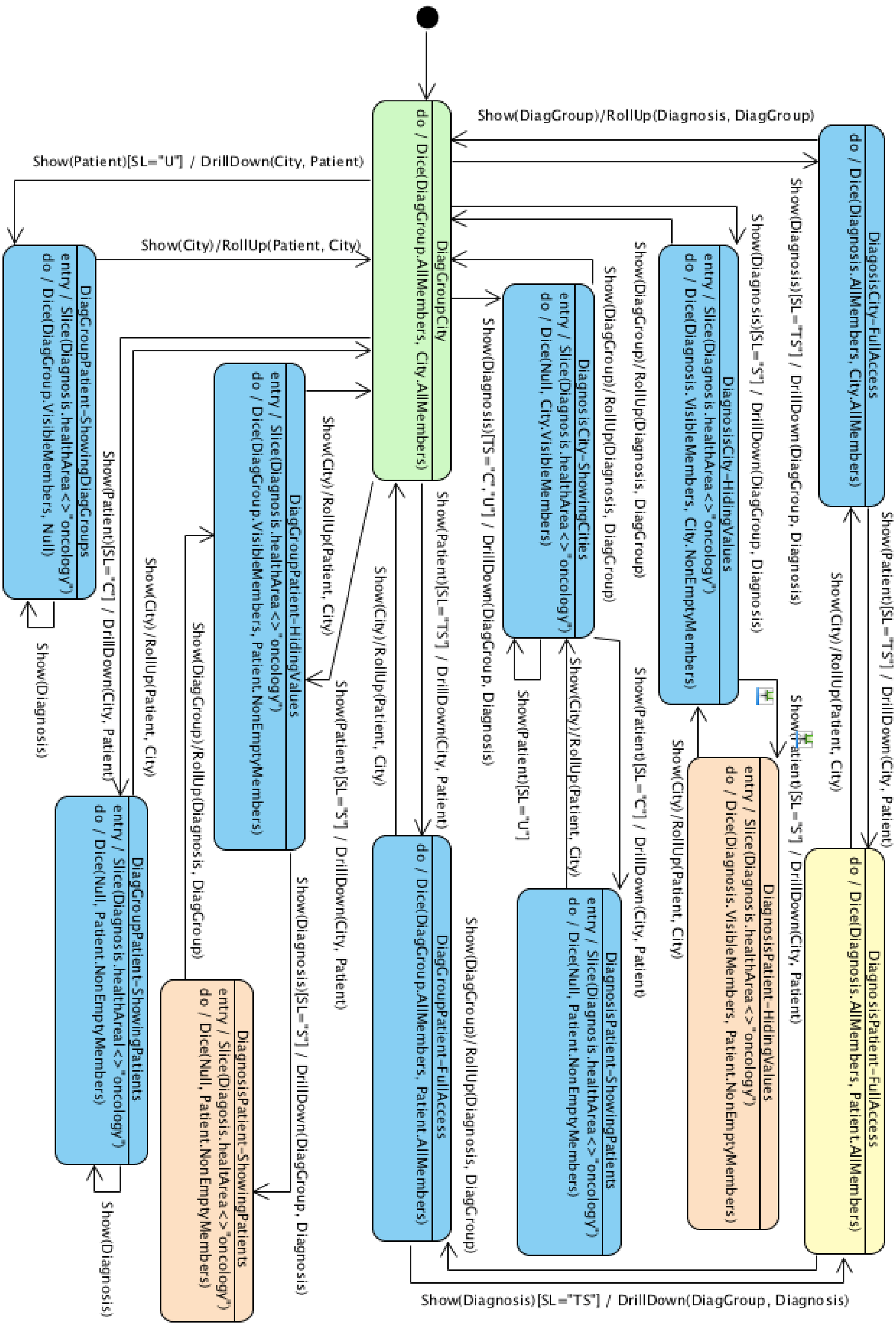

DiagGroupCity
do / Dice(DiagGroup.AllMembers, City.AllMembers)
DiagosisCity-FullAccess
do / Dice(Diagnosis.AllMembers, City.AllMembers)
DiagnosisCity-HidingValues
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(Diagnosis.VisibleMembers, City.NonEmptyMembers)
DiagnosisCity-ShowingCities
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(Null, City.VisibleMembers)
DiagGroupPatient-ShowingDiagGroups
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(DiagGroup.VisibleMembers, Null)
DiagGroupPatient-HidingValues
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(DiagGroup.VisibleMembers, Patient.NonEmptyMembers)
DiagGroupPatient-ShowingPatients
entry / Slice(Diagnosis.healthAreal<>"oncology")
do / Dice(Null, Patient.NonEmptyMembers)
DiagnosisPatient-ShowingPatients
entry / Slice(Diagosis.healthArea<>"oncology")
do / Dice(Null, Patient.NonEmptyMembers)
DiagGroupPatient-FullAccess
do / Dice(DiagGroup.AllMembers, Patient.AllMembers)
DiagnosisPatient-ShowingPatients
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(Null, Patient.NonEmptyMembers)
DiagnosisPatient-HidingValues
entry / Slice(Diagnosis.healthArea<>"oncology")
do / Dice(Diagnosis.VisibleMembers, Patient.NonEmptyMembers)
DiagnosisPatient-FullAccess
do / Dice(Diagnosis.AllMembers, Patient.AllMembers)
Show(DiagGroup)/RollUp(Diagnosis, DiagGroup)
Show(Patient)[SL="U"] / DrillDown(City, Patient)
Show(City)/RollUp(Patient, City)
Show(Diagnosis)[SL="TS"] / DrillDown(DiagGroup, Diagnosis)
Show(Diagnosis)[SL="S"] / DrillDown(DiagGroup, Diagnosis)
Show(DiagGroup)/ RollUp(Diagnosis, DiagGroup)
Show(DiagGroup)/ RollUp(Diagnosis, DiagGroup)
Show(Diagnosis)[TS="C","U"] / DrillDown(DiagGroup, Diagnosis)
Show(City)/RollUp(Patient, City)
Show(Patient)[SL="C"] / DrillDown(City, Patient)
Show(City)/RollUp(Patient, City)
Show(Diagnosis)
Show(DiagGroup)/RollUp(Diagnosis, DiagGroup)
Show(Patient)[SL="S"] / DrillDown(City, Patient)
Show(City)/RollUp(Patient, City)
Show(Patient)[SL="TS"] / DrillDown(City, Patient)
Show(Patient)[SL="U"]
Show(City)/RollUp(Patient, City)
Show(Patient)[SL="C"] / DrillDown(City, Patient)
Show(City)/RollUp(Patient, City)
Show(Patient)[SL="S"] / DrillDown(City, Patient)
Show(City)/RollUp(Patient, City)
Show(Patient)[SL="TS"] / DrillDown(City, Patient)
Show(Diagnosis)[SL="S"] / DrillDown(DiagGroup, Diagnosis)
Show(DiagGroup)/RollUp(Diagnosis, DiagGroup)
Show(Diagnosis)
Show(Diagnosis)[SL="TS"] / DrillDown(DiagGroup, Diagnosis)

Fig. 4. Secure state model for Diagnosis/Patient

The users might then wish to carry out another drill down operation over cities in order to see “Diagnosis” and “Patients” combined. The same procedure will be used to move them to different states with different constraints. For example, users in the “DiagnosisCity-ShowingCities” state will be split into two groups: users with “Undefined” and “Confidential” security levels. This is a designer’s decision in which he/she considers that users with an “Undefined” security level should not be able to access information about patients (in the static model “Patient” requires a security level of “C”) in the same state. However, users with a “Confidential” security level should be able to see this information and can access the “DiagnosisPatient-ShowingPatients” state. As the model shows, a designer has established some visibility constraints in this state in order to avoid inferences. The designer has hidden information about diagnosis and only shows the members of “Patient” which are not empty.

At the beginning of a query evolution, the majority of the generated states correspond with a division of the security privileges according to the security rules defined in the static model, and the designer only makes certain visibility decisions. However, once users have visited several states and thus have more information, the designer’s role takes on more importance. The designer knows each possibility of query evolution and can define different security constraints depending on the states visited.

This example shows how a user with a security level of “Secret” who wishes to query a diagnosis related to patients could reach two different states depending on the states previously visited: “DiagnosisPatient-HiddingValues” and “DiagnosisPatient-ShowingPatients” (these states have been coloured on orange in the diagram). Both states apply a slice that hides the “oncology” diagnosis and only show non-empty “Patients” members, but the designer has decided that the state first mentioned should also show the members of visible diagnosis and the second mentioned state hides this information.

Finally, this state model shows how the joint rule defined in the static model has been specified by creating the “DiagnosisPatient-FullAccess” state, which is the most restrictive and can only be accessed by users with a “TS” security level, and several states which restrict certain information according to user privileges and their history of actions. The other constraint which is present in the joint rule that defines a security level of “S” to combine diagnosis groups and patients has also been considered in the state model. In this case, “TS” users go to a full information access state and “S” users go to the “DiagGroupPatient-HiddingValues” state, which shows this combination by applying certain restrictions (slides). On the other hand, the designer has decided that “C” and “U” users should be have access to different information for this combination, and two different states have therefore been created which only show patients to “C” users and only show diagnosis groups to “U” users.

4 Conclusions

Information security is a crucial requirement which must be taken into account in the development of DWs. Some proposals model DWs at different abstraction levels by including security issues but they use only static models and do not consider query evolution modelling. Although static security measures have been defined, an attacker could achieve a sequence of OLAP operations which would provide him/her with unauthorized information.

In this paper, a new kind of security rules (joint rules) has been added to our static modelling proposal in order to define sensitive combinations of information and the security privileges needed to query this information.

This approach also fulfils static models with a secure dynamic modelling. State models have been proposed in which the use of several states defines what information has to be provided in each case, by considering aspects that it is impossible to model in a static manner, such as the information that has been shown by the user (previously visited states).

Our future work is focused on the improvement of the session control to include an automatic detection of possible inferences and to recommend actions to the administrator.

Acknowledgments. This research is part of the BUSINESS (PET2008-0136) Project financed by the "Ministerio de Ciencia e Innovación" (Spain), the PEGASO/MAGO (TIN2009-13718-C02-01) Project financed by the "Ministerio de Ciencia e Innovación" (Spain) and "Fondo Europeo de Desarrollo Regional FEDER", the SISTEMAS (PII2I09-0150-3135) Project financed by the "Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha" (Spain), the QUASIMODO (PAC08-0157-0668) Project financed by the "Viceconsejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" (Spain), the MEDUSAS (IDI-20090557) Project financed by the "Centro para el Desarrollo Tecnológico Industrial. Ministerio de Ciencia e Innovación (CDTI)" (Spain). Partial support from the EU project "Security Engineering for Lifelong Evolvable Systems (Secure Change)" (ICT-FET-231101) is gratefully acknowledged.

References

1. Inmon, H.: Building the Data Warehouse, 3rd edn. John Wiley & Sons, USA (2002)
2. Kimball, R.: The Data Warehouse Toolkit. John Wiley & Sons, Chichester (2002)
3. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-based modeling language for model-driven security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, p. 426. Springer, Heidelberg (2002)
4. Mouratidis, H., Giorgini, P.: Integrating Security and Software Engineering: Advances and Future Vision. Idea Group Publishing, USA (2006)
5. Fernández-Medina, E., et al.: Model-Driven Development for secure information systems. Information and Software Technology 51(5), 809–814 (2009)
6. Jürjens, J.: Secure Systems Development with UML. Springer, Heidelberg (2005)
7. Jurjens, J.: Principles for Secure Systems Design, PhD Thesis, Oxford University (2002)
8. Houmb, S.H., et al.: Cost-Benefit Trade-Off Analysis using BBN for Aspect-Oriented Risk-Driven Development. In: International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 195–204. IEEE Computer Society, Shanghai (2005)
9. Thuraisingham, B., Kantarcioglu, M., Iyer, S.: Extended RBAC-based design and implementation for a secure data warehouse. International Journal of Business Intelligence and Data Mining (IJBIDM) 2(4), 367–382 (2007)
10. Priebe, T., Pernul, G.: A Pragmatic Approach to Conceptual Modeling of OLAP Security. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, p. 311. Springer, Heidelberg (2001)

11. Fernández-Medina, E., Trujillo, J., Piattini, M.: Model Driven Multidimensional Modeling of Secure Data Warehouses. European Journal of Information Systems 16, 374–389 (2007)
12. Saltor, F., et al.: Building Secure Data Warehouse Schemas from Federated Information Systems. In: Bestougeff, H., Dubois, J.E., Thuraisingham, B. (eds.) Heterogeneous Inf. Exchange and Organizational Hubs, pp. 123–134. Kluwer Academic Publisher, Dordrecht (2002)
13. Trujillo, J., et al.: A UML 2.0 Profile to define Security Requirements for DataWarehouses. Computer Standard and Interfaces 31(5), 969–983 (2009)
14. Trujillo, J., et al.: An Engineering Process for Developing Secure Data Warehouses. Information and Software Technology 51(6) (2009)
15. Fernández-Medina, E., et al.: Access Control and Audit Model for the Multidimensional Modeling of Data Warehouses. Decision Support Systems 42, 1270–1289 (2006)
16. Soler, E., et al.: Building a secure star schema in data warehouses by an extension of the relational package from CWM. Computer Standard and Interfaces 30(6), 341–350 (2008)
17. Blanco, C., et al.: Applying QVT in order to implement Secure Data Warehouses in SQL Server Analysis Services. Journal of Research and Practice in Information Technology 41(2), 135–154 (2009)
18. Fernández-Medina, E., et al.: Developing Secure Data Warehouses with a UML extension. Information Systems 32(6), 826–856 (2007)

A Formal P3P Semantics for Composite Services

Assadarat Khurat[1], Dieter Gollmann[1], and Joerg Abendroth[2]

[1] Hamburg University of Technology,
Harburger Schlossstr. 20, 21079 Hamburg, Germany
assadarat@yahoo.com, diego@tu-harburg.de
[2] Nokia Siemens Networks,
St.Martin-str.53, 81669 Munich
joerg.abendroth@nsn.com

Abstract. As online services are moving from the single service to the composite service paradigm, privacy is becoming an important issue due to the amount of user data being collected and stored. The Platform for Privacy Preferences (P3P) was defined to provide privacy protection by enabling services to express their privacy practices, which in turn helps users decide whether to use the services or not. However, P3P was designed for the single service model, bringing some challenges when employing it with composite services. Moreover the P3P language may lead to misinterpretation by P3P user agents due to its flexibility and may have internal semantic inconsistencies due to a lack of clear semantics. Therefore, we enhance P3P to be able to support composite services, propose a formal semantic for P3P to preserve semantic consistency, and also define combining methods to obtain the privacy policies of composite services.

Keywords: formal semantics, P3P, privacy policy, composite service.

1 Introduction

In the beginning of the online services era, most services were single and independent, employing and developing proprietary technology to serve their customers. Nowadays, there is strong competition in the online market to expand the number of customers. This is an incentive for developing new and better services to better serve user demands. It is then a promising approach to build new services by combining existing services which can reduce development cost and time compared to implementing a new service from scratch. Thus, one can say that the online service world now has changed from single to composite services [12].

In online services, privacy is an important issue due to the large amount of user data collected and stored. Users need some mechanism to secure their data. To protect privacy, users ought first to be aware of what the services will do with their data so that they can decide whether to use the services or not. This facility is provided by the Platform for Privacy Preferences (P3P) [1], a policy language for describing data practices of websites. Comparing these practices with the

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 113–131, 2010.

© Springer-Verlag Berlin Heidelberg 2010

users' preferences, expressed, e.g., in the P3P Preference Exchange Language (APPEL) [5], thus helps users to protect their privacy to a certain degree.

When applying P3P policy for composite services, some challenges are raised. For example, P3P was originally designed for the single service model, how can we obtain the P3P policy of a composite service consisting of several services is one challenge. Besides, P3P policy from some service members may be conflict with each other. How can these discrepancies be detected is another challenge. From these challenges, we enhance P3P so that it can handle composite service scenario and we also define constraints to analyse conflicts between the privacy policies of service members. This analysis also helps avoiding a combination of services with incompatible privacy policies, which will fail at run-time. Moreover, a combining scheme is also proposed to obtain privacy practices of the composite service.

The semantics of the P3P language is ambiguous so that the same P3P policy can be interpreted differently by different user agents [2,10]. This potential semantic inconsistencies of a P3P policy elaborated in [2,3,4,9,11] will be inherited to the composite service consisting of services the ambiguous statement belongs to. We, therefore, consider formal semantics for P3P so that semantic consistency can be preserved. Our formal semantics for P3P is based on work from T. Yu et al.[2] which proposed a data centric formal semantics for P3P by employing relational tables. They claim that in some cases it does not depend on purpose whether or not the collected data is used. We, however, argue that purpose is important to notify the reason for data collection. Thus, data-purpose centric is used in our formal semantics P3P.

The organization of this paper is as follows. Section 2 describes the P3P syntax, its possible semantic inconsistencies and P3P in composite service. The enhancement of P3P is proposed in Section 3. A formal semantics of P3P and constraints for inconsistency verification are shown in Section 4. Section 5 indicates a combining algorithm for composite services. An example of this algorithm is presented in Section 6. Related work is discussed in Section 7 and we conclude our paper in Section 8.

2 The Platform for Privacy Preferences

The Platform for Privacy Preferences (P3P) [1] is an XML-based policy language specified by W3C to enable websites to describe their privacy practices in a standard format. Users can compare P3P policy of a website with their preferences before making a decision whether to use the website. This comparison can be done automatically by a P3P user agent installed at the users' side, e.g. by a web browser. The P3P specification defines syntax and semantics of the P3P policies, mechanisms for associating policies with web resources, e.g. by HTTP, and a standard set of data elements in a hierarchy which all P3P user agents should understand.

2.1 P3P Syntax

The main elements of P3P policy are ENTITY, ACCESS, DISPUTES-GROUP and STATEMENT. In a policy, there are an ENTITY element identifying a legal entity, i.e. a service or website issuing this policy, an ACCESS element indicating the ability of individuals to access their data, a DISPUTES-GROUP describing the resolution procedure when disputes about this privacy practices occur, and one or more STATEMENT elements illustrating possible usage of the website regarding data elements. P3P also defines *categories* for its defined standard set of data elements such as `physical` (e.g. name)and `online` (e.g. email address). These base data are structured in hierarchy. An example of P3P statements of Superpages [13] website that is a service providing search function for e.g. restaurants in desired location is illustrated in Fig. 1.

```
Statement1 { purpose: admin, current, develop
             recipient: ours
             retention: indefinitely
             data: #dynamic.clickstream, #dynamic.http,
                   #dynamic.searchtext}
Statement2 { purpose: develop, tailoring
             recipient: ours
             retention: business-practices
             data: #dynamic.cookies [optional] (category: uniqueid)}
```

Fig. 1. P3P Statements of Superpages Website

In a STATEMENT, the major elements are *data*, *purpose*, *recipient* and *retention*. The *data* element expresses data and possibly its category the services may collect. For which purpose and for how long the data is collected can be described by the *purpose* and *retention* elements respectively. P3P defines five *retention* values, `no-retention`, `stated-purpose`, `legal-requirement`, `business-practice` and `indefinitely`. Brief descriptions as defined in P3P 1.1 are given in Table 1. In some cases services distribute data collected to third parties. This can be specified in the *recipient* element. P3P defines six values for this element, `ours`, `same`, `delivery`, `other-recipient`, `unrelated` and `public`. Their short descriptions as defined in P3P 1.1 are given in Table 2.

For *data*, *purpose* and *recipient* elements, there is an attribute describing whether they are required for the website or optional. P3P 1.1 defines an extension element called STATEMENT-GROUP-DEF helping in indicating which statement belongs to the same group of user interactions with the services. Services can also define usages of the same statement group to opt-in or opt-out via the *consent* attribute of this element.

P3P allows multiple *data* elements, multiple *purpose* values, multiple *recipient* values and a *retention* value in one statement. This means the stated *data* elements can be employed for all stated purposes, can be distributed to all listed

Table 1. Descriptions of P3P Retention Values

Retention Values	Descriptions
`no-retention`	Information is not retained for more than a brief period of time necessary to make use of it during the course of a single online interaction.
`stated-purpose`	Information is retained to meet the stated purpose. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`legal-requirement`	Information is retained to meet the stated purpose but the retention period is longer because of a legal requirement or liability. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`business-practices`	Information is retained under a service provider's stated business practices. Sites MUST have a retention policy that establishes a destruction time table. The retention policy MUST be included in or linked from the site's human-readable privacy policy.
`indefinitely`	Information is retained for an indeterminate period of time.

Table 2. Descriptions of P3P Recipient Values

Recipient Values	Descriptions
`ours`	Ourselves and/or entities acting as our agents or entities for whom we are acting as an agent
`same`	Legal entities following our practices
`delivery`	Delivery services possibly following different practices (may use data other than stated purpose including delivery services with unknown data practices)
`other-recipient`	Legal entities following different practices
`unrelated`	Unrelated third parties (data usage practices are unknown)
`public`	Public fora

third party recipients, and are stored as long as defined in the retention policy. Besides, P3P allows specifying the same *data* elements in more than one statement. This flexibility on one hand brings expressiveness to the policy. However, on the other hand it may also cause semantic inconsistency problems.

2.2 Potential Semantic P3P Inconsistencies

There exist possible semantic inconsistencies due to non-corresponding between values of some elements of P3P policy, which were elaborated in [2,3,4,9,11]. We briefly describe them in the following.

Conflict between purpose and data category: For example, a statement contains *purpose* value `individual-analysis` but does not contain any *data* element from one of `physical`, `online`, `financial`, `purchase` and `government` data categories. This does not make sense since the *purpose* `individual-analysis` requires identified data. Similar conflicts are listed in [3] and in the P3P 1.1 specification (User Agent Guidelines).

Conflict between purpose and retention: For instance, a statement contains *purpose* value `develop` and *retention* value `no-retention`. This introduces a conflict since `no-retention` allows data to be stored only for a brief period of time which is less than `develop` requires.

Conflict between purpose and recipient: For example, when a statement contains *purpose* values `admin`, `develop` and/or `tailoring` with *recipient* values only `delivery`, `same` or `public`. This does not make sense since according to the specified purposes, the *recipient* value `ours` must be present.

Conflict between recipient and retention: in a statement, some defined *recipient* values do not correspond to some *retention* values. When, for instance, a statement contains *recipient* value `public` and a *retention* value other than `indefinitely`.

Optional attributes: With attributes expressing whether *data*, *purpose* and *recipient* elements are required or optional, ambiguities may exist, for example when *data* is required while *purpose* and *recipient* are optional. It is unclear whether the data is collected or not at first. In cases when the STATEMENT-GROUP-DEF element is employed, the value of these attributes in the statement belonging to the group must correspond to the value of the *consent* attribute.

With these possible conflicts, it is not trivial to write P3P policies without considerably caution to be able to protect semantic consistency. Thus, there is a need for formal semantics for P3P to mitigate this problem. For each type of conflict above, there could be more cases introducing semantic inconsistencies in P3P policies. However, our work is focusing on formal semantic for P3P for composite service. Finding out all cases of these conflicts is out of scope for our work. We neither consider discrepancies between P3P policy and its natural language.

2.3 P3P Policy in Composite Services Analysis

Composite services come from Service Oriented Architecture (SOA) approach. A composite service consists of multiple services, being single or composite. Each service is seen as a component that is loosely coupled and make use of data from other services in the combination to provide a new service. The service composition is supported by technologies e.g. BPEL [6], WSDL, etc. Recently, a method to combine also web-based services called "mashup" was proposed in [12].

P3P defined *recipient* value to describe which type of third party the collected data may be distributed to. In a composite service, some data are transferred between service members. Thus the services receiving data become third party services of the data sender service. However, noticing from descriptions of *recipient* values, it could be implied that the *recipient* values can be determined when the recipient services (third party services) are known by the data sender service. In a service composition process, which services are selected as members depends on targets and goals of the desired composite service. Thus, we will never know which services will be combined together beforehand. With this dynamic characteristics, there might be unknown services in the combination. Thus, none of the existing *recipient* values can be used to capture these services.

Basically, each of service members has its own P3P policy. With the single service approach, comparing a user's preferences with P3P policies from all single services is done one by one. However, in a combination this is not sufficient since there might be conflicts between the privacy policies of services. The P3P framework does not provide any mechanism regarding this issue. For example, when privacy policy of Service A does not allow sending data needed by Service B in providing a composite service, thus, this composite service cannot function.

The comparison between users' preferences and privacy practices of the websites are normally done at the user agent, i.e. at the user side. Obtaining the privacy policy of the composite service allows this analysis to occur only once resulting in less resource consumption at the end-users than with the former approach. In addition, this privacy policy of the composite service can be used for further service compositions in the future.

3 P3P Enhancements

In order for P3P to support the composite service paradigm, we enhance P3P in three points. We define a new *recipient* value, require a third party service list for *recipient* values, and define a *destruction time* table for retention policies. The last two extensions facilitate in deriving privacy policy of composite service.

3.1 Recipient Value

Since the existing P3P *recipient* values are not sufficient for composite service,we propose a new *recipient* value, "`prospective-composite-service`", to fill this gap. Similar to all *recipient* values besides `ours`, this value can have a *required* attribute. The new *recipient* value serves as a placeholder for future services in a composite service, implying that the data may be distributed to legal entities that are members of the prospective composite service but with the users' consent. With this value service providers can express their *recipient* value for prospective services that are not known before enabling service composition. Hence, the new composite service can be successfully created but the data of the users who are using a service in the composite service are not distributed to other service members without the users' consent.

3.2 Third Party Service List

Existing single service defines *recipient* values of data based on privacy practices of the data's recipients whether they are the same, different, unknown, etc. to the original service itself. Therefore, each single service has its own scope to determine the *recipient* value which is different from the others. In order to specify the *recipient* value of a composite service, it is necessary to find out the scope of *recipient* values of the composite service in which the scope of all service members must be taken into account. However, since each member has different scope, some recipients of a *recipient* value of a particular data of a service may overlap which makes it extremely hard to resolve the *recipient* values of the composite service instead of just combining all *recipient* values of the data in case of non-overlapping. For example, let a composite service consists of service A and service B. For a particular data, service A has a *recipient* value `other-recipient`. The recipients of `other-recipient` of service A may include service B or some recipients that have the same privacy practices as service B or some recipients that have different privacy practices as service B, resulting in various possible or insolvable *recipient* values for the composite service depending on the *recipient* value of service B.

This problem can be solved if we know exactly the recipients in which we call "third party services" belonging to the *recipient* values of each service member. Therefore besides `prospective-composite-service` and `public`, we require that there must be a list of third party services attached to each *recipient* value of a particular data. We expect that this will not be a difficult task for service providers since it is necessary to know the third party services such that they could specify their *recipient* values. The defined lists for `ours`, `same`, `delivery`, `other-recipient` and `unrelated` values are called "ours-list", "same-list", "delivery-list", "other-recipient-list" and "unrelated-list" respectively. The ours-list from single service contains its own service. We also define another list called "service-member-list" containing all member services of the composite service. In addition, in case of services with known privacy practices their P3P policies must be available.

Since the *recipient* values of existing P3P policies, i.e. without the third party service list, are defined coarse-grained and with single service environments, the P3P policies of a service provider will not change often. On the other hand, P3P policies with a third party service list which could be altered dynamically may have to be updated more frequently. This might be inconvenient especially for the users who normally have to approve the new list every time it changes. However, knowing the list helps the users to protect their data disclosure to unwanted third party services both when they first check whether to use this service or not and later when the list changes. In addition, if the service provider allows the users to specify a third party service black list e.g. at subscription time, when the list changes the service providers can obtain the users' decision automatically whether they agree with the new list or not without contacting the users again.

3.3 Destruction Time Table

It is also not trivial to determine the *retention* value for a composite service. For example, though two services specify their *retention* value as `stated-purpose`, it does not mean they store the data for the same period of time. It is even harder to combine *retention* values between `stated-purpose`, `legal-requirement` and `business-practices` since they are incomparable. A possible way to solve this problem is to know the exact length of the retention time. P3P policy is specified that for *retention* values; `stated-purpose`, `legal-requirement` and `business-practices`, there must be a retention policy establishing a destruction time table included in or linked from the site's human-readable privacy policy but it does not further define what the retention policy looks like. To us, it is foreseeable that this destruction time table can be employed to resolve the *retention* value for composite service. We thus propose that the P3P policy from each service must have a destruction time table as shown in Table 3 for the retention values `stated-purpose`, `legal-requirement` and `business-practices`.

The destruction time table expresses how long a service stores a particular data using the dayTimeDuration type. As stated in the P3P specification for `stated-purpose` and `legal-requirement`, the determination of the duration time depends on the *purpose* element. For `business-practices`, P3P states that the retention policy can be "determined by service provider's business practices" [1]. We suggest to base the retention duration of `business-practices` also

Table 3. Destruction Time Table

Retention Value	Purpose	Retention Duration (dayTimeDuration)
`stated-purpose`	`current`	PT10M
	`admin`	P3M
	`develop`	P3M
	`tailoring`	P3M
`legal-requirement`	`current`	P6M
	`admin`	P6M
	`contact`	P2Y
	`historical`	P2Y
	`telemarketing`	P2Y
`business-practices`	`current`	P3M
	`admin`	P5M
	`tailoring`	P3M
	`develop`	P6M
	`pseudo-decision`	P1Y

on purpose. Thus, for each *retention* value in the table, the retention duration is determined for each applied *purpose* value. Not all *purpose* values may be present since the service may not use the data for that purpose.

4 Formal Semantics for P3P in Composite Services

In our formal semantics for P3P in composite service environments, we assume that the conflicts illustrated in Section 2.2 have already been analysed. Thus, the P3P policies obtained from each service have no conflicts on those issues. In this section, we show how to provide formal semantics of P3P and describe some constraints to further verify inconsistencies of privacy policies in each service and also between services.

4.1 Data-Purpose Centric Semantics for P3P

To provide formal semantics for P3P, we use the same technique as in [2] i.e. employing a relation database. However, instead of following a data-centric approach, we will use data-purpose (dp) as the key for our relation table. The reason is that *purpose* is an important criterion for data usage. For example, to determine whether a data element is required depends pretty much on the purpose the data is being employed for.

For each data element, P3P defines an attribute "optional" to specify whether the data is required by the services. The value of this attribute can be either `yes` (optional) or `no` (required). The default value is `no`. For this relation we define a relation table data-purpose-optional (DPO). In addition, each data element can belong to several categories. We define another relation table for this relation as data-purpose-category (DPC).

Similar to the *optional* attribute of data element, for each *purpose* value except `current` P3P defines an attribute "required" describing whether it is required. There are three values of this attribute, i.e. `always` (required), `opt-out` (users must request to not use the data for this purpose) and `opt-in` (data may only be used for this purpose if the users have agreed). The default value when no *required* attribute is present is `always`. We define a relation table for this relation as data-purpose-required (DPR).

How long a particular data element should be stored as expressed by the *retention* element depends on the purpose the data is being utilized for. According to our P3P enhancement in Section 3.3, each *retention* value for a *purpose* value has a duration time. We define a relation table of this relation as data-purpose-retention-duration (DPRD). The duration represents the retention duration time derived from the destruction time table as defined in Section 3.3.

To whom a particular data may be distributed depends also on the purpose of usage. As same as *purpose* element, each *recipient* value except `ours` can have a *required* attribute, i.e. `always`, `opt-out` or `opt-in`. The default value when no *required* attribute is present is `always`. Extending with the third party service lists, we define a relation table for this relation as data-purpose-recipient-third_party_service-required (DPRTR).

Table 4. Details of Relation Tables

Relation Tables	Fields	Key of the relation
DPO	data purpose optional	(data,purpose)
DPR	data purpose required	(data,purpose)
DPC	data purpose category	(data,purpose,category)
DPRD	data purpose retention duration	(data,purpose)
DPRTR	data purpose recipient third_party_service required	(data,purpose,recipient, third_party_service)
service-entity	ENTITY (`#business.name`) ACCESS DISPUTES (resolution-type) DISPUTES (REMEDIES)	(ENTITY)

We also rewrite the ENTITY, ACCESS and DISPUTES-GROUPS elements in a table called service-entity. The key element of this table is ENTITY, `#business.name` of the service. The details of these tables are shown in Table 4.

4.2 Constraints for Integrity Verification

We use similar constraints as shown in [2] and an additional constraint according to the data-purpose centric approach to ensure the integrity of P3P policies for each service as follows:

Multiple statements for a data element: P3P allows specifying more than one statement for the same data-purpose. Thus, multiple *retention* values can be assigned to the same data-purpose. For two tuples having the same data-purpose values, it does not make sense to have several *retention* values. If this is the case, the policy is considered invalid. This can be applied also for the DPO, DPR and DPRTR relations.

$$\forall p, q \in DPRD, p.(data, purpose) = q.(data, purpose)$$
$$\rightarrow p.retention = q.retention$$

Data hierarchy: in P3P, data is structured in a hierarchy. It does not make sense if the upper level (e.g. #user.bdate) has more restrictions than its lower level (e.g. #user.bdate.ymd.year). For instance, if the data optional attribute of #user.bdate is optional, the data optional attribute of #user.bdate.ymd.year can be optional or required. But if the data optional attribute of #user.bdate is required, the data optional attribute of #user.bdate.ymd.year must be required. This concept can be applied to DPR and DPRTR relations.

$$\forall p, q \in DPO, p.data \text{ is the upper level of } q.data, p.optional = \text{no} \\ \rightarrow q.optional = \text{no}$$

Purpose and data optional constraints: if the *purpose* value of a data is `current`, the *optional* value of that data must be `no` since it is necessary that the data must be collected. Otherwise the service cannot be provided.

$$\forall p \in DPO, p.purpose = \text{current} \rightarrow p.optional = \text{no}$$

4.3 Constraints for Integrity Verification between Services

In the composite service environment, several single services and/or composite services cooperate with each other to perform a service. Hence, input data of one service can be output of other services. This may introduce conflicts when a service supposed to send data as an input to another service does not do it due to its privacy policies. Thus, we first have to find out which personal data is transferred between services. Then we further analyse its privacy policies for both sender and recipient services.

The personal data transferred can be derived e.g. from a WS-BPEL [6] description of a service. How the personal data are derived is out of scope of our work. In the following, the data structure *shared* with components *shared.data* and *shared.provider* will hold the personal data shared between services and their providers. For personal data items with more than one provider there would be several entries. We define constraints to check the conflicts for each service member of a composite service as follows:

Data collection: the data used between services are obviously needed to fulfill the service provisioning, thus their collection must be required. Otherwise, this composite service cannot function.

$$\forall p \in DPO, p.data \in shared.data \rightarrow p.optional = \text{no}$$

Purpose: for the same reason as for data collection, the purpose values of these data must contain at least `current` value with *required* value as `always`.

$$\forall p \in DPR, p.data \in shared.data \rightarrow p.purpose = \text{current} \wedge p.required = \text{always}$$

Recipient: if the *recipient* value of the transferred data of the sender service contains at least `prospective-composite-service`, the combination is feasible.

Let $DPRTR_i$ be the DPRTR table for provider (i). We check the following constraints on that data:

$$\forall p \in DPRTR_i, (p.data, i) \in shared$$
$$\rightarrow p.recipient = \text{prospective-composite-service}$$

If the *recipient* value of the transferred data contains only `ours`, this means that the service does not send that data to other parties. Thus, the composite service cannot function. In other cases, the third party service list must be checked. If the recipient service is not in the list as specified, this service combination cannot function.

$$\forall p \in DPRTR_i, (p.data, i) \in shared \rightarrow p.recipient \neq \text{ours}$$

5 Combining Mechanism

We determine relation tables for the composite service (CS) by combining the same table type of each member service. For each type of table we add all tuples from all services that have different data-purpose values into a new table i.e. belonging to the composite service. Let dp be (data, purpose) attributes in relations. For tuples having the same dp values, the following combination schemes are applied before adding them into the new table:

a) data-purpose-optional (DPO)

It does not make sense for a service to specify several values for the data optional attribute of a data-purpose value. As we assume that the P3P policy from each service is validated beforehand, the *optional* value for a dp from each service must be only one value, either `yes` (optional) or `no` (required). For the same reason, the *optional* value of a dp for a composite service must be only one value.

With n tuples from DPO (dp_1,optional_1), (dp_2,optional_2),$\ldots$, (dp_n,optional_n) having the same data-purpose ($\text{dp}_1 = \text{dp}_2 = \cdots = \text{dp}_n$), we define that the *optional* value of the composite service is the most restrictive one (`no` >`yes`). This means if at least one of the *optional* values is `no` then the *optional* value of the composite service is `no`, otherwise `yes`.

From the same reason shown in Section 4.2, data hierarchy is also considered. With two data elements, if *optional* value of the upper level data has higher restriction than the lower one, the *optional* value of the lower one must be changed to be the same as the upper level data.

b) data-purpose-required (DPR)

This *required* value represents the requirement of the *purpose* element. Similarly to data optional, the *required* value of a dp from each service must be only one value i.e. `always`, `opt-out` or `opt-in`. With n tuples from DPR (dp_1,required_1), (dp_2,required_2),$\ldots$, (dp_n,required_n) having the same data-purpose ($\text{dp}_1 = \text{dp}_2 = \cdots = \text{dp}_n$), the *required* value of the composite service is the most restrictive one (`always` >`opt-out` >`opt-in`).

c) data-purpose-category (DPC)

A data-purpose relation can have several *category* values. Since we assume that possible inconsistencies between purpose and data category are validated beforehand, this conflict will not occur when combining P3P policies from different services. With n tuples from DPC (dp_1,$\mathrm{category}_1$), (dp_2,$\mathrm{category}_2$),. . ., (dp_n,$\mathrm{category}_n$) having the same data-purpose ($\mathrm{dp}_1 = \mathrm{dp}_2 = \cdots = \mathrm{dp}_n$), the *category* value of the composite service ($\mathrm{category}_{CS}$) can be obtained as the union of the *category* values from all of them ($\mathrm{category}_{CS} = \mathrm{category}_1 \cup \mathrm{category}_2 \cup \cdots \cup \mathrm{category}_n$).

d) data-purpose-retention-duration (DPRD)

As previously described in Section 3.3, it is not trivial to combine *retention* values for a composite service. The duration value derived from destruction time table can help in determining the *retention* value as below.

Similar to the *optional* and *required* attributes for a data-purpose relation from a service, the *retention* value must be only one value. With n tuples from DPRD (dp_1,$\mathrm{retention}_1$,$\mathrm{duration}_1$), (dp_2,$\mathrm{retention}_2$,$\mathrm{duration}_2$),. . ., (dp_n,$\mathrm{retention}_n$,$\mathrm{duration}_n$) having the same data-purpose ($\mathrm{dp}_1 = \mathrm{dp}_2 = \cdots = \mathrm{dp}_n$), the *retention* value of the composite service is the most restrictive one (`indefinitely` > `business-practices`, `legal-requirement`, `stated- purpose` > `no-retention`). Due to the fact that `business-practices`, `legal-requirement` and `stated-purpose` are incomparable, there can be cases where the *retention* value cannot be resolved by using only this constraint. Thus the duration time comes into play.

Since the *retention* value describes how long the service keeps the data, we consider the longest storing period and its *retention* value to become the retention duration and *retention* value of the composite service respectively. For example, let the purpose for collecting a particular data be `admin` and the *retention* value of the first service be `stated-purpose` with a retention duration of three months, while the second service collects the same data for the same purpose for six months with the *retention* value `business-practices`. The *retention* value and its duration of the composite service for this data-purpose relation is `business-practices` with six months duration due to the longer time duration. In case when the retention duration from both is equal, we consider the one with the most restrictive meaning (`business-practices` >`legal-requirement` >`stated-purpose`).

e) data-purpose-recipient-third_party_service-required (DPRTR)

The *recipient* is quite a complicated element when considered in a composite service. Though it is possible for a data-purpose relation to have several *recipient* values, determining the *recipient* value for the composite service is not possible by just simply taking the union of the *recipient* values from all services having the same data-purpose relation as previously described in Section 3.2. Therefore, we use the third party service list in determining the *recipient* value of the composite service.

Besides the *recipient* value of the composite service and the *required* value of each *recipient* value, we also have to find out the member services of each list. Unlike the first four relations that the tuples having different data-purpose values are added in the new table belonging to composite service without further processing. Due to the different scope of single service and composite service, all tuples of DPRTR relation after being added in the new table are analysed. The *recipient* value, its *required* value and member services of each third party service list of the composite service can be derived in the following steps:

1. For each *recipient* value (except `prospective-composite-service` and `public`), there is a list of third party services attached as well as a *required* value. We attach this *required* value to every third party service in the list (tps-required). For the tuples of DPRTR table from all services having different data-purpose (dp) values, first they are added into the new table waiting for further processing as shown in the following steps. With n tuples from DPRTR (dp_1,$\mathrm{recipient}_1$,tps_1,$\mathrm{required}_1$), (dp_2,$\mathrm{recipient}_2$,tps_2,$\mathrm{required}_2$), . . . , (dp_n,$\mathrm{recipient}_n$,tps_n,$\mathrm{required}_n$) having the same data-purpose ($\mathrm{dp}_1 = \mathrm{dp}_2 = \cdots = \mathrm{dp}_n$), we combine their *recipient* value by using union before adding them in the new table. For each *recipient* value, we also take the union to combine the third party services associated with. Each third party service can have only one *required* value. With n tuples from tps-required (tps_1,$\mathrm{required}_1$), (tps_2,$\mathrm{required}_2$),. . ., (tps_n,$\mathrm{required}_n$) having the same service ($\mathrm{tps}_1 = \mathrm{tps}_2 = \cdots = \mathrm{tps}_n$), the *required* value is the most restrictive one (`always` >`opt-out` >`opt-in`). In this step, if the *recipient* values contain `public`, then `public` is one of the *recipient* values for the composite service respectively. The *required* value of *recipient* value `public` is the most restrictive one (`always` >`opt-out` >`opt-in`).
2. Since the scope of *recipient* values from each member service of the composite service may be different, there might be some third party services in same-list, delivery-list, other-recipient-list and unrelated-list belonging to the member of the composite service. This overlapping must be eliminated. We thus check services in each list with the service-member-list whether they are redundant. If there are, those services must be removed from their current lists. This also implies that the *recipient* value `ours` is one of the *recipient* value for the composite service.
3. In addition, there might be some third party services in delivery-list, other-recipient-list and unrelated-list that have the same privacy practices as the composite service. We thus check this case in two steps. First we check if there are any services in delivery-list, other-recipient-list and unrelated-list that are the same as the services in the same-list. If there are, those services must be removed from their current lists. Secondly we also have to check whether the remaining services in those lists have the same data practices as the composite service or not by verifying their privacy policies against the privacy policies of the composite service which will be explained later. If any of these services conform to the composite service's privacy policies, they are moved to the same-list. In case

there are no privacy policies available (i.e. all services in unrelated-list and some services in delivery-list) or the services do not conform to the composite service's privacy practices, they are left in their current lists. We do not have to check redundancy of services between delivery-list, other-recipient-list and unrelated-list since from the definitions they do not overlap with each other.

4. For the ours-list, same-list, delivery-list, other-recipient-list and unrelated-list, if they are not empty, then their corresponding *recipient* values, i.e. `ours`, `same`, `delivery`, `other-recipient` and `unrelated` respectively are one of the *recipient* values for the composite service and they become also the third party service lists of the composite service. Here there could be several *required* values attached to services in each list coming from different member services, the new *required* value of the *recipient* values for the composite service is the most restrictive one (`always` >`opt-out` >`opt-in`).
5. For the *recipient* value `prospective-composite-service`, since now the scope is different than from the single service, we remove this *recipient* value associated with data that are not the output of the composite service. These data can be derived from e.g. WS-BPEL.

Regarding the checking of the same privacy practices in step 3, by now we already obtained the first four relation tables of the composite service i.e. DPO, DPR, DPC and DPRD. The verification between the privacy policies of the remaining services in delivery-list, other-recipient-list and unrelated-list and the privacy practices of the composite service is done by checking with these four relation tables and with restricted *recipient* values. For a data-purpose relation if the *optional* value, *required* value, *category* value and *retention* value of the remaining services in those lists are equal or less restrictive than the *optional*, *required*, *category* and *retention* values of the composite service respectively and if the *recipient* values of the services in those lists contain only `ours` or `same` or both, this implies that their privacy policies conform to the composite service. We do not take the *required* value of the *recipient* value `same` into consideration, since even if it is required, the recipient services in the same-list still have the same privacy practices as the composite service.

For the service-entity table, we can obtain the table of the composite service by just concatenating every row from each service into one table.

6 Example

In this section, we present a scenario to obtain privacy policy of a potential composite service, Restaurantlocator. The Restaurantlocator service may consist of Superpages and GPS-data-stream [14] services. Given a city name in United States, a list of restaurants can be obtained from the Superpages. After choosing the restaurant users want to visit, the GPS-data-stream provides a point of interest file containing exact location of the restaurant which can be directly loaded to users' GPS device. P3P statement of GPS-data-stream is shown in Fig.2.

```
Statement { purpose: admin, current, develop
            recipient: ours
            retention: indefinitely
            data: #dynamic.clickstream, #dynamic.http,
                  #dynamic.cookies (category: navigation, online)}
```

Fig. 2. P3P Statement of GPS-data-team

Due to the limitation of space, we give example in P3P statement instead of whole policy and with only one data, `#dynamic.clickstream`. In order to obtain privacy policy of the Restaurantlocator, existing P3P policies of both Superpages and GPS-data-stream must be extended beforehand. The privacy practices of Superpages and GPS-data-stream services for data, `#dynamic.clickstream`, are mapped to relation tables DPO, DPR, DPC, DPRD and DPRTR with their combining result belonging to composite service, Restaurantlocator, as illustrated in Table 5,6,7,8,9 respectively.

From all relation tables, an overlapping of data and *purpose* values between Superpages and GPS-data-stream services is data `#dynamic.cookies` with *purpose* value `develop`. For the DPO relation in Table 5 and DPR relation in Table 6, the *optional* and *required* values for the Restaurantlocator of the data are `no` and `always` correspondingly according to the rules that the most restrictive value is chosen as proposed in Section 5. For the DPC relation in Table 7, the *category* values for Restaurantlocator of the overlapping data can be derived by applying union function. Deploying the retention duration values from the Destruction time table in Table 3 for Superpages service, the duration of *purpose* values, `develop` and `tailoring`, are six months and three months. Since the *retention* value of GPS-data-stream for the *purpose* value, `develop`, is `indefinitely`, the *retention* value of composite service shown in Table 8 for that data becomes `indefinitely`. In the Table 9 expressing DPRTR relation, since all the *recipient* values from both service members are `ours`, further checking on whether the third party services have the same or different privacy practices is not needed and the *required* values of all data are `always`. For the `#dynamic.clickstream` data with *purpose* value , `develop`, of Restaurantlocator, both service members are in its ours-list.

Table 5. Data-Purpose-Optional (DPO) Relation

	Data	Purpose	Optional
Superpages	`#dynamic.cookies`	`develop`	`yes`
	`#dynamic.cookies`	`tailoring`	`yes`
GPS-data-stream	`#dynamic.cookies`	`admin`	`no`
	`#dynamic.cookies`	`current`	`no`
	`#dynamic.cookies`	`develop`	`no`
Restaurantlocator	`#dynamic.cookies`	`admin`	`no`
	`#dynamic.cookies`	`current`	`no`
	`#dynamic.cookies`	`develop`	`no`
	`#dynamic.cookies`	`tailoring`	`yes`

Table 6. Data-Purpose-Required (DPR) Relation

	Data	Purpose	Required
Superpages	#dynamic.cookies	develop	always
	#dynamic.cookies	tailoring	always
GPS-data-stream	#dynamic.cookies	admin	always
	#dynamic.cookies	current	always
	#dynamic.cookies	develop	always
Restaurantlocator	#dynamic.cookies	admin	always
	#dynamic.cookies	current	always
	#dynamic.cookies	develop	always
	#dynamic.cookies	tailoring	always

Table 7. Data-Purpose-Category (DPC) Relation

	Data	Purpose	Category
Superpages	#dynamic.cookies	develop	uniqueid
	#dynamic.cookies	tailoring	uniqueid
GPS-data-stream	#dynamic.cookies	admin	navigation
	#dynamic.cookies	admin	online
	#dynamic.cookies	current	navigation
	#dynamic.cookies	current	online
	#dynamic.cookies	develop	navigation
	#dynamic.cookies	develop	online
Restaurantlocator	#dynamic.cookies	admin	navigation
	#dynamic.cookies	admin	online
	#dynamic.cookies	current	navigation
	#dynamic.cookies	current	online
	#dynamic.cookies	develop	navigation
	#dynamic.cookies	develop	online
	#dynamic.cookies	develop	uniqueid
	#dynamic.cookies	tailoring	uniqueid

Table 8. Data-Purpose-Retention-Duration (DPRD) Relation

	Data	Purpose	Retention	Duration
Superpages	#dynamic.cookies	develop	business-practices	P6M
	#dynamic.cookies	tailoring	business-practices	P3M
GPS-data-stream	#dynamic.cookies	admin	indefinitely	
	#dynamic.cookies	current	indefinitely	
	#dynamic.cookies	develop	indefinitely	
Restaurantlocator	#dynamic.cookies	admin	indefinitely	
	#dynamic.cookies	current	indefinitely	
	#dynamic.cookies	develop	indefinitely	
	#dynamic.cookies	tailoring	business-practices	P3M

Table 9. Data-Purpose-Recipient-Third_party_service-Required (DPRTR) Relation

	Data	Purpose	Recipient	Third_party_service	Required
Superpages	#dynamic.cookies	develop	ours	Superpages	always
	#dynamic.cookies	tailoring	ours	Superpages	always
GPS-data-stream	#dynamic.cookies	admin	ours	GPS-data-stream	always
	#dynamic.cookies	current	ours	GPS-data-stream	always
	#dynamic.cookies	develop	ours	GPS-data-stream	always
Restaurant-locator	#dynamic.cookies	admin	ours	GPS-data-stream	always
	#dynamic.cookies	current	ours	GPS-data-stream	always
	#dynamic.cookies	develop	ours	GPS-data-stream	always
	#dynamic.cookies	develop	ours	Superpages	always
	#dynamic.cookies	tailoring	ours	Superpages	always

7 Related Work

T. Yu et al. proposed a data centric relational semantics database for P3P in [2]. They discussed potential semantic inconsistencies of P3P policies and also defined some integrity constraints to validate the policy, i.e. data-centric constraints, data hierarchy constraints and semantic vocabulary constraints. Our work was inspired by them. However, we use a data-purpose centric approach since it is more appropriate and can better reflect the P3P policies.

M. Schunter et al. suggested augmenting P3P elements, i.e. *retention* by a concrete time-span and *recipient* by a contact address [4,9]. Our work also enhances P3P similarly for *retention* but for *recipient* we extend by introducing a new recipient value and in addition require a third party service list. Our extension comes from the composite service paradigm while their idea still refers to the single service model.

F. Satoh and T. Tokuda proposed in [7] a security policy composition mechanism for composite services from existing policies of service members (called external services in their work). The three types of security policies they focus on are Data Protection Policies (DPP), Access Control Policies (ACP) and Composite Process Policies (CPP). These policies which are derived from BPEL, WSDL and natural language are translated to predicate logic (Prolog) helping in validating policy consistency.

Y. H. Li el al. proposed a graph transformation based approach for verifying consistency between internal business processes using BPEL and an organization's privacy policy using P3P [8]. However, they did not consider the composite service environment.

8 Conclusions

Privacy has become an important issue especially in online services. An existing technology employed to protect privacy is P3P, which enables services to express their privacy practices so that users can decide whether to use these services.

The change of the service architecture from single to composite service paradigm effects P3P policy which was designed for the single service. To mitigate these problems, we enhance P3P policy with a new *recipient* value, a third party service list and a destruction time table; propose a formal semantics for P3P based on data-purpose; define constraints for conflicts verification; and introduce a combining mechanism for the composite service.

References

1. Cranor, L., et al.: The Platform for Privacy Preferences 1.1 (P3P1.1) Specification, W3C Working Group Note (November 2006)
2. Yu, T., Li, N., Antón, A.: A Formal Semantics for P3P. In: ACM Workshop on Secure Web Services (October 2004)
3. Cranor, L.: P3P 1.1 User Agent Guidelines, P3P User Agent Task Force Report 23 (May 2003)
4. Schunter, M., Herreweghen, E.V., Waidner, M.: Expressive Privacy Promises-How to Improve the Platform for Privacy Preferences (P3P). In: Position paper for W3C Workshop on the Future of P3P (September 2002)
5. Cranor, L., Langheinrich, M., Marchiori, M.: A P3P Preference Exchange Language 1.0 (APPEL 1.0), W3C Working Draft (April 2002)
6. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0, OASIS Standard (April 2007)
7. Satoh, F., Tokuda, T.: Security Policy Composition for Composite Services. In: ICWE 2008: Proceedings of the 8th International Conference on Web Engineering, IEEE Computer Society, Los Alamitos (2008)
8. Li, Y.H., Paik, H.Y., Benatallah, B., Benbernou, S.: Formal Consistency Verification between BPEL Process and Privacy Policy. In: PST 2006: Proceedings of International Conference on Privacy, Security and Trust. ACM, New York (2006)
9. Karjoth, G., Schunter, M., Herreweghen, E.V., Waidner, M.: Amending P3P for Clearer Privacy Promises. In: Proceedings of the 14th International Workshop on Database and Expert Systems Applications. IEEE Computer Society, Los Alamitos (September 2003)
10. Antón, A.I., Bertino, E., Li, N., Yu, T.: A Roadmap for Comprehensive Online Privacy Policy Management. Communications of the ACM 50(7), 109–116 (2007)
11. Cranor, L.F., Egelman, S., Sheng, S., McDonald, A.M., Chowdhury, A.: P3P Deployment on Websites. Electronic Commerce Research and Applications Autumn. 7(3), 274–293 (Autumn 2008)
12. Liu, X., Hui, Y., Sun, W., Liang, H.: Towards Service Composition Based on Mashup. In: IEEE Congress on Services, pp. 332–339 (2007)
13. `http://www.superpages.com/`
14. `http://poi.gps-data-team.com/`

A Geometric Approach for Efficient Licenses Validation in DRM

Amit Sachan[1], Sabu Emmanuel[1], and Mohan S. Kankanhalli[2]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
[2] School of Computing, National University of Singapore, Singapore
{amit0009,asemmanuel}@ntu.edu.sg, mohan@comp.nus.edu.sg

Abstract. In DRM systems contents are distributed from the owner to consumers, often through multiple middle level distributors. The owner issues redistribution licenses to its distributors. The distributors using their received redistribution licenses can generate and issue new redistribution licenses to their sub-distributors and new usage licenses to consumers. For the rights violation detection, all the newly generated licenses must be validated. The validation process becomes complex when there exist multiple redistribution licenses for a content with the distributors. In such cases, it requires the validation using an exponential number of validation equations, which makes the validation process much computation-intensive. Thus to do the validation efficiently, in this paper we propose a method to geometrically derive the relationship between different validation equations to identify the redundant validation equations. These redundant validation equations are then removed using graph theory concepts. Experimental results show that the validation time can be significantly reduced using our proposed approach.

1 Introduction

Prolific growth of the Internet technology over the last decade has made the Internet a convenient mode of digital contents distribution. However, it has also increased the fear of illegal contents redistribution and usage. Digital Rights Management(DRM)systems[5][9][6][1][2] emerge as one of the possible solutions to prevent illegal redistribution and usage. DRM systems often involve multiple parties such as owner, multiple distributors and consumers[5][9]. The owner gives the rights for redistribution of contents to distributors by issuing redistribution licenses. The distributors in turn can use their received redistribution license to generate and issue new different types of redistribution licenses to their sub-distributors and new usage licenses to the consumers. A redistribution license allows a distributor to redistribute the content to its sub-distributors and consumers as per the permissions and constraints [11] specified in the redistribution license that it has received. Thus, as part of the rights violation detection, it is necessary to validate these newly generated licenses against the redistribution licenses with the distributors.

The format of both redistribution(L_D) and usage licenses (L_U) for a content K is defined as: (K; P; $I_1, I_2, ..., I_M$; A), where P represents a permission(e.g.

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 132–149, 2010.
© Springer-Verlag Berlin Heidelberg 2010

play, copy, rip, etc.[4][9]), I_i represents the i^{th} ($1 \leq i \leq M$) instance based constraint and A represents aggregate constraint. Instance based constraints in the redistribution licenses are in the form of range of allowed values for distribution, such as region allowed for distribution, period of distribution, etc. Instance based constraints in usage licenses, such as expiry date of license, region allowed for play etc. may be in the form of a single value or range. The range/value of an instance based constraint in further generated licenses using a redistribution license must be within the respective instance based constraint range in it[9]. The aggregate constraint decides the number of permission P *counts* that can be distributed or consumed using a redistribution license or usage license respectively. For aggregate constraints, the sum of the aggregate constraint counts in all the licenses generated using a redistribution license must not exceed the aggregate constraint's value in it. For an issued license to be valid, it must satisfy both instance based and aggregate constraints in the redistribution license which is used to generate it. A validation authority does the validation of newly generated licenses based on instance based and aggregate constraints.

For business flexibility reasons, distributors may need to acquire multiple redistribution licenses for the same content. In case of multiple redistribution licenses, the validation becomes complex[10]. This is because a newly generated license can satisfy all the instance based constraints in more than one redistribution license(say a set of redistribution licenses S). So, for the aggregate constraints validation, the validation authority needs to select a redistribution license from set S. However, selecting a redistribution license randomly for the validation from S may cause potential loss to distributors as we discuss in section 2. To avoid a random selection, a better aggregate validation approach using validation equations is proposed in [10](equations are described in section 2). Validation equations use the whole set of redistribution license S instead of randomly selecting a redistribution license from it.

Doing the validation using the validation equations, the loss to distributors can be reduced but the approach requires the validation using an exponential number (to the number of redistribution licenses present for media)of validation equations. Also each equation may contain up to an exponential number of summation terms. This makes the validation process computationally intensive and necessitates to do the validation efficiently. In [10] an efficient offline aggregate validation method using the *validation tree* was proposed. The *validation tree* uses a prefix tree[7][8] based structure to calculate summation terms in each validation equation efficiently. But, still it requires the validation using exponential number of validation equations. Thus, in this paper we propose a method to geometrically derive the relationship between different validation equations to identify the redundant validation equations. These redundant validation equations are then removed by modification in original validation tree using graph theory concepts. Both theoretical analysis and experimental results show that our proposed method can reduce the validation time significantly.

Rest of this paper is organized as follows. In section 2, we discuss preliminaries required for this work. In section 3, a geometric approach for efficient validation is discussed. In section 4, we modify the original validation tree to use the algorithm proposed in section 3. The performance of our proposed method is analyzed in section 5. Finally, the paper is concluded in section 6.

2 Preliminaries

In this section we first discuss the problem of validation in case of multiple redistribution licenses. Then we give an overview of the *validation tree*[10].

2.1 Validation in Case of Multiple Licenses

In case of multiple received redistribution licenses at the distributors, a newly generated license can satisfy all the instance based constraints in more than one redistribution license(say a set of redistribution licenses S). For the validation purpose, the validation authority needs to select a redistribution license from S. Selecting one redistribution license randomly out of multiple redistribution licenses may cause potential loss to the distributors as illustrated using example 1.

Example 1. Consider five redistribution licenses acquired by a distributor to distribute the play permissions according to two instance based constraints(validity period T, and region allowed R) and aggregate constraint A.

$L_D^1 = (K; Play; I_D^1 : T = [10/03/09, 20/03/09], R = [Asia, Europe]; A_D^1 = 2000)$
$L_D^2 = (K; Play; I_D^2 : T = [15/03/09, 25/03/09], R = [Asia];\ A_D^2 = 1000)$
$L_D^3 = (K; Play; I_D^3 : T = [15/03/09, 30/03/09], R = [America];\ A_D^3 = 3000)$
$L_D^4 = (K; Play; I_D^4 : T = [15/03/09, 15/04/09], R = [Europe];\ A_D^4 = 4000)$
$L_D^5 = (K; Play; I_D^5 : T = [25/03/09, 10/04/09], R = [America];\ A_D^5 = 2000)$

Now, the distributor generates a usage license $L_U^1 = (K;\ Play; I_U^1 : T = [15/03/09, 19/03/09], R = [India]; A_U^1 = 800)$. L_U^1 satisfies all instance based constraints for L_D^1 and L_D^2. Let the validation authority randomly picks L_D^2 for validation then remaining *counts* in L_D^2 will be 200(i.e. 1000-800). Next, let the distributor generates $L_U^2 = (K; Play; I_U^2 : T = [21/03/09, 24/03/09], R = [Japan]; A_U^2 = 400)$. L_U^2 satisfies all the instance based constraints only for L_D^2. The validation authority will now consider L_U^2 as invalid as L_D^2 now cannot be used to generate more than remaining 200 counts. In this case, a better solution would be to validate L_U^1 using L_D^1, and L_U^2 using L_D^2. This will result in both L_U^1 and L_U^2 as valid licenses. Thus, the challenge is to do the validation such that the distributors can use their redistribution licenses in an intelligent way.

A Method for Validation: In this section, we present the symbols(see table 1) used and validation equations. Details about the derivation of validation equations can be found in [10].

An issued license is said to belong to a set S of redistribution licenses if it satisfies all the instance based constraints in all redistribution licenses in set S.

Table 1. List of Symbols

Symbol	Explanation
S^N	The set of all N redistribution licenses for a content i.e. $S^N = [L_D^1, L_D^2, ..., L_D^N]$.
$SB^r[S]$	The r^{th} subset of set S of redistribution licenses. If the set S contains k redistribution licenses then $r \leq 2^k - 1$, and $k \leq N$.
$C[S]$	Aggregate of the *permission counts* in all previously issued licenses which belong to the set S of redistribution licenses.
$A[S]$	Sum of aggregate *permission counts* in all redistribution licenses in the set S.
$C\langle S\rangle$	LHS of the validation equation for the set S.

For example, L_U^1 in example 1 belongs to the set $\{L_D^1, L_D^2\}$. $C[S]$ denotes the sum of permission counts in all previously issued licenses that belongs to the set S of redistribution licenses. Let the table 2 represents the licenses issued using redistribution licenses L_D^1, L_D^2,..., L_D^5 in example 1. After L_U^6 being issued, the value of $C[\{L_D^1,\ L_D^2\}]$, $C[\{L_D^2\}]$, $C[\{L_D^1,\ L_D^2,\ L_D^4\}]$, $C[\{L_D^3,\ L_D^5\}]$ and $C[\{L_D^5\}]$ will be 840, 400, 30, 800 and 20 respectively. $A[S]$ denotes the sum of aggregate permission counts in all the redistribution licenses in the set S. For example, $A[\{L_D^1,\ L_D^2,\ L_D^3\}]$ for the redistribution licenses in example 1 will be sum of aggregate permission count in L_D^1, L_D^2 and L_D^3 i.e. 2000+ 1000+ 3000= 6000.

Table 2. Table of log records

Issued Licenses	Set(S)	*Set Counts*(C)
L_U^1	$\{L_D^1,\ L_D^2\}$	800
L_U^2	$\{L_D^2\}$	400
L_U^3	$\{L_D^1,\ L_D^2\}$	40
L_U^4	$\{L_D^1,\ L_D^2,\ L_D^4\}$	30
L_U^5	$\{L_D^3,\ L_D^5\}$	800
L_U^6	$\{L_D^5\}$	20

If there exists N received redistribution licenses for a content then we need to do the validation using $2^N - 1$ validation equations [10], one for each subset of S^N(see table 1 for explanation). The validation equation for the r^{th} subset of S^N, $SB^r[S^N]$, is of the form:

$$\sum_{l=1}^{2^m-1} C[SB^l[\mathbf{SB^r}[\mathbf{S^N}]]] \leq A[\mathbf{SB^r}[\mathbf{S^N}]].$$
$$\text{where}, m = |\mathbf{SB^r}[\mathbf{S^N}]|,\ \text{and } 1 \leq m \leq N \qquad (1)$$

The LHS of the equation 1 does the summation of counts for the sets that are subset of the set $SB^r[S^N]$. Since there can be $2^m - 1$ subsets excluding the null set of a set containing m redistribution licenses therefore the summation has a limit from 1 to $2^m - 1$. The RHS is the summation of aggregate constraint counts in all redistribution licenses in the set $SB^r[S^N]$.

Example 2. consider five redistribution licenses in example 1. Since there are five redistribution licenses therefore N=5 in this case and total 2^5 – 1=31 validation equations are required. The validation equation for an example set $\{L_D^2, L_D^3, L_D^4\}$ will be $C[\{L_D^2\}]+C[\{L_D^3\}]+C[\{L_D^4\}]+C[\{L_D^2, L_D^3\}]+ C[\{L_D^2, L_D^4\}]+C[\{L_D^3, L_D^4\}]+C[\{L_D^2, L_D^3, L_D^4\}]\leq \mathrm{A}[\{L_D^2, L_D^3, L_D^4\}]$.

From table 1, LHS of equation 1 can also be referred as $C\langle SB^r[S^N]\rangle$. *Therefore, further in this paper, in short, we will refer validation equation for a set S(by replacing $SB^r[S^N]$ with S in equation 1) as:* $\mathbf{C}\langle\mathbf{S}\rangle \leq \mathbf{A}[\mathbf{S}]$.

Complexity of Validation: If N number of redistribution licenses are present with a distributor then $2^N - 1$ validation equations are possible. Let a newly generated license satisfies all instance based constraints in k redistribution licenses out of the total N redistribution licenses. The set formed by these k redistribution licenses will be a subset of $2^{(N-k)}$ sets thus it will be present in $2^{(N-k)}$ validation equations. So, we need to do the validation using $2^{(N-k)}$ validation equations. Validation using such a large number of validation equations every time a new license is issued is computationally intensive. Also, the violation of aggregate constraints is not a frequent event as the received redistribution licenses generally have sufficient *permission counts* to generate thousands of licenses. So, instead of doing validation every time a new license is issued, the aggregate validation is done offline by collecting the logs of the sets of redistribution licenses(to which the issued licenses satisfies all instance based constraints) and *permission counts* in issued licenses. The format of the log is as shown in table 2.

2.2 Overview of Validation Tree

In this section, we briefly discuss about construction of *validation tree* using the log records and validation using *validation tree*[10].

Validation Tree Construction: As shown in figure 1, each node in the *validation tree* stores the following fields: name of a redistribution license(L), a count(C) and links to its child nodes. The count value C determines the count associated with the set formed by the redistribution license in the node and all its prefix nodes(nodes in the path from root node to the current node) in the same branch. For example, count=840 for the node $root \rightarrow L_D^1 \rightarrow L_D^2$ implies that the *set count* associated with the set $\{L_D^1, L_D^2\}$ is 840. In the *validation tree* child nodes of a node are ordered in increasing order of their indexes. To generate the *validation tree* initially a *root* node is created. Validation tree is then expanded by inserting the log records using the insertion algorithm(algorithm 1). The insertion algorithm one by one reads the records in the log and then one by one reads the indexes of redistribution licenses in each record. In algorithm 1, let the record to be inserted currently be given by R=[r, R'], where r is the first

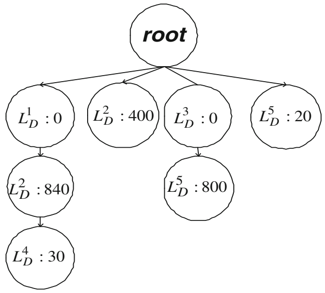


Fig. 1. The validation tree

redistribution license and R' is the set of remaining redistribution licenses and let *count* denotes the set count associated with record R. Algorithm 1 inserts the record R in the validation tree with the *root* node initially denoted by T. The algorithm traverses the validation tree according to the redistribution licenses in the log record and inserts/adds the count in the last node traversed for the record. Figure 1 shows the validation tree generated for the records in table 2.

Algorithm 1. *Insert*(T, R, *count*)

```
1. Sequentially traverse the child nodes of $T$ until all child nodes are traversed or we find a child $T'$ such that $T.L \geq r$.
2. If $T'.L=r$ then go to step 4.
3. Else add a node $T'$ such that $T'.L=r$ and $T'.C=0$ as the child node of $T$(in the order).
4. If $R'$=null set then $T'.C= T'.C+count$. Else, call Insert($T'$, $R'$, $count$).
```

Validation Using Validation Tree: Algorithm 2 is used to do the validation for all possible validation equations. The parameter i takes care that validation is done for all possible sets. If N redistribution licenses are present, it takes value from i=1 to i=$2^N - 1$. Each value of i corresponds to a particular set of redistribution licenses(and hence validation equation), which is determined by bits=1 in binary format of i, e.g. i=7 has first, second and third bits from LSB as 1 and rest as 0. So, it represents validation equation for set $\{L_D^1, L_D^2, L_D^3\}$.

The algorithm calculates and compares LHS(CV) and RHS(AV) of each validation equation(see equation 1) to determine whether an equation is valid or not. The RHS of each validation equation is calculated using an array A of size N containing aggregate constraint values in all N received redistribution licenses. The j^{th} element, $A(j)$, of A contains the aggregate constraint value in the j^{th} received redistribution license. Since the set of redistribution license is decided by the position of bits=1 in i in algorithm 2 so we only need to add the aggregate constraint values for the redistribution licenses that correspond to bits=1. It is taken care by left shift and AND operation in algorithm 2. Parameter $licNumber$ in algorithm 2 is calculated to facilitate the traversal of validation tree. It is equal to the number of redistribution licenses corresponding to the current validation equation(or number of bits=1 in i). The LHS for the validation equation for a set S does the summation of set counts for set S and all its subsets. It is calculated by traversing the validation tree. Detailed about tree traversal algorithm are out of scope of this paper and can be found in [10].

Algorithm 2. Validation($root$, A, N)

```
Temporary Variables:$AV$=0, $CV$=0
for $i$=1 to $2^N - 1$ do
    $licNumber$=0.
    for $j$=1 to $N$ do
        if $(1 << (j-1)\ AND\ i) \neq 0$ then
            /* <<: left shift operator */
            $AV$=$AV$+ $A(j)$.
            $licNumber$=$licNumber$+1;
    Call $CV$=$VaLHS(root, i, licNumber)$.

    if $CV \leq AV$ then
        Declare(Valid Equation).
    else
        Declare(Invalid Equation)
```

3 Proposed Efficient Validation Approach

In this section, we present a geometric approach to identify the redundant validation equations and a method to remove the redundant validation equations.

3.1 Geometric Representation of Licenses

If there are M number of instance based constraints in the licenses then each redistribution /usage license can be represented as an M dimensional hyper-rectangle e.g. figure 2 shows five redistribution licenses and two usage licenses represented in a 2-dimensional space. The set of redistribution licenses to which an issued license can be instance validated is given by the set of redistribution licenses whose hyper-rectangles completely contain the hyper-rectangle formed by the issued license. For example, in figure 2, hyper-rectangle formed by L_U^1 is completely within L_D^4 only. Thus, the set of redistribution licenses to which L_U^1 can be instance based validated is given by $\{L_D^4\}$. However, L_U^2 is not completely within any of the 5 redistribution licenses so it cannot be instance based validated using any of the 5 redistribution licenses and it is treated as invalid.

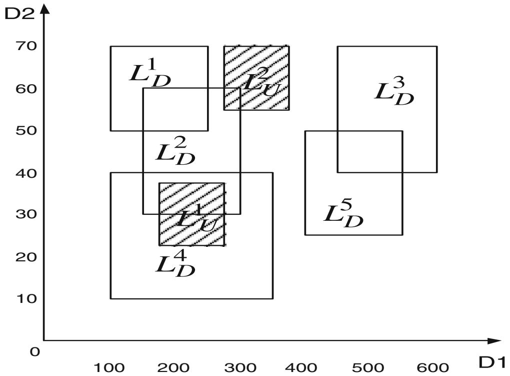


Fig. 2. Representation of 5 received redistribution licenses with two constraints

3.2 Redundant Validation Equations

In this section we first define overlapping redistribution licenses and non-overlapping sets of redistribution licenses. Then with the help of theorems 1 and 2, we present a method to identify the redundant validation equations.

Overlapping Redistribution Licenses. Two redistribution licenses, L_D^j and L_D^k, with instance based constraints $\{I_1^j, I_2^j, ..., I_M^j\}$ and $\{I_1^k, I_2^k, ..., I_M^k\}$, are overlapping if $I_m^j \cap I_m^k \neq \emptyset$, $\forall$ $m \leq M$, where M is the number of total instance based constraints. Geometrically, two redistribution licenses L_D^j and L_D^k are overlapping if all their constraint dimensions have an overlap. Thus, in figure 2, licenses L_D^1 and L_D^2 are overlapping but L_D^1 and L_D^4 are non-overlapping.

Non Overlapping Sets. Two sets S_1 and S_2 of received redistribution licenses are said to be non overlapping if S_1 and S_2 contain different licenses(i.e. $S_1 \cap S_2 = \emptyset$) and any of the constraint ranges in the license in the set S_1 does not overlap with any license in the set S_2. For example, the sets $S_1 = \{L_D^1, L_D^2\}$ and $S_2 = \{L_D^5\}$ are non overlapping in figure 2 as neither L_D^1 nor L_D^2(in set S_1) overlaps with L_D^5(in set S_2).

Theorem 1. *If all the redistribution licenses in a set S do not have a common overlapping region then count($C[S]$) associated with that set will always be 0.*

Proof. If all the redistribution licenses in a set S do not have a common region then any valid issued license cannot be simultaneously within the hyper-rectangles formed by all redistribution licenses in the set. Thus, the count value for the set will always be 0. For example, redistribution licenses L_D^1, L_D^2, and L_D^3 in figure 2 do not form a common region and hence the hyper-rectangle formed by any issued license cannot be inside the hyper-rectangles of L_D^1, L_D^2, and L_D^3 simultaneously. So, $C[\{L_D^1, L_D^2, L_D^3\}]$ will always be 0.

Corollary 1.1. Set S of received redistribution licenses formed by taking at least one redistribution license from two or more non overlapping sets of received redistribution licenses will always have *set count* equal to 0. This is because if we take at least 1 received redistribution licenses from two different non overlapping sets of received redistribution licenses then a common region cannot be formed by all the received redistribution licenses in S.

Theorem 2. *If a set S containing n received redistribution licenses can be represented with m number of non overlapping sets($S_i \cap S_j = \emptyset$, $1 \leq i < j \leq m \leq n$) of received redistribution licenses such that $S_1 \cup S_2 \cup ... \cup S_m = S$ then the validation equation for the set S can be expressed as the sum of validation equations for the sets S_1, S_2, ... and S_m. Validation equation for a set S_i is: $C\langle S_i \rangle \leq A[S_i]$.*

$$\begin{aligned} &C\langle S \rangle = C\langle S_1 \rangle + C\langle S_2 \rangle + C\langle S_3 \rangle + ... + C\langle S_m \rangle \leq \\ &A[S] = A[S_1] + A[S_2] + A[S_3] + ... + A[S_m] \end{aligned} \tag{2}$$

Thus, if the validation equations for the sets S_1, S_2, ... and S_m are already validated then we do not need to validate the equation for the set S.

Proof. m number of non overlapping sets of received redistribution licenses are represented by S_1, S_2, ...S_m, where $2 \leq m \leq N$. Now consider a set formed by at least one received redistribution license each from $m^{'}$ $(2 \leq m^{'} \leq m)$sets out of total m sets. As $S_i \cap S_j = \emptyset$ therefore using corollary 1.1, any such set formed will always have the *set count* value equal to 0. Thus, all the sets which are created using taking at least one received redistribution license each from any $m^{'}$ sets out of m sets will always have count value equal to 0. So, $C\langle S \rangle$ will be only due to the sets formed due to redistribution licenses within each of the m sets individually. Thus, $C\langle S \rangle$ can be written as: $C\langle S \rangle = C\langle S_1 \rangle + C\langle S_2 \rangle + C\langle S_3 \rangle + ... + C\langle S_m \rangle$. Also, the sets S_1, S_2, ... and S_m are non overlapping and $S_1 \cup S_2 \cup ... \cup S_m = S$. Therefore, $A[S]$ can be written as: $A[S] = A[S_1] + A[S_2] + A[S_3] + ... + A[S_m]$.

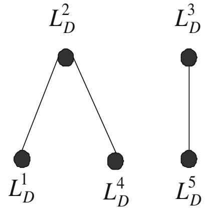
Thus, equation 2 is correct and it can be obtained by summation of the validation equations for the individual sets S_1, S_2, ... and S_m. Hence if the validation equations for the sets S_1, S_2, ... and S_m are evaluated then the validation equation for the set S can be removed.

For example, the redistribution licenses in figure 2 can be divided in two groups, with group 1 and group 2 containing the redistribution licenses (L_D^1, L_D^2, L_D^4) and (L_D^3, L_D^5) respectively. According to theorem 2, if validation equations for all subsets of the sets $\{L_D^1, L_D^2, L_D^4\}$ and $\{L_D^3, L_D^5\}$ are evaluated independently then we need not evaluate the validation equations for any set generated by taking at least one redistribution license from each group. So, equations for the sets $\{L_D^1, L_D^3\}$, $\{L_D^1, L_D^2, L_D^3\}$, etc. need not be evaluated.

3.3 Identification of Disconnected Groups

To use the basis discussed in the previous sub-section, we need to identify the disconnected groups of redistribution licenses. In this section, we present an algorithm to find the disconnected groups of redistribution licenses.

We use a graph[12][3] G=(V, E) to identify the disconnected groups of redistribution licenses, where V is the set of vertices and E is the set of edges in G. As shown in figure 3, each redistribution license is represented using a vertex in G. There is an edge between the vertices corresponding to the i^{th} and j^{th} redistribution licenses if they are overlapping. We represent the graph using an adjacency matrix *Adj* of size N X N. The entry in the i^{th} row and j^{th} column, $Adj_{i,j}$, is 1 if the i^{th} and j^{th} redistribution licenses are overlapping else it is 0. Figure 3 shows the graph and matrix *Adj* for redistribution licenses in figure 2.



	L_D^1	L_D^2	L_D^3	L_D^4	L_D^5
L_D^1	0	1	0	0	0
L_D^2	1	0	0	1	0
L_D^3	0	0	0	0	1
L_D^4	0	1	0	0	0
L_D^5	0	0	1	0	0

Fig. 3. Graph and adjacency matrix for redistribution licenses in figure 2

We perform depth first search(DFS)[12] in algorithm 3 on G to identify the number of groups of redistribution licenses and redistribution licenses in each group. The DFS starts from the node corresponding to the first redistribution license. It finds all the nodes that are directly or indirectly(through other nodes) connected to the node corresponding to the first redistribution license. All such nodes form one group of redistribution licenses. Next, the algorithm sequentially searches for a redistribution license, which is not present in any previously formed groups. Then, finds all the nodes that are directly or indirectly connected to the node corresponding to the searched redistribution license. The process continues until a group is allocated to every redistribution license. To perform DFS, two arrays viz. *Visited* and *Group* of size N and N X N, respectively, are initialized with each element equal to 0 in algorithm 3. Array *Visited* keeps information about all the nodes traversed in the graph during the DFS. Array *Group* is modified by the algorithm and after completion of algorithm it stores the information about the redistribution licenses in different groups of redistribution licenses. Each row in *Group* corresponds to a group of redistribution licenses and stores the information about the redistribution licenses present in that group. If j^{th} redistribution license is in the i^{th} group then $Group_{i,j}$=1 else it is 0. Please note that the number of rows in *Group* are N as maximum number of groups formed can be N(assuming no connected licenses). If there are g number of groups then only first g rows give information about groups and redistribution licenses in each group. For example, the first two rows in array *Group* for the graph in figure 3 are given by (1, 1, 0, 1, 0) and (0, 0, 1, 0, 1) as the redistribution licenses in the first and second group are (L_D^1, L_D^2, L_D^4) and (L_D^3, L_D^5) respectively. Rest of the rows have all entries 0. Algorithm 3 also calculates array *GroupSize*, which determines the number of redistribution licenses in each group.

Algorithm 3. Group Formation

```
int $g$=0.
array $Visited$: Size $N$. All elements are initialized to 0.
array $Group$: Size $NXN$. All elements are initialized to 0.
array $GroupSize$: Size $N$. All elements are initialized to 0.
$Adj$: Adjacency matrix of the graph.
for $i$=1 to $i \leq N$ do
    if $Visited[i]$=0 then
        $g$=$g$+1.
        Call Depth_first($i$, $g$).
Print(Number of groups=$g$).
Subroutine:Depth_first($i$, $k$)
{
$Group[k][i]$=1, $Visited[i]$=1
$GroupSize[k]$=$GroupSize[k]$+1.
for $j$=$i$+1 to $i \leq N$ do
    if $Adj[i][j]$=1 and $Visited[j]$=0 then
        Call Depth_first($j$, $k$).
}
```

4 Validation Algorithm

In this section, we modify the *validation tree* [10] to enable it to use the approach in section 3 for efficient validation. We do this by dividing the original *validation tree* in g parts and modifying the index of the nodes in each newly generated *validation tree*, where g is the number of groups of redistribution licenses determined by algorithm 3.

4.1 Division of Validation Tree

Using corollary 1.1, a set formed by taking at least one redistribution license from two or more groups out of the total g groups will have *set count* equal to 0. So, in the log records any such set cannot exist. This implies that in *validation tree* any branch will not contain the redistribution licenses from two or more groups out of the total g groups. For example, in figure 2, any issued license cannot belong to the sets $\{L_D^1, L_D^3\}$ or $\{L_D^1, L_D^3, L_D^5\}$. So these will not be present in the logs and there cannot be branches $root \rightarrow L_D^1 \rightarrow L_D^3$ or $root \rightarrow L_D^1 \rightarrow L_D^3 \rightarrow L_D^5$. Thus, the *validation tree* can be divided into g independent parts; each part contains the nodes corresponding to the redistribution licenses in a particular group. Algorithm 4 (by calling $Separation(root, g)$) is used to divide the original *validation tree* into g new *validation trees*. The algorithm checks for the group to which a child node of the *root* node belongs. If a child node belongs to the j^{th} group then link the child nodes to the j^{th} newly generated validation tree with root node as $root_j$. Figure 4 shows two validation trees obtained after division of the validation tree in figure 1.

Algorithm 4. *Separation*(T, g)

```
Initialize $m$ number of root nodes.
Let the root of the $i^{th}$ validation tree be defined as $root_i$.
foreach Child of $T$ do
    Let current child be $T'$
    if $T'.index \in Group[j]$ then
        Link $T'$ as child node of $root_j$.
```

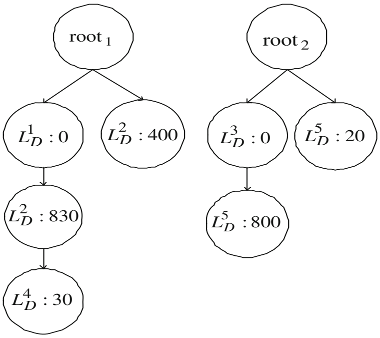


Fig. 4. Division of Validation tree

4.2 Modification of Indexes

In this step, the indexes of the nodes in each new validation tree are modified. This is to ensure that if $N_k(=GroupSize[k]$, calculated in algorithm 3) redistribution licenses are in the k^{th} group then the indexes of nodes in the k^{th} validation tree vary from 1 to N_k, which is required for the validation algorithm (algorithm 2). Algorithm 5 is used to modify the indexes for each validation tree. To modify the indexes of the nodes in the k^{th} validation tree (with root node $root_k$), algorithm *Modification*($root_k$, A_k) is called. The algorithm calculates and uses an array named $position_k$ to modify the indexes. If a redistribution license(say j^{th}) is in the k^{th} group then the value stored at $position_k[j]$ determines the new index of the j^{th} redistribution license. The original index is replaced by the new index in the last step in algorithm 5. For instance, consider array $position_2$ for the second validation tree in figure 4. According to algorithm 5, it will be (0, 0, 1, 0, 2). So, the indexes 3 and 5 in the nodes of the second *validation tree* in figure 4 are replaced by 1 and 2 respectively in figure 5. The *validation trees* in figure 4 after modification of indexes are shown in figure 5.

For each group, we also need to calculate the array containing the aggregate constraint values in the redistribution licenses in the group. Let it be A_k for the k^{th} group (corresponding to k^{th} validation tree). The size of array A_k is given by N_k, where N_k is the number of redistribution licenses in the k^{th} group. A_k is derived using the initial array A(see section 2.2) containing aggregate constraint counts for all redistribution licenses. The procedure for derivation

Algorithm 5. *Modification*($root_k$, A_k)

```
array Group: Same as calculated in algorithm 3.
array $position_k$: size $N$. All elements are initialized to 0.
$p$=1.
for $j=1$ to $N$ do
    if $Group[k][j]=1$ then
        $position_k[j]=p$.
        $A_k[p]=A[j]$
        $p=p+1$.
Traverse the $k^{th}$ validation tree.
for Each node traversed(except $root_k$) do
    Let the index of the node be given by index.
    Replace index by $position_k[index]$.
```

of A_k using A is mentioned in algorithm 5. After the modification of indexes and calculation of aggregate constraint array, we can directly use the validation algorithm(algorithm 2) in section 2.2 for each validation tree. So, we call Validation($root_k$, A_k, N_k) for each value of $k \leq g$, where g is the total number of groups of redistribution licenses.

The validation time performance gain of our proposed approach depends on the number of groups formed and number of redistribution licenses in each group. If N_k number of redistribution licenses are present in the k^{th} group then $2^{N_k}-1$ validation equations are needed for the k^{th} group. Therefore, the total number of validation equations required to validate will be $\sum_{k=1}^{g}(2^{N_k}-1)$. Whereas, without using our approach the total number of validation are 2^N-1. Thus, the approximate performance gain(G) can be given as:

$$G \approx \frac{2^N-1}{\sum_{k=1}^{g}(2^{N_k}-1)} \tag{3}$$

The value of G varies from 1 to $(2^N-1)/N$ for m=1 to m=N. Thus, the performance gain always remains greater than or equal to 1.

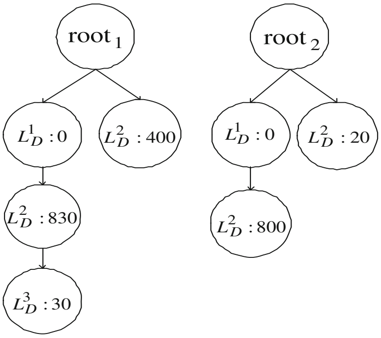


Fig. 5. Modification of Indexes

As an illustration, consider the five redistribution licenses in example 1. These redistribution licenses can be divided in two groups (group1:(L_D^1, L_D^2, L_D^4) and group2:(L_D^3, L_D^5)). So, the approximate gain in this case would be $(2^5-1)/((2^3-1)+(2^2-1))$=3.1 times.

5 Performance Analysis

In this section, we analyze the theoretical and experimental performance of our proposed algorithm in terms of validation time complexity, storage space requirements, and validation tree division and modification time. All the experiments were performed on Intel(R) core(2) 2.40 GHZ CPU with 2 GB RAM. All the programs are written in Java. To perform the experiments, first we created a number of redistribution licenses and issued licenses. The set of redistribution licenses for which an issued license satisfies all instance based constraints along with the aggregate constraint counts is saved in the log records. For the experiments, each redistribution license is assumed to contain 4 instance based constraints and aggregate constraint count in between 5000 and 20000. Each issued license is assumed to contain permission counts in between 10 and 30. The number of log records in our experiments varies from about 600 for n=1 redistribution license to 22000 for n=35 redistribution licenses.

A. Validation Time Complexity: First, we show the variation of number of groups of redistribution licenses with the number of redistribution licenses(N) in figure 6. The number of groups in our experiments varies from 1 to 5 for different values of N. As in figure 6, the number of groups may remain same increase or decrease after addition of a new redistribution license. For example, let in figure 2 a new redistribution license L_D^6 is added. The number of groups will remain same if L_D^6 is connected to redistribution licenses in only one group out of two existing groups. The number of groups will increase(increased to 3) if L_D^6 is not connected to redistribution licenses in any group. The number of

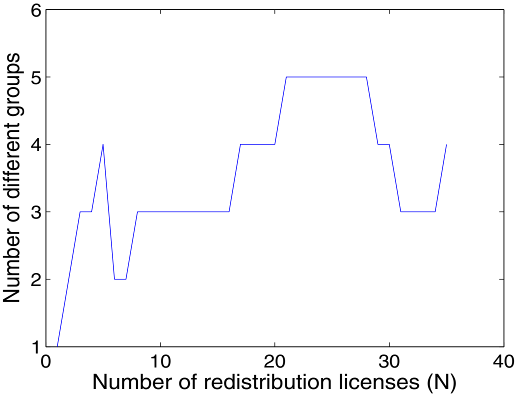


Fig. 6. Variation of number of groups

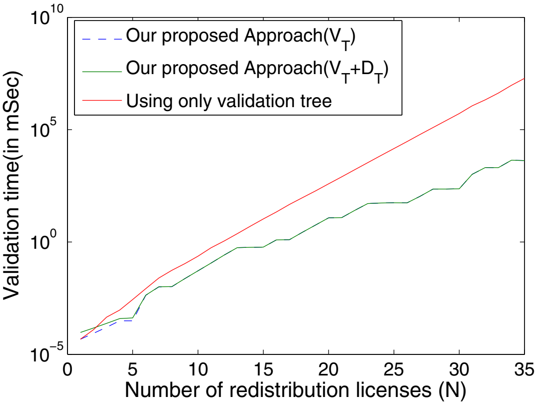


Fig. 7. Validation Time Complexity

groups will decrease(decreased to 1) if L_D^6 is connected to redistribution licenses in both groups.

Figure 7 shows a comparative analysis of validation time(V_T) required using our proposed method with the approach in [10]. To show that the time required for division(D_T) of the original validation tree is small as compared to V_T, we also plot the curve for V_T+D_T. From the experiments, it can be observed that using our proposed method the validation time is significantly reduced. Also, the effect of D_T becomes very small as compared to V_T for N>2.

To show whether the result is in accordance with equation 3, we also compare the theoretical(using equation 3) and experimental gain in figure 8. We observe that the experimental gain is always greater or equal to the theoretical gain. This is because when we divide the *validation tree* in g parts, we only traverse one validation tree out of g *validation trees*. Due to this some redundant traversals required in the original validation tree are not required.

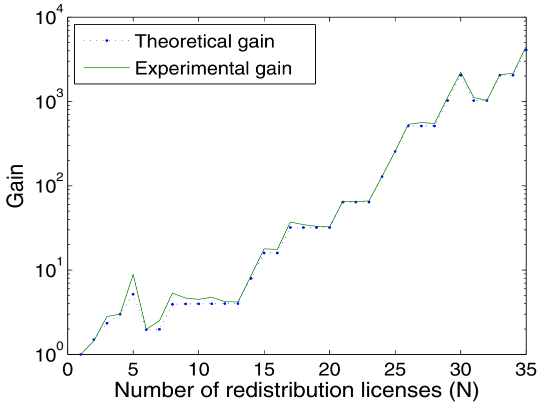


Fig. 8. Theoretical Vs. Experimental Gain

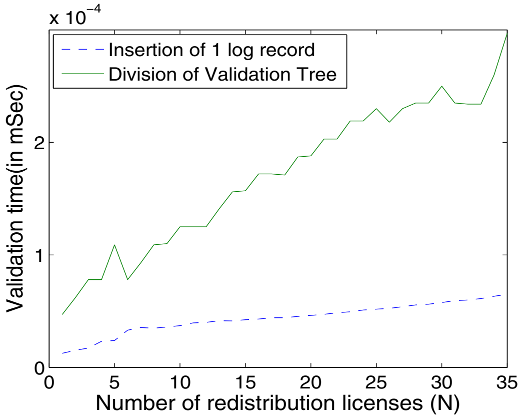


Fig. 9. Insertion time complexity

B. Construction Time of Data Structure: The construction time of the data structure is the sum of *validation tree* construction time(C_T) and time required for division(D_T) of the original *validation tree*. In comparison to [10], the additional time required is D_T. The time required to divide the *validation tree* includes the time required to identify the groups of redistribution licenses using graph and traverse the child nodes of the *root* node. Figure 9 shows the comparison between the time required to insert 1 log record and time required for division of *validation tree*. The time required for the division of *validation tree* is only 3-4 times as compared to time required for insertion of a single record. But, the insertion process generally requires thousands of log records insertion whereas division of *validation tree* is performed only once. So, the overhead due to the *validation tree* division is very small.

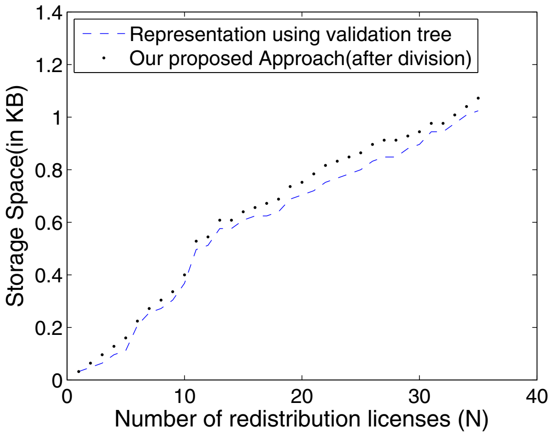


Fig. 10. Storage space complexity

C. Storage Space Requirement: Storage Space requirement is equal to the space required to store the *validation tree(s)*. Figure 10 compares the storage space required for storing new *validation trees* after the division of original *validation tree* with the original *validation tree*. As no new nodes(except the *root* nodes) for the validation trees generated after the division of *validation tree* so the storage space requirement will also be almost same for both original *validation tree* and *validation trees* formed after division.

6 Conclusion

In this paper, we presented an efficient method of doing license validation in DRM systems by removing the redundant validation equations. For this purpose, first we proposed a geometry based approach to identify the redundant validation equations. Then we removed the redundant validation equations by grouping of redistribution licenses and division of the *validation tree*. The theoretical analysis and experimental results show that the proposed method can do the validation much efficiently as compared to doing using the original *validation tree*. The experiments show that the time required for identification of redundant validation equations and division of the *validation tree* is very small. Also, the storage and insertion time complexities are almost same as that required for original *validation tree* based validation method.

Acknowledgment. Thanks to the Agency for Science, Technology and Research (A-STAR), Singapore for supporting this work under the project "Digital Rights Violation Detection for Digital Asset Management" (Project No: 0721010022).

References

1. Conrado, C., Petkovic, M., Jonker, W.: Privacy-preserving digital rights management. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178, pp. 83–99. Springer, Heidelberg (2004)
2. Diehl, E.: A four-layer model for security of digital rights management. In: Proceedings of the 8th ACM workshop on Digital rights management, pp. 19–28 (2008)
3. Gross, J., Yellen, J.: Handbook of Graph Theory and Applications. CRC Press, Boca Raton (2003)
4. Hilty, M., Pretschner, A., Basin, D., Schaefer, C., Walter, T.: A policy language for distributed usage control. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 531–546. Springer, Heidelberg (2007)
5. Hwang, S.O., Yoon, K.S., Jun, K.P., Lee, K.H.: Modeling and implementation of digital rights. The Journal of Systems and Software 73(3), 533–549 (2004)
6. Iannella, R.: Digital rights management (DRM) architectures. D-Lib Magazine 7(6) (2001)
7. Knuth, D.E.: The art of computer programming. Sorting and searching, vol. 3. Addison Wesley Longman Publishing Co., Inc., Redwood City (1998)
8. Liu, G., Lu, H., Lou, W., Xu, Y., Yu, J.X.: Efficient mining of frequent patterns using ascending frequency ordered prefix-tree. Data Mining and Knowledge Discovery 9(3), 249–274 (2004)

9. Sachan, A., Emmanuel, S., Kankanhalli, M.S.: Efficient license validation in MPML DRM architecture. In: 9th ACM workshop on digital rights management(DRM 2009), Chicago, pp. 73–82 (2009)
10. Sachan, A., Emmanuel, S., Kankanhalli, M.S.: Efficient aggregate licenses validation in DRM. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 313–319. Springer, Heidelberg (2010)
11. Safavi-Naini, R., Sheppard, N.P., Uehara, T.: Import/export in digital rights management. In: Proceedings of the 4th ACM workshop on Digital rights management, pp. 99–110. ACM, New York (2004)
12. West, D.: Introduction to graph theory. Prentice Hall, Upper Saddle River (2001)

Differentially Private Data Release through Multidimensional Partitioning

Yonghui Xiao[1,2], Li Xiong[1], and Chun Yuan[2]

[1] Emory University, Atlanta GA 30322, USA
[2] Tsinghua University Graduate School at Shenzhen, Shenzhen 518055, China

Abstract. Differential privacy is a strong notion for protecting individual privacy in privacy preserving data analysis or publishing. In this paper, we study the problem of differentially private histogram release based on an interactive differential privacy interface. We propose two multidimensional partitioning strategies including a baseline cell-based partitioning and an innovative kd-tree based partitioning. In addition to providing formal proofs for differential privacy and usefulness guarantees for linear distributive queries , we also present a set of experimental results and demonstrate the feasibility and performance of our method.

1 Introduction

As information technology enables the collection, storage, and usage of massive amounts and types of information about individuals and organizations, privacy becomes an increasingly important issue. Governments and organizations recognize the critical value in sharing such information while preserving the privacy of individuals. Privacy preserving data analysis and data publishing [5,10,3] has received considerable attention in recent years as a promising approach for sharing information while preserving data privacy. There are two models for privacy protection [5]: the interactive model and the non-interactive model. In the interactive model, a trusted *curator* (e.g. hospital) collects data from *record owners* (e.g. patients) and provides an access mechanism for *data users* (e.g. public health researchers) for querying or analysis purposes. The result returned from the access mechanism is perturbed by the mechanism to protect privacy. In the non-interactive model, the curator publishes a "sanitized" version of the data, simultaneously providing utility for data users and privacy protection for the individuals represented in the data.

Differential privacy [6,4,5,3] is widely accepted as one of the strongest known unconditional privacy guarantees with the advantage that it makes no assumption on the attacker's background knowledge. It requires the outcome of computations to be formally indistinguishable when run with and without any particular record in the dataset, as if it makes little difference whether an individual is being opted in or out of the database. Many meaningful results have been obtained for the interactive model with differential privacy [6,4,5,3]. Non-interactive data release with differential privacy has been recently studied with hardness

W. Jonker and M. Petković (Eds.): SDM 2010, LNCS 6358, pp. 150–168, 2010.
© Springer-Verlag Berlin Heidelberg 2010

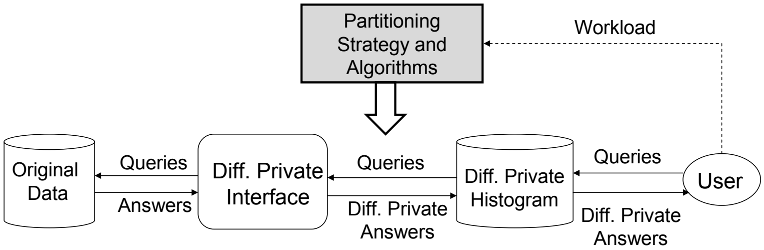


Fig. 1. Differentially Private Histogram Release

results obtained and it remains an open problem to find efficient algorithms for many domains [7].

In this paper, we study the problem of differentially private histogram release based on an interactive differential privacy interface, as shown in Figure 1. A *histogram* is a disjoint partitioning of the database points with the number of points which fall into each partition. An interactive differential privacy interface, such as the Privacy INtegrated Queries platform (PINQ) [19], provides a differentially private access to the raw database. An algorithm implementing the partitioning strategy submits a sequence of queries to the interface and generates a differentially private histogram of the raw database. The histogram can then serve as a sanitized synopsis of the raw database and, together with an optional synthesized dataset based on the histogram, can be used to support count queries and other types of OLAP queries and learning tasks.

An immediate question one might wonder is what is the advantage of the non-interactive release compared to using the interactive mechanism to answer the queries directly. A common mechanism providing differential private answers is to add carefully calibrated noise to each query determined by the privacy parameter and the sensitivity of the query. The composability of differential privacy [19] ensures privacy guarantees for a sequence of differentially-private computations with additive privacy depletions in the worst case. Given an overall privacy requirement or budget, expressed as a privacy parameter, it can be allocated to subroutines or each query in the query sequence to ensure the overall privacy. When the number of queries grow, each query gets a lower privacy budget which requires a larger noise to be added. When there are multiple users, they have to share a common privacy budget which degrades the utility rapidly. The non-interactive approach essentially exploits the data distribution and the query workload and uses a carefully designed algorithm or query strategy such that the overall noise is minimized for a particular class of queries. As a result, the partitioning strategy and the algorithm implementing the strategy for generating the query sequence to the interface are crucial to the utility of the resulting histogram or synthetic dataset.

Contributions. We study two partitioning strategies for the differentially private histogram release for random query workload and evaluate their utility. We summarize our contributions below.

- We study two partitioning strategies for the differentially private histogram release problem: 1) a baseline strategy using the most fine-grained cell partitioning, and 2) a kd-tree based partitioning strategy. There are several innovative features in our kd-tree based strategy. First, we incorporate a uniformity measure in the partitioning process which seeks to produce partitions that are close to uniform so that approximation errors within partitions are minimized. Second, we implement the strategy using a two-step algorithm that generates the kd-tree partitions based on the histogram generated from a cell partitioning so that the access to the differentially private interface is minimized.
- We present formal results and discuss the applications of the histogram for general online analytical processing (OLAP) and learning, and present a set of experimental evaluations and show the actual performance of our algorithm. We show that the cell partitioning strategy, while simple, provides formal bounded utility for linear distributive queries including count and sum. The kd-tree based partitioning, on the other hand, achieves better results empirically.

2 Related Works

Privacy preserving data analysis and publishing has received considerable attention in recent years. We refer readers to [5,10,3] for several up-to-date surveys. We briefly review here the most relevant work to our paper and discuss how our work differs from existing work.

There has been a series of studies on interactive privacy preserving data analysis based on the notion of differential privacy [6,4,5,3]. A primary approach proposed for achieving differential privacy is to add Laplace noise [6,5,4] to the original results. McSherry and Talwar[21] give an alternative method to implement differential privacy based on the probability of a returned result, called the exponential mechanism. Roth and Roughgarden [23] proposes a median mechanism which improves upon the Laplace mechanism. McSherry implemented the interactive data access mechanism into PINQ[19], a platform providing a programming interface through a SQL-like language.

There are recently a few works that studied general non-interactive data release with differential privacy [6,2,8,24]. Blum et al. [2] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. It also proposed an inefficient algorithm based on the exponential mechanism. The result largely remains theoretical and the general algorithm is inefficient for the complexity and required data size. Feldman et al. [8] proposed the notion "private coreset" to release data for certain queries: k-median, k-mean, k-center. X. Xiao et al. [24] developed a differentially private data release algorithm for predicate queries using wavelet transforms. In addition, several recent work studied differentially private mechanisms for particular kinds of data such as search logs [16,11] or for specific applications such as recommender systems [20] or record linkage [14].

It is important to note that [14] uses several tree strategies including k-d tree in its partitioning step and our results show that our our uniformity-driven k-d tree strategy achieves better utility for random count queries. Another closely related work is [13] which generates differentially private histograms for single dimensional range queries through a consistency check technique. Several works [12,17] studies mechanisms for a given query workload. [12] proposes an enhanced Laplace mechanism by examining the geometry shape of a given set of linear queries. [17] proposes a query matrix mechanism that generates an optimal query strategy based on the query workload of linear count queries. It is worth noting that the cell-based partitioning in our approach is essentially the identity query matrix referred in [17]. On the other hand, our kd-tree based partitioning will generate a query matrix that is dependent on the approximate data distribution.

In summary, our work complements and advances the above works in that we focus on differentially private histogram release for random query workload using a multidimensional partitioning approach that is "data-aware". The method provides a formal utility guarantee for a set of queries and also supports applications for general OLAP and learning.

3 Preliminaries and Definitions

Given an original database D, we use $\mathcal{A}(D)$ to denote an interactive mechanism to access the database D. For a query $Q(D)$ the interactive query mechanism returns a perturbed result of $\mathcal{A}_Q(D)$. In the non-interactive mechanism, our goal is to release a database $\hat{D}$ to answer user queries, which satisfies differential privacy. For simplicity, we assume the output range of queries is arbitrary. In this section, we formally introduce the definitions of differential privacy, (ϵ, δ)-usefulness, and the notion of a data cube to facilitate our discussions.

3.1 Differential Privacy

Definition 1 (α-Differential privacy[4]). *In the interactive model, an access mechanism $\mathcal{A}$ satisfies α-differential privacy if for any neighboring databases*[1] *D_1 and D_2, for any query function Q, $r \subseteq Range(Q)$, $\mathcal{A}_Q(D)$ is the mechanism to return an answer to query $Q(D)$,*

$$Pr[\mathcal{A}_Q(D_1) = r] \leq e^{\alpha} Pr[\mathcal{A}_Q(D_2) = r]$$

In the non-interactive model, a data release mechanism $\mathcal{A}$ satisfies α-differential privacy if for all neighboring database D_1 and D_2, and released output $\hat{D}$,

$$Pr[\mathcal{A}(D_1) = \hat{D}] \leq e^{\alpha} Pr[\mathcal{A}(D_2) = \hat{D}]$$

[1] We use the definition of neighboring databases consistent with [19] which treats the databases as multisets of records and requires their symmetric difference to be 1.

To achieve differential privacy, we use the Laplace mechanism [6] that adds random noise of Laplace distribution to the true answer of a query Q, $\mathcal{A}(x) = Q(x) + Y$, where Y is the Laplace noise. The magnitude of the noise depends on the privacy level and the query's sensitivity.

Definition 2 (Sensitivity). *For arbitrary neighboring databases D_1 and D_2, the sensitivity of a query Q is the maximum difference between the query results of D_1 and D_2,*

$$GS_Q = max|Q(D_1) - Q(D_2)|$$

To achieve α-differential privacy for a given query Q on dataset D, it is sufficient to return $Q(D) + Y$ in place of the original result $Q(D)$ where we draw Y from $Lap(GS_Q/\alpha)$ [6].

3.2 Composition

The composability of differential privacy [19] ensures privacy guarantees for a sequence of differentially-private computations. For a general series of analysis, the privacy parameter values add up, i.e. the privacy guarantees degrade as we expose more information. In a special case that the analyses operate on disjoint subsets of the data, the ultimate privacy guarantee depends only on the worst of the guarantees of each analysis, not the sum.

Theorem 31 (Sequential Composition [19]). *Let M_i each provide α_i-differential privacy. The sequence of M_i provides $(\sum_i \alpha_i)$-differential privacy.*

Theorem 32 (Parallel Composition [19]). *If D_i are disjoint subsets of the original database and M_i provides α-differential privacy for each D_i, then the sequence of M_i provides α-differential privacy.*

3.3 Sufficient Bound of α

The level of differential privacy is determined by the parameter α. However, there is no specific guidelines on how to choose a proper α value. We attempt to analyze the risk of large α, and give a sufficient condition of α to guarantee privacy through the analysis of the prior and posterior probability of a value disclosure.

Theorem 33. *Assume an attacker has a priori belief for a target victim's value being d as P_0. After l queries that include the target victim using the Laplace noise based differential privacy mechanism, we have $(l\alpha)$-differentially privacy [19]. The posteriori belief after l queries, P_l, satisfies $P_l \leq P_0 * exp(l\alpha)$. (Proof in Appendix A)*

Corollary 31 (Sufficient bound of α). *$\alpha < ln(x)/l$ is a sufficient bound for guaranteeing $P_l/P_0 < x$. $\alpha < -ln(P_0)/l$ is a sufficient bound for guaranteeing $P_l < 1$.*

3.4 Data Cube

We use a "data cube" to represent the data space $\mathcal{D}^n$. For example, if the database has N dimensions, it is an N-dimensional cube. We denote the data cube by Ω, a query $Q : \Omega \rightarrow \mathcal{R}$ maps the data in Ω to output range $\mathcal{R}$. All the records in the database are points in the data cube. We use the term "partition" to refer to any sub-cube in the data cube. We denote any sub-cube that that is not divided by any more dimensions by "cell", meaning it's the "smallest" sub-cube. We denote the number of cells by β.

3.5 (ϵ, δ)-usefulness

We represent the utility of the released data, by considering whether it is (ϵ, δ)-useful.

Definition 31 ((ϵ, δ)-usefulness[2]). *A database mechanism $\mathcal{A}$ is (ϵ, δ)-useful for queries in class C if with probability $1 - \delta$, for every $Q \in C$, and every database D, $\mathcal{A}(D) = \hat{D}$, $|Q(\hat{D}) - Q(D)| \leq \epsilon$.*

3.6 Categorization of Aggregate Queries

Definition 32 (Distributive query [18]). *A distributive aggregate query is a function that can be computed for a given data set by partitioning the data into small subsets, computing the function of each subset, and then merging the results in order to arrive at the function's value for the original (entire) data set.*

Definition 33 (Linear distributive query). *A linear distributive query is a function with result that can be computed as a linear function of the result from each subset.*

For example, sum() can be computed by first partitioning the data, then summing up the sums of each partition. avg() can not be distributively computed but it can be computed by a function of sum() and count().

In this paper, we mainly focus on linear count queries which will be used to generate the histogram and to form a query workload to evaluate the released histogram. We will discuss later how the released histogram can be used to support other types of OLAP queries such as sum and average.

Definition 34 (count query). *Absolute count query AC and relative count RC on a multi-dimensional database D is defined to be:*

$$AC_{P(x)}(D) = \sum_{x \in D} P(x) \qquad RC_{P(x)}(D) = \frac{\sum_{x \in D} P(x)}{n}$$

where $P(x)$ returns 1 or 0 depending on the predicate.

Note that $GS_{AC} = 1$, $GS_{RC} = \frac{1}{n}$ [2].

[2] n should be the upper bound of data size, so it's constant.

3.7 PINQ

PINQ [19] is a programming interface that provides a differentially private interface to a database. It provides operators for database aggregate queries such as count (**NoisyCount**) and sum (**NoisySum**) which uses Laplace noise and the exponential mechanism to enforce differential privacy. It also provides a **Partition** operator that can partition the dataset based on the provided set of candidate keys. The **Partition** operator takes advantage of parallel composition and thus the privacy costs do not add up.

4 Multidimensional Partitioning Approach

4.1 Overview

For differentially private histogram release, a multi-dimensional histogram on a set of attributes is constructed by partitioning the data points into mutually disjoint subsets called *buckets* or partitions. The counts or frequencies in each bucket is then released. Any access to the original database is conducted through the differential privacy interface to guarantee differential privacy. The histogram can be then used to answer random count queries and other types of queries.

The partitioning strategy will largely determine the utility of the released histogram to arbitrary count queries. Each partition introduces a bounded Laplace noise or *perturbation error* by the differential privacy interface. If a query predicate covers multiple partitions, the perturbation error is aggregated. If a query predicate falls within a partition, the result has to be estimated assuming certain distribution of the data points in the partition. The dominant approach in histogram literature is making the *uniform distribution assumption*, where the frequencies of records in the bucket are assumed to be the same and equal to the average of the actual frequencies[15]. This introduces an *approximation error*.

We illustrate the errors and the impact of different partitioning strategies through an example shown in Figure 2. Consider an original database that has 4 cells, each of which has 100 data points. In the first partitioning, the 4 cells are grouped into one partition and we release a noisy count for the partition. Alternatively, the 4 cells are separated into 4 partitions, each of which contain one cell, and we release a noisy count for each of the partitions or cells. Note that the noise are independently generated for each partition. Because the sensitivity of the count query is 1 and the partitioning only requires parallel composition of differential privacy, the magnitude of the noise in the two approaches are the same. Then if we have a query, count(A), to ask how many data points are in the region A, the best estimate for the first strategy based on the uniform distribution assumption will be an approximate answer, which is $100 + Y/4$. So the query error is $Y/4$. In this case, the approximation error is 0 because the cells in the partition are indeed uniform. If not, approximation error will be introduced. In addition, the perturbation error is also amortized among the cells. For the cell-based partitioning, the query error is Y which only consists of the perturbation error.

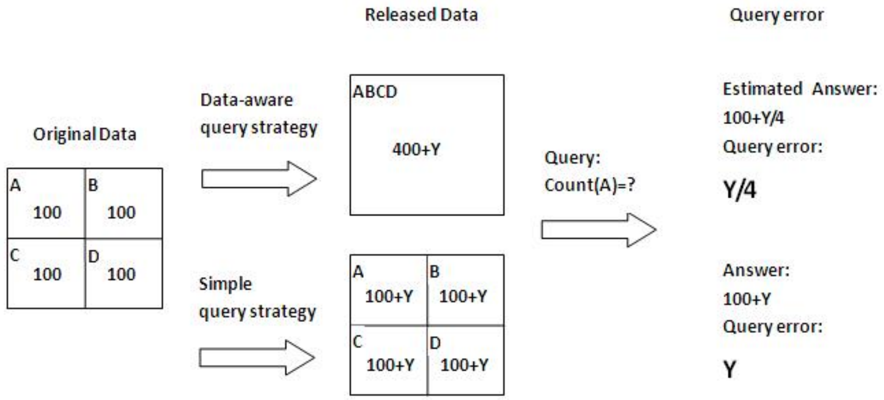


Fig. 2. Data-aware query strategy and simple query strategy

In general, a finer-grained partitioning will introduce smaller approximation errors but larger aggregated perturbation errors. Finding the right balance in this tradeoff to optimize the overall approximation of the data distribution and minimize the overall error for a random query workload is a key question. Not surprisingly, finding the optimal multi-dimensional histogram even without the privacy constraints is a challenging problem and optimal partitioning even in two dimensions is NP-hard[22]. Motivated by the above example and guided by the composition theorems, we summarize our two design goals: 1) generate uniform or close to uniform partitions so that the approximation error with the partitions is minimized, and 2) carefully and efficiently use the privacy budget to minimize the perturbation error. In this paper, we study two heuristic strategies: 1) the most fine-grained cell-based partitioning as a baseline strategy, which does not introduce approximation error but only perturbation error, and 2) a kd-tree based partitioning strategy that seeks to produce close to uniform partitions and an efficient implementation that seeks to minimize the perturbation error.

4.2 Cell-Based Algorithm

A simple strategy is to partition the data based on the domain and then release the count for each cell. The implementation is quite simple, taking advantage the **Partition** operator followed by **NoisyCount** on each partition.

Algorithm 1. Cell-based algorithm

```
Require: $\alpha$: differential privacy budget
  1. Partition the data based on all domains.
  2. release NoisyCount of each partition using privacy parameter $\alpha$
```

Theorem 41. *Algorithm 1 achieves α-differential privacy.*

Proof. Because every cell is a disjoint subset of the original database, according to theorem 32, it's α-differentially private.

Utility. We present a general theorem following a lemma that states a formal utility guarantee with cell-based partitioning for linear distributive queries.

Lemma 41. *If Y_i is the random variables i.i.d from Lap(b) with mean 0, then*

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

Theorem 42. *The released $\hat{D}$ of algorithm 1 maintain (ϵ, δ)-useful for linear distributive query $Q(D) = \sum_{i=1}^{x} Q(D_i)$, if $GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$, where x is the number of cells contained in the predicate, $x \leq \beta$, β is the number of partitioned cells of the data cube.*

Proof. By interactive mechanism of differential privacy, the returned answer $\mathcal{A}_Q(D) = Q(D) + Y$, where $Q(D)$ is the true answer of the query and Y is the Laplace noise Lap(b) where b=GS/α. we use D_i to present the data in the cells, then the returned answer of $\hat{D}$ is

$$Q(\hat{D}) = \sum_{i=1}^{x}(Q(D_i) + Y_i) = \sum_{i=1}^{x} Q(D_i) + \sum_{i=1}^{x} Y_i = Q(D) + \sum_{i=1}^{x} Y_i$$

With Lemma 41(proved in Appendix B), we have

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

If $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, then we can get

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \delta$$

So, $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, b=GS/$\alpha$, we have

$$GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$$

Corollary 41. *Algorithm 1 is (ϵ, δ)-useful for absolute count queries if $\epsilon \geq \beta ln(\beta/\delta)/\alpha$, for relative count queries if $n \geq \frac{\beta ln(\beta/\delta)}{\alpha\epsilon}$.*

4.3 K-d Tree Based Algorithm

We now present our kd-tree based partitioning strategy. A kd-tree (k-dimensional tree) is a space-partitioning data structure for organizing dat points in a k-dimensional space. A typical kd-tree construction starts from the root node which covers the entire space. At each step, a splitting dimension and a split value, typically median, from the range of the current partition on that dimension are chosen heuristically to divide the space into subspaces. The algorithm repeats until a pre-defined requirement (such as tree height or number of data points in each partition) are met. This method leads to a balanced kd-tree, in which each leaf node is about the same distance from the root, which is desired in indexing. In our setting, our main design goal is to generate uniform or close to uniform partitions so that the approximation error within the partitions is minimized. Thus we propose a uniformity based heuristic to make the decision whether to split the current partition. Concretely, we do not split a partition if it is close to uniform and split it otherwise. There are several metrics that can be used to measure the uniformity of a partition such as information entropy and variance. In our current implementation, we use a variance-like metric H defined below. If $H > \xi_1$ where ξ_1 is a threshold, then we don't split.

Definition 41. *Assuming we have an sub-cube D_0 with β cells, the average count would be*
$a_0 = \sum_{c_i \in D_0} count(c_i)/\beta$ *where c_i is each cell in D_0. The heuristic metric is defined as:*

$$H(D_0) = \sum_{c_i \in D_0} |count(c_i) - a_0|$$

A straightforward implementation of the above kd-tree strategy is similar to that used in [14]. At each step, a **NoisyCount** is requested for each value of the splitting attribute in order to determine the split value median. The process repeats until the stop criteria is reached. As each step queries the original database, the composition theorem applies which results in cumulated privacy cost. Hence the privacy budget needs to be divided among all the steps and the final step to obtain the **NoisyCount** of each partition which is an inefficient use of the privacy budget. To this end, we propose a two-step algorithm that generates the kd-tree partitions based on the histogram generated from the cell partitioning. First, we generate a synthetic database D_c based on the cell-based algorithm. Then we perform kd-tree partitioning on D_c and use the resulting partitioning keys to **Partition** the original database. We finally release **NoisyCount** for each of the partitions. Essentially, the kd-tree is based on an approximate distribution of the original data. The original database is not queried during the kd-tree construction which saves the privacy budget. Our experiments verify that the benefit of having smaller perturbation error outweighs the aggregated approximation error. Algorithm 2 is the framework; Algorithm 3 is the step 2 of Algorithm 2.

Theorem 43. *Algorithm 2 is α-differentially private.*

Proof. Step 2 and Step 5 are $\alpha/2$-differentially private. So the sequence is α-differentially private because of theorem 31.

[3] In this paper, we take the dimension which has the largest range in the current partition.

Algorithm 2. K-d tree based algorithm

```
Require: $\beta$: number of cells; $\alpha$: the overall privacy budget
  1. Partition the original database based on all domains.
  2. get NoisyCount of each partition using privacy parameter $\alpha/2$ and generate a
  synthetic dataset $D_c$.
  3. Partition $D_c$ by algorithm 3.
  4. Partition the original database based on the partition keys returned from step
  3. 5. release NoisyCount of each partition using privacy parameter $\alpha/2$
```

Algorithm 3. K-d tree partitioning

```
Require: $D_t$: input database; $\xi_0$: threshold of generating k-d tree; $\xi_1$: threshold for
  function $H$;
  1. Find a dimension of $D_t{}^3$;
  2. Find the median $m$ of the dimension;
  3.
  if $H(D_t) < \xi_1$ or $Count(D_t) > \xi_0$ then
     Divide $D_t$ into $D_{t1}$ and $D_{t2}$ by $m$.
     partition $D_{t_1}$ and $D_{t_2}$ by algorithm 3.
  end if
  return partitions
```

Utility. The utility of the algorithm is directly decided by the parameter ξ_1 and xi_0 as well as the data distribution. We resort to experiments to evaluate the utility of the algorithm.

4.4 Applications

Having presented the multidimensional partitioning approach for differentially private histogram release, we now briefly discuss the applications that the released histogram can support.

OLAP. On-line analytical processing (OLAP) is a key technology for business-intelligence applications. The computation of multidimensional aggregates, such as count(), sum(), max(), avg(), is the essence of on-line analytical processing. We discuss the applications of the above released data to common OLAP aggregate functions. We assume all queries have a predicate φ. An example form of the predicate can be φ_1 *and* $\varphi_2 \ldots$ *and* φ_m where φ_j can be a range predicate of the form $a_l \leq A_j \leq a_h$. The predicate determines a set of cells S_φ that satisfies the predicate. We denote the value of attribute A for cell i as a_i, and the perturbed count for cell i as c_i.

- Count queries are supported directly by the released data. The cell-based partitioning provides a guaranteed utility bound.
- Sum queries sum(A) for an attribute or dimension A can be computed as $\sum_{i \in S_\varphi}(a_i * c_i)$. Based on Theorem 42, the cell-based partitioning also provides a formal guarantee for sum query which is a linear distributive query.

- Average queries avg(A) for an attribute or dimension A can be computed as $\frac{\sum_{i \in S_\varphi}(A_i * c_i)}{\sum_{i \in S_\varphi}(c_i)}$. The algorithm, however, does not provide a formal utility guarantee for the average queries. We experimentally evaluate the utility for average queries in Section 5.

Learning. The released data can be also used for learning tasks such as construction of decision tree. Below we use a simple learning task of learning parity function to illustrate the idea. Given a non-zero vector $v \in \{0,1\}^{d-1}$, a data set $\mathrm{D} \in \{0,1\}^d$, let k be the kth row of the data set and $x_k^{(j)}$ is the jth attribute of x_k. Without loss of generality, we define parity function as the inner product of v and the first $d-1$ attributes of x_i modulo 2 and the result is saved to the dth attribute of x_i, that is

$$g(k, v) = \oplus_{j \leq d-1} x_k^{(j)} v^{(j)} = x_k^{(d)},$$

Without loss of generality, we assume the parity query has the form:

$$PQ_v = Pr[g(v) = 1] = \frac{|\{i \in [n] : g(k, v) = 1\}|}{|D|}$$

So if without noise, the more $Pr[g(v)]$ is near 1, the more likely hypothesis v is correct.

Our problem is to learn the parity function based on the perturbed counts with noise. [1,9] proposed a learning algorithm for parity function in the presence of random classification noise and adversarial noise. We introduce the algorithm in Appendix C.

5 Experiment

We use the CENSUS data(http://www.ipums.org) for our experiments. It has 1 million tuples and 4 attributes: Age, Education, Occupation and Income, whose domain sizes are 79, 14, 23 and 100 respectively. All experiments were run on a computer with Intel P8600($2 * 2.4$ GHz) CPU and 2GB memory.

5.1 Cell-Based Algorithm

Absolute count queries. We use the Income attribute to simulate 1D count queries. The following parameters are used for the desired level of differential privacy and utility guarantee: $\beta = 100; \alpha = 0.05; \delta = 0.05$. The run time for this experiment is 0.155 seconds.

Figure 3, 4 show the original and released distribution. The x-axis is the Income, the y-axis is the corresponding count of each income value over the complete domain. Figure 5 is the difference(Laplace noise) between the original and released distribution. We performed 100 independent releases and tested 1000 randomly generated queries for each data release and averaged the results. The average error is 62.2. We counted the times each error happens in Figure 6. The x-axis is the query error ϵ, and y-axis is the probability of each error. Figure 7 shows the actual error that is far less than the theoretical error.

5.2 Average Query

We use average query to demonstrate our method could be used for other queries which are not distributive queries. We extend the 1D count to 2D count using Age and Income as dimensions. Then we compare the difference of average(age) between original data and released data by each Income. Figure 8 shows the comparison. The blue line is the result of original data, red line is the result of perturbed data. We can see that the difference of the two sets of value differs lightly.

5.3 K-d Tree Based Algorithm

We use the Age and Income attribute to simulate 2D rectangle queries with data generated by algorithm 2. We also implemented an alternative kd-tree strategy similar to that used in [14], referred to as hierarchical kd-tree, for comparison.

Hierarchical kd-tree. The height of hierarchical kd-tree is the key for the query error of random workload. Figure 9 and 12 show the query error of absolute count and relative count. We can see that the least noise appears when the height is around 13.

Query error vs different thresholds. We analyze the different thresholds of ξ_0 and ξ_1 in our kd-tree algorithm. Figure 11 and 11 show the results. We can see that the threshold ξ_0 significantly affects the query error.

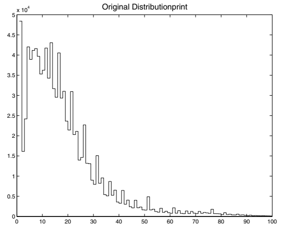


Fig. 3. Original Distribution of Income

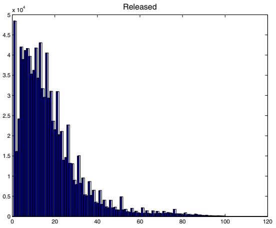


Fig. 4. Released Distribution of Income

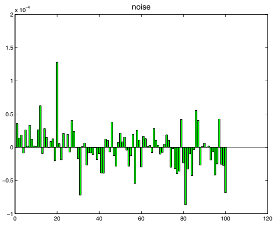


Fig. 5. Perturbation Noise

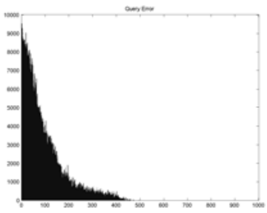


Fig. 6. Query Error of 1D Count Queries

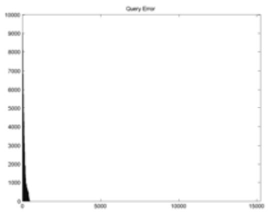


Fig. 7. Actual Error and Bound

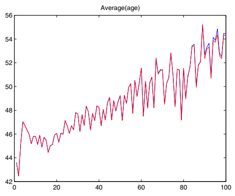


Fig. 8. Average(age) by Income

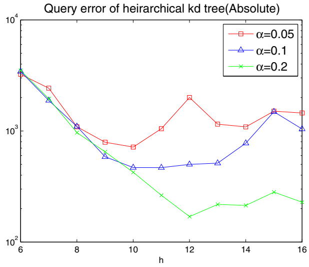


Fig. 9. Hierarchical kd-tree: Absolute count

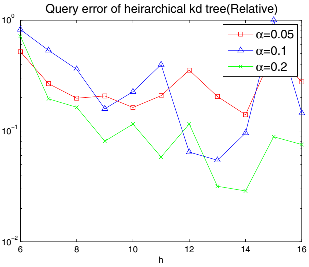


Fig. 10. Hierarchical kd-tree: Relative count

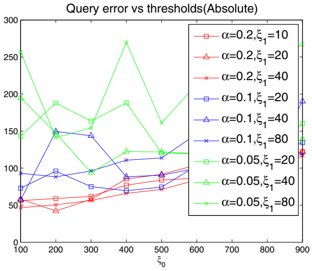


Fig. 11. Query error and thresholds: Absolute count

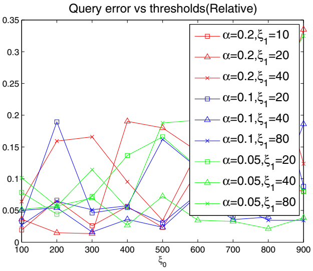


Fig. 12. Query error and thresholds: Relative count

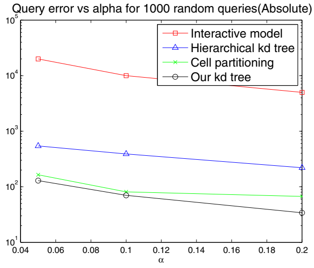


Fig. 13. Query error and α: Absolute count

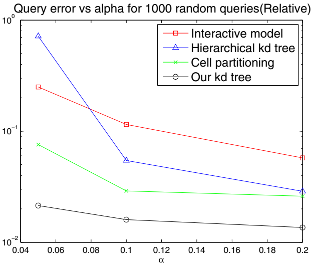


Fig. 14. Query error and α: Relative count

Query error vs alpha. We fix the height in hierarchical kd-tree algorithm and the threshold in our kd-tree algorithm to compare the query error. Figure 13 and 14 show the results. We can see that our kd tree algorithm provides best utility for random workload. Note that for relative count query, the utility outperforms other algorithm much more because for sparse data, the error of relative count query vary very much. Therefore, our kd-tree algorithm provides better results for sparse data.

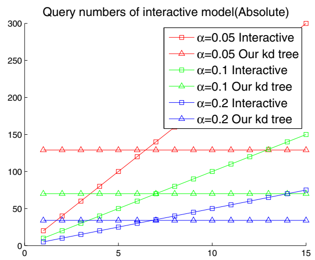


Fig. 15. Query error and number of queries: Absolute count

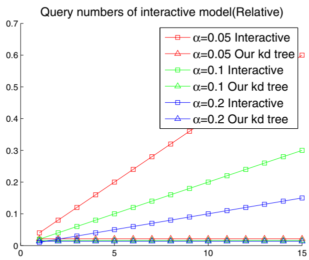


Fig. 16. Query error and number of queries: Relative count

Query error vs Number of queries in interactive model. When number of queries is small, the interactive model provides better result than our non-interactive approach. However, when the number of queries increases, the query error of random workload in interactive model may becomes larger. We can see form Figure 15 and 16 that when the number of queries are around 3 and 7, our kd-tree algorithm outperforms interactive model for absolute query and relative query.

6 Conclusions and Future Works

We have described our preliminary studies on two multidimensional partitioning algorithms for differentially private histogram release based on an interactive differential privacy mechanism. By carefully designing the partitioning strategy and allocating the usage of the privacy budget, we show that our kd-tree based algorithm outperforms the baseline cell-based algorithm and an alternative kd-tree based algorithm for a random workload of counting queries. We also show that the result is significantly better than answering the queries directly through the interactive mechanism which is oblivious of the correlations of the queries and the data. We are interested in mapping the algorithms to the matrix mechanism framework [17] and conduct a formal error analysis. In addition, we are interested in exploring other spatial indexing techniques and integrating them with the consistency check techniques [13]. Finally, we plan to investigate in algorithms that are both data- and workload-aware to boost the accuracy for a specific workload.

Acknowledgements

The research is supported in part by an International Science Cooperation Fund of Shenzhen, China, GJ200807210023A and a Career Enhancement Fellowship by the Woodrow Wilson Foundation. The authors would also like to thank the chairs for their patience and support and the anonymous reviewers for their insightful comments that helped improve the paper.

References

1. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM 50(4), 506–519 (2003)
2. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: STOC 2008: Proceedings of the 14th Annual ACM International Symposium on Theory of Computing, Victoria, Canada, May 17-20, pp. 609–617 (2008)
3. Dwork, C.: A firm foundation for private data analysis (to appear)
4. 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)
5. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
6. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd Theory of Cryptography Conference, pp. 265–284 (2006)
7. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC 2009: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 381–390. ACM, New York (2009)
8. Feldman, D., Fiat, A., Kaplan, H., Nissim, K.: Private coresets. In: STOC 2009: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, Bethesda, MD, May 31-June 2, pp. 361–370 (2009)
9. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: New results for learning noisy parities and halfspaces. In: Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science, Berkeley, CA, October 21-24, pp. 563–572 (2006)
10. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey on recent developments. ACM Computing Surveys 42(4) (2010)
11. Götz, M., Machanavajjhala, A., Wang, G., Xiao, X., Gehrke, J.: Privacy in search logs. CoRR, abs/0904.0682 (2009)
12. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: STOC 2010: Proceedings of the 42nd ACM symposium on Theory of computing, pp. 705–714. ACM, New York (2010)
13. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially-private queries through consistency. In: VLDB (2010)
14. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: 13th International Conference on Extending Database Technology, EDBT (2010)
15. Ioannidis, Y.: The history of histograms (abridged). In: Proc. of VLDB Conference (2003)
16. Korolova, A., Kenthapadi, K., Mishra, N., Ntoulas, A.: Releasing search queries and clicks privately. In: WWW (2009)
17. Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A.: Optimizing linear counting queries under differential privacy. In: PODS 2010: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data, pp. 123–134. ACM, New York (2010)
18. Kamber, M., Han, J.W.: Data mining: concepts and techniques, 2nd edn. Morgan Kaufman, San Francisco (2006)

19. McSherry: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: SIGMOD 2009: Proceedings of the 35th SIGMOD international conference on Management of data, pp. 19–30. ACM, New York (2009)
20. McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the net. In: KDD 2009: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 627–636. ACM, New York (2009)
21. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science, Providence, RI, October 20-23, pp. 94–103 (2007)
22. Muthukrishnan, S., Poosala, V., Suel, T.: On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In: ICDT, pp. 236–256 (1999)
23. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: STOC 2010, pp. 765–774 (2010)
24. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. CoRR abs/0909.5530 (2009)

A Proof of Theorem 33

Proof.

$$Pr[D|A_1, A_2] = \frac{Pr[A_1, A_2|D]}{Pr[A_1, A_2]} = \frac{Pr[A_2|D] \cdot Pr[A_1|D] \cdot Pr[D]}{Pr[A_1, A_2]}$$
$$= \frac{Pr[A_2|D] \cdot Pr[D|A_1] \cdot Pr[A_1]}{Pr[A_2] \cdot Pr[A_1]} = \frac{Pr[A_2|D] \cdot Pr[D|A_1]}{Pr[A_2]}$$

Let's recap the model of differential privacy. Assume the attacker knows all the record except the ith record d_i. Let q_l be the lth query, $Q(q_l)$ be the true answer of q_l, a_l be the perturbed answer of q_l. Let x be the range of d, D be the event that $d_i = x_1$, P_l be the posterior probability of $Pr[D|a_1, a_2, \cdots a_l]$. Let $A_1 = \{a_1, a_2, \cdots, a_{l-1}\}$, $A_2 = \{a_l\}$, then

$$P_l = Pr[D|A_1, A_2] = P_{l-1} \cdot \frac{Pr[A_2|D]}{Pr[A_2]}$$
$$= P_{l-1} \cdot \frac{Pr[a_l|d_i = x_1]}{\sum_{x_j \in x} Pr[a_l|d_i = x_j] \cdot Pr[d_i = x_j]}$$

If we adopt Laplace noise to achieve differential privacy, then

$$\frac{p_l}{p_{l-1}} = \frac{\frac{1}{2b} exp(-|a_l - Q(q_l, x_1)|/b)}{\sum_{x_j \in x} \frac{1}{2b} exp(-|a_l - Q(q_l, x_j)|/b) \cdot Pr[d_i = x_j]}$$
$$\leq \frac{1}{\sum_{x_j \in x} exp(-|Q(q_l, x_j) - Q(q_l, x_1)|/b) \cdot Pr[d_i = x_j]}$$

Recall the definition of GS, then

$$|Q(q_l, x_j) - Q(q_l, x_1)| \leq GS$$

$$\frac{P_l}{P_{l-1}} \leq \frac{1}{\sum_{x_j \in x} exp(-GS/b) \cdot Pr[d_i = x_j]}$$

Because $b = GS/\alpha$, then $P_l/P_{l-1} \leq e^{\alpha}$. So, $Pr[V = d] \leq P_0 * exp(l\alpha)$.

B Proof of Lemma 41

Lemma B1. *If Y_i is the random variables i.i.d from Lap(b) with mean 0, then*

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

Proof (of Lemma 41). We assume each $|Y_i| \leq \epsilon_1$ where $\epsilon_1 = \epsilon/\beta$. Otherwise we call $|Y_i| > \epsilon_1$ a FAILURE. If no FAILURE happens,we have

$$\sum_{i=1}^{\beta} |Y_i| \leq \beta \cdot \epsilon_1 = \epsilon$$

If a FAILURE happens, then we have $|Lap(b)| > \epsilon_1$, which means

$$\text{Pr[a FAILURE]=}2\int_{\epsilon_1}^{\infty} \tfrac{1}{2b} exp(-\tfrac{x}{b}) = e^{-\epsilon_1/b}$$

For each Y_i, Pr[no FAILURE happens]=1-Pr[FAILURE happens] and each Y_i is i.i.d distributed, we have

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq (1 - e^{-\epsilon_1/b})^{\beta}$$

Let $F(x) = (1-x)^{\beta} + \beta x - 1$, then $F(0) = 0$. $F'(x) = -\beta(1-x)^{\beta-1} + \beta = \beta(1-(1-x)^{\beta-1}) > 0$ when $0 < x < 1$. Note that $0 < e^{-\epsilon_1/b} < 1$, so $F(e^{-\epsilon_1/b}) > 0$. We get $(1 - e^{-\epsilon_1/b})^{\beta} > 1 - \beta \cdot e^{-\epsilon_1/b}$. Because $\epsilon_1 = \epsilon/\beta$,

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

C Learning Algorithm of Parity

We assume the data is uniformly distributed and the noise added is classification noise. For the cell that satisfies the parity vector r, even if we add Laplace noise to the cell by the mechanism of differential privacy, it is not noise for the parity query. We use η_1 to present the original noise rate. Our algorithm add η_2 noise rate to the data. The total noise rate η in the released data is $\eta_1 + \eta_2$. Therefore, $\eta_2 \leq \epsilon$, $\eta \leq \eta_1 + \epsilon$.

Theorem C1 ([1]). *the length-d parity problem, for noise rate η equal to any constant less than 1/2, can be solved with number of samples and total computation-time $2^{O(d/logd)}$.*

Therefore, if $\eta < 1/2$, the $\hat{D}$ is learnable. $\eta \leq \eta_1 + \epsilon$ and $n \geq \frac{ln\frac{\beta}{\delta}}{\alpha\epsilon}\beta$, so if $n \geq \frac{ln\frac{\beta}{\delta}}{\alpha(1/2-\eta_1)}\beta$, the $\hat{D}$ is learnable.

Lemma C1 ([1]). *Take s samples from a database with parity vector v and noise rate η, then*

$$Pr[\oplus_{i=1}^{s} x_i^{(d)} = \oplus_{i=1}^{s} g(i, v)] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s.$$

note that each $x_i^{(d)}$ may not be the correct answer of g(i,v) because of noise.

We give the algorithm as follows:

```
Learning steps for parity vector:
For every target bit $j$ of vector $v$, $j = 1 : d-1$
1. Draw $s$ samples from the released data where $s > d-1$ to avoid linear dependence.
2. for $k = 1 : d-1$,
    choose a record $x_i$ where $x_i^{(k)} = 1$, remove $x_i$ from the samples and XOR the rest samples with $x_i$, $x = x \oplus x_i$.
3. if in left samples doesn't exists $x^{(j)} = 1$, then goto 1; else after the elimination of step 2, we discard those samples which become 0, then randomly draw a sample t from the left samples.
    $$Pr[v^{(j)} = 1] = Pr[t^{(j)} = 1] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$$
```

with the lemma described above, we know that every $x^{(j)}$ has probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$ to be correct. Therefore, with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$, we output the correct bit of vector v.

D Expected Error of Laplace Noise

Theorem D1 (Expected error of laplace noise). *The expected error of laplace noise with parameter α is GS/α.*

Proof.

$$err = |Q(\hat{D}) - Q(D)| = |Lap(GS/\alpha)| \tag{1}$$

$$b = GS/\alpha \tag{2}$$

$$E(err) = \int_{-\infty}^{+\infty} err * (\frac{1}{2b} exp(-\frac{|x|}{b})) \tag{3}$$

solve the equations, we get $E(err) = GS/\alpha$.

Author Index