Bulletin of the Technical Committee on

Data Engineering

March 2004 Vol. 27 No. 1 **IEEE Computer Society**

Letters

Special Issue on Data Privacy and Security

Conference and Journal Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

The Data Engineering Bulletin web page is http://www.research.microsoft.com/research/db/debull.

Letter from the Editor-in-Chief

The Current Issue

The needs for our systems to provide security and privacy have escalated dramatically as a result of what might be called the ”internet age”. Once data is on-line, it becomes a target for both over-zealous exploitation and malicious purpose. While database systems have had security functionality since their birth, the new requirements of the internet age require much more. This is by no means captured completely by access rights or strong authentication, though these surely provide some help.

There are a number of really hard problems in this area. One security problem is that in a widely dispersed system, security is hostage to the vulnerabilities of the weakest links. This is similarly true for privacy. So ”end-to-end” solutions become a requirement. Privacy can be even more subtle than security. The issue here is permitting some useful and hopefully innocuous access while preventing the escape and misuse of private information that might be accessible via determined effort. The current issue pursues these and a number of other very hard problems and gives some insights into how researchers are attacking them.

So this issue is both timely, given our hopes for further exploiting the web as an information resource, and challenging in the difficulties being faced. I want to thank Johannes Gehrke for both suggesting the topic and for editting this special issue of the Bulletin on ”Security and Privacy”. These problems will be with us for a long while, and are very important to our industry. So surely they represent a great research opportunity for our field. The current issue gives a snapshot of some of the approaches currently being pursued. It makes for very interesting reading (and thinking).

David Lomet
Microsoft Corporation

TCDE Election Result

New TCDE Chair for 2004-2005

The election for Chair of the IEEE Computer Society Technical Committee on Data Engineering (TCDE) for the period January 2004 to December 2005 has concluded. We are pleased to announce that Professor Erich Neuhold, Darmstadt University of Technology (neuhold@ipsi.fhg.de) has been elected to a second term as Chair of the TCDE. Erich’s biography and a position statement can be found in the December 2003 issue of the DE Bulletin.

Paul Larson, Masaru Kitsuregawa,Betty Salzberg
Nominating Committee

Letter from the Special Issue Editor

The exponential growth in the amount of digital data has resulted in the creation of databases of unprecedented scale. At the same time concerns about data security and privacy have become more severe, with evidence of a wealth of new work inside and outside the database research community in this area.

This issue gives an overview of selected ongoing work on data privacy and security. The first article by Günter Karjoth, Matthias Schunter, and Michael Waidner describes an architecture for managing and enforcing privacy policies within an enterprise. The second article by Dan Boneh, Joan Feigenbaum, Avi Silberschatz, and Rebecca Wright gives an overview of the PORTIA project which encompasses research on privacy-preserving data mining, data access control and access policy enforcements, and identity identification. While the first two articles concentrate on limiting data release to enforce data privacy, the third article by Mikhail Atallah, Sunil Prabhakar, Keith Frikken, and Radu Sion looks at limiting data release from the direction of protecting the intellectual property rights of digital content owners.

We switch from privacy to security with an article by Michael Gertz, April Kwong, Charles Martel, and Glen Nuckolls who describe how a database management service provider could prove to a client that the service actually computed the correct answer to a client's query and did not cheat maliciously. The next article by Gerome Miklau and Dan Suciu describes techniques for ensuring confidentiality and integrity for secure data exchange over the Internet. We conclude this issue with an article by Arnon Rosenthal and Edward Sciore on new ideas for data access control that facilitates collaboration between adminitrators.

Herbert Hoover once said that "There are only two occasions when Americans respect privacy, especially in Presidents. Those are prayer and fishing." I hope that this collection of articles stimulates discussions on more than Hoover's quote, and that it gives a taste of the excitement, diversity, and importance of research in the areas of data privacy and security.

Johannes Gehrke
Department of Computer Science
Cornell University
Ithaca, NY 14850

Privacy-enabled Management of Customer Data

Günter Karjoth, Matthias Schunter, and Michael Waidner
IBM Research, Zurich Research Laboratory,
Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{gka,mts,wmi}@zurich.ibm.com

Abstract

Customers are required to trust selected enterprises to protect their personal information. The large amounts of personal data that these enterprises store constitutes a risk. Accidental disclosure of personal data or misuse by employees often leads to bad publicity and can result in substantial liabilities. The first step towards proper privacy management is to inform the customers how their data is used and to whom it may be disclosed. This can be declared in a privacy notice that can be formalized using the W3C Platform for Privacy Preferences (P3P). Unfortunately, P3P does not provide means to enforce these privacy promises throughout and across multiple enterprises. This article describes technology for privacy-enabled management and exchange of customer data. Using a comprehensive privacy-specific access control language, we can express restrictions on the access to personal data, possibly shared between multiple enterprises. We separate the enterprise-specific deployment policy from the privacy policy that covers the complete life cycle of collected data. In addition, we introduce a viable separation of duty between the three "administrators" of a privacy system: The privacy officer designs and deploys privacy policies, the security officer designs access control policies, and the customers can give consent while selecting opt-in and opt-out choices.

1 Introduction

There is no viable technology than enables consumers to enforce proper use of their personal information throughout an enterprise[1]. As a consequence, customers are required to trust an enterprise once they disclose their personal data.

Many enterprises are aware of this risk and of the market share they might loose if they violate the trust of their customers. As a consequence enterprises publish privacy statements that promise fair information practices. Written in natural language or formalized using P3P [6], they merely constitute privacy promises and are not necessarily backed up by technological means.

In this article, we describe technology that enables an enterprise to enforce the privacy promises made to its customers. The goal is to enable the management of an enterprise (including the Chief Privacy Officer) to prevent misuse or inappropriate disclosure of personal data by regular employees. It solves different aspects of privacy enforcement:

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1]Note that this goal of protecting data (provided by a customer) against its holder (enterprise) would be similar to the problem of Digital Rights Enforcement.

- Enterprises store a variety of personally identifiable information (PII or *personal data* for short). Enterprises may not know what privacy statement applies to what piece of data.
- Enterprises may not know the consent a customer has given nor the legal regulations that apply to a specific customer record.
- Enterprises exchange customer data. The privacy statement fixes usage restrictions at the point of collection. It is unclear how to communicate these restrictions once personal data is disclosed to another enterprise.

Whenever an enterprise collects, stores, or processes personal information, our concepts can be used to ensure that the data flows and usage practices of an enterprise comply with the privacy statement of that enterprise.[2] We cover the following areas:

- *Formalized Privacy Policies:* The policy language "Enterprise Privacy Authorization Language (EPAL)" [2] enables an enterprise to formalize a privacy policy into a machine-readable language that can be enforced automatically.
- *Formalized Policy Options:* An EPAL privacy policy can identify opt-in as well as opt-out choices or options that depend on the collected data (e.g., whether the given data pertains to a child). These options enable a company to use a limited number of policies while still providing freedom of choice to its customers.
- *Policy Enforcement:* Given collected personal data and its policy, the policy needs to be enforced. Policy enforcement covers several cooperating enterprises if personal data is exchanged among them. The core technology is a scheme for privacy-enabling access control that allows only actions that are authorized by the applicable privacy policy. Besides granting or denying access, privacy obligations have to be enforced as well (such as "we delete collected data if consent is not given within 15 days").
- *Compliance Audit:* A challenge of a privacy audit is that privacy policies are intertwined with applications that handle personal data. By separating privacy policies from applications, privacy audits are simplified since they can focus on the privacy policies and usage logs that can be accessed via a privacy management system.

Note that this is only the technical core of privacy-enabled customer data management. Another important building block is to provide additional business processes that implement customer privacy services. Customers should be enabled to inspect and update the data and usage logs stored about them. In addition, an enterprise may offer the option to delete the personal data. Ideally, customers should retain maximum control over their data. Once privacy-management has been implemented, it needs to be audited by external parties that are trusted by the customers. Together with resulting privacy seals, this can increase the trust of the consumers.

For related work, we need to refer to our earlier work that survey different aspects of privacy management [1, 3, 5].

2 Application Model and Prerequisites

In our application model, an enterprise runs *legacy applications* that use collected data. Each application can perform certain *tasks*. For example, a "customer relationship management system (CRM)" application may

[2] Note that our scheme only protects against systematic privacy violations within the system. It cannot prevent misuse by an employee with legitimate access or misuse that falls outside the boundaries of the system. An example is that it cannot prevent an employee from copying information from the screen to a note pad.

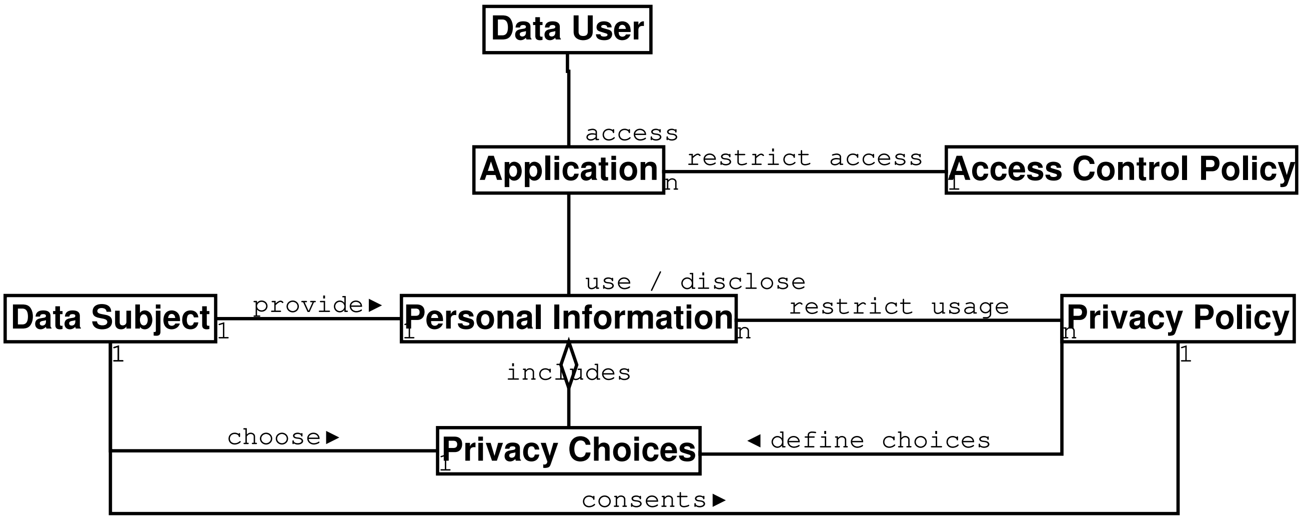

Figure 1: Separation of duties for privacy authorization.

perform the tasks “create new customer record” or “update existing customer record”. Enterprise privacy policies reflect the authorized flow and usage of personal information within an enterprise. As a consequence, the flows and usages have to be identified in order to build a privacy enforcement system:

- A *business-process model* for the collection and use of customer data defines the scope of the data management system. The business-process model identifies the players that use collected data, the data they use, and how and for what purposes they use the data. The business model is formalized as the declaration of data, players and operations of an EPAL privacy policy.

- A collection of *informal privacy policies* that govern the use of personal data in the business processes. They can be structured as bilateral privacy agreements that describe how data that is sent from one player to another may be used. Informal privacy policies are formalized as EPAL privacy policy rules.

3 Defining Policies based on Separation of Duties

Privacy and security authorization in an enterprise involves at least four types of players. The *data subjects* are the players about whom personal data is collected. The most common data subjects are the customers of an enterprise. Other data subjects are employees or customers of cooperating enterprises. The next players are the *data users* within an enterprise who use collected data by executing *tasks* of *applications*. The other two players are the *privacy officer* (PO), who is responsible for privacy services, and the *security officer* (SO), who is responsible for security services.

Our privacy management architecture introduces the following intermediate abstractions and the corresponding policies (see Figure 1) in order to separate the duties of these players:

Privacy Policy The PO defines a *privacy policy*. A privacy policy describes what *operations* for which *purpose* by which *data user* can be performed on each *PII type*. For example, the “marketing department” may be allowed to “read” the PII type “contact data” for purpose “e-mail marketing”. In addition, a privacy policy may define opt-in and opt-out choices for the data subjects as well as certain privacy obligations such as “delete my data after 30 days unless parental consent has been given”.

The privacy policy should be enterprise- and application-independent. Enterprise-internals such as the role structure should not be used in order to enable exchange of policy-protected data between cooperating enterprises.

Deployment The PO and the SO define a *deployment policy* that maps legacy applications and their tasks onto the privacy-specific terminology used by the privacy policies. This mapping is specific to each enterprise. For example, whereas one enterprise maps a CRM system performing “product notification” as well as a printer for mass-mailings onto the action “read” for purpose “marketing”, another enterprise, which uses a legacy application instead of the off-the-shelf CRM system, maps this legacy application onto “read” for “marketing”.

The PO defines an *obligation mapping* that translates application-independent obligations of the privacy policy (such as “delete”) into specific implementations. For example, a delete may be translated into an “unsubscribe” of the mailing list.

Access Control Policy The SO defines an *access control policy* that defines the roles and users within an enterprise. In addition, it defines which users or roles can execute which tasks of which applications.

Data Collection Personal data is collected at collection points. Each collection point has a set of fields, and fields have a type (e.g., string) as well as a PII type (e.g., “medical record”, “address data”, or “order data”). A form groups personal data and associates this data with its data subject. A “customer data” collection point, for example, may collect the fields “name”, “street”, and “town”. Each collection point has a default policy that is assigned to data collected via this collection point. The system stores the data including opt-in or opt-out choices as well as a reference to the collection point that fixes the PII types and privacy policy.

4 Collecting Personal Data and Consent

When personal data is collected, it is core to add privacy management information that needs to be consented by the data subject. This main elements are the applicable privacy policy as well as the opt-in and opt-out choices of each individual data subject.

The collection catalog identifies the data and choices that are collected as well as the privacy policy and the PII types of the collected data. At a given collection point, the data subject enters its data in the fields of the given form. The collected data includes the fields and PII types of the entered data as well as a default policy. The data subject may then choose opt-in and opt-out choices defined by the policy. By submitting the form after reading the privacy statement, the data subject consents to the policy with respect to the selected choices. The choices are added to the form and the content of the form is stored.

Note that an enterprise privacy policy that can model access down to the employee level us usually too complex for end-users. As a consequence, it is advisable to present a coarser-grained privacy policy to the customer (either as text or P3P) and to define an EPAL policy for internal enforcement.

An important aspect is the management of the data subject’s consent on a per-person and a per-record basis. This must be guided by the *sticky policy paradigm*: When submitting data to an enterprise, the user consents to the applicable policy and to the selected opt-in and opt-out choices. The form then associates the opt-in and opt-out choices as well as the consented policy with the collected data. This holds even if the data is disclosed to another enterprise. Note that policy management on a per-user basis is useful if consent and different sources are issues to be considered. Examples are managing data of different policy versions (e.g., due to different collection times), different user roles (e.g., paying users vs. users funded by advertising), or users from different jurisdictions (e.g., Europe and US).

5 Enforcing Privacy

At collection, the collected data has been associated with the privacy policy, and the selected privacy choices. This information is used to decide whether an access by an application shall be granted. Authorization is granted in two levels. Whereas access control focuses on restricting the access of employees to enterprise applications, privacy control restricts the access of applications to collected data.

Access Control: An employee acting as a data user with certain roles requests permission to perform a task of an application. The access control policy is used to verify that the data user with the given roles is in fact allowed to perform the requested task. If this is the case, the task is executed. This access control system is independent of the privacy authorization.

Privacy Control: Once a running task of a corresponding application has requested access to certain fields of collected data, the privacy enforcement system retrieves the form and uses it to allow or deny the given request as follows:

1. The request identifies the task of an application as well as the fields to be accessed.
2. The deployment maps the task onto a privacy-relevant operation and a purpose.
3. The collection point identifies the PII types of the requested fields.
4. The privacy policy and the data subject's choices are used to decide whether the operation for this purpose is allowed on the given PII types.
5. If the operation is denied, the access for the given task on the given fields is rejected.
6. If the operation is allowed and the privacy policy specifies a privacy obligation, the obligation mapping maps the obligation to a task of an application.[3]
7. If the operation is allowed, the task can be executed on the requested fields.

6 Components for Enterprise Privacy Enforcement

The authorization procedure described in Section 5 can be implemented by the privacy enforcement components depicted in Figure 2. The components interact as follows to decide whether a task executed by a legacy application is allowed to access a protected resource:

1. A *legacy application* tries to execute a task on a protected resource.
2. A resource-specific *resource monitor* shields the resource and captures the request of a certain task for certain fields. For each task, it asks the privacy management system for authorization.
3. The resource-independent *privacy management system* obtains an authorization query identifying the fields to be accessed by a certain task of a certain application. It performs steps 1 to 3 of the authorization procedure in Section 5 to deploy the authorization query.

[3]Applications are responsible for managing their data. As a consequence, they are required to implement tasks that correspond to obligations in the privacy policy.

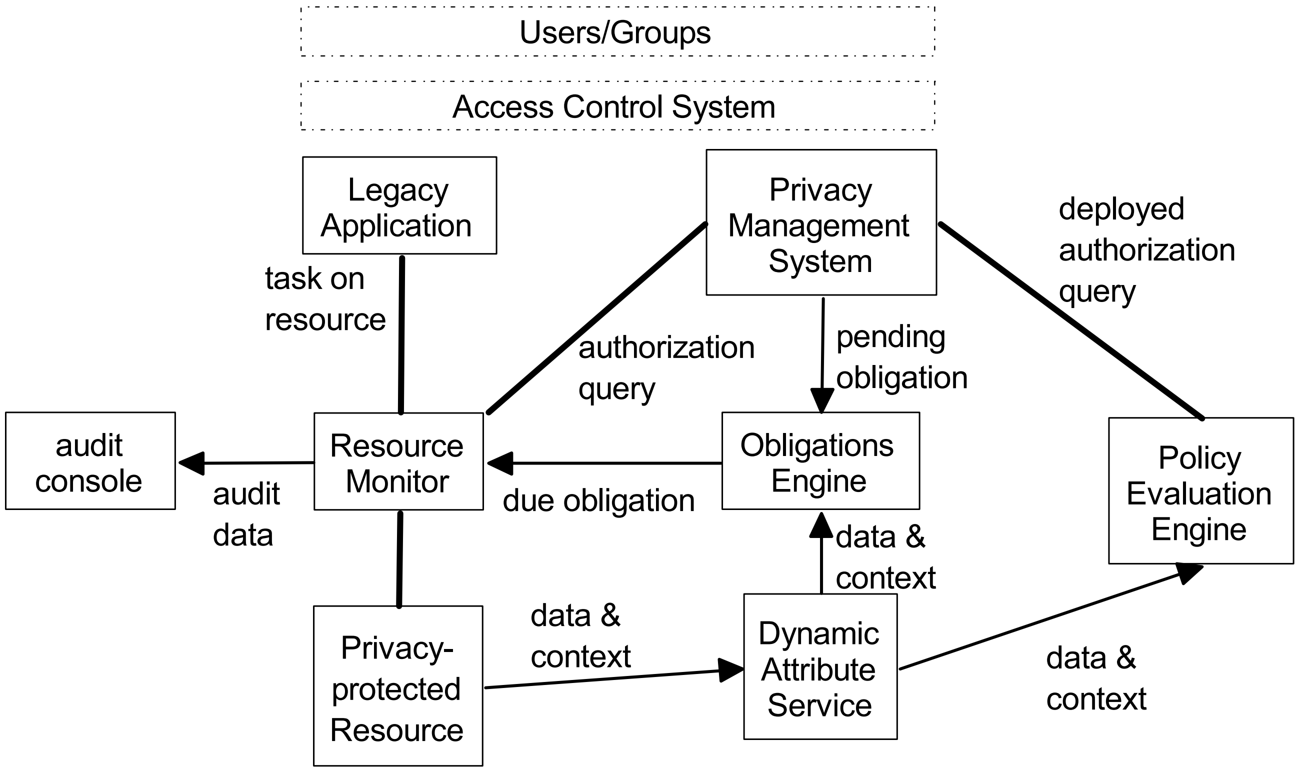

Figure 2: Architecture for privacy enforcement

4. The *policy evaluation engine* performs step 4 of the authorization procedure. The policy evaluation engine needs context and data to evaluate conditions. The resource monitor abstracts from resource and storage details by using a dynamic attribute service that provides values for data and context variables on request.

   The policy evaluation engine returns the decision as well as any resulting obligations to the privacy management system.

5. The *privacy management system* returns the decision of the policy evaluation engine to the resource monitor. If obligations were returned, the applications are mapped onto tasks (step 6) and sent to the obligations engine.

6. The *resource monitor* performs the tasks if it has been authorized. If not, the task is denied. In addition, the resource monitor sends log data to the audit monitor.

7. The resource-independent *obligations engine* stores all pending obligations. It evaluates the associated conditions based on values obtained from the dynamic attribute service. When a cancel-condition becomes valid, the obligation is removed. When a start-condition becomes valid, the obligation is sent to the resource monitor for execution.

7 Conclusion

We have described a comprehensive solution enterprise privacy management. EPAL enables an enterprise to formalize its privacy policy in an application-independent way. The deployment scheme enables enforcement of this common privacy policy for a variety of legacy systems. The viable separation of duties between the privacy officer and the security administrator enables secure and efficient management in practice. The intuitive consent-management paradigm enables customers to retain greater control over their personal data.

Our methodology enables an enterprise to protect personal data against misuse or unauthorized disclosure. It does not try to protect data if the enterprise systems or administrator are not trusted. Therefore, it merely augments a privacy-aware design of enterprise services that minimizes the data collected. In the desirable (but unlikely) scenario where an enterprise can offer its services without collecting personal data, our privacy management methodology would be rendered obsolete.

To correctly specify privacy rights and obligations that are being promised by privacy statements and mandated by a number of legislatures, the privacy officer must be able to reconcile easily what should be authorized with what is actually authorized. Therefore, we have developed a formal model for authorization management and access control in privacy protecting systems [4].

Acknowledgments

We thank Paul Ashley, Michael Backes, Kathy Bohrer, Nigel Brown, Jan Camenisch, Satoshi Hada, Calvin Powers and Els Van Herreweghen for valuable discussions and constructive comments.

References

[1] Michael Backes, Birgit Pfitzmann, Matthias Schunter: A Toolkit for Managing Enterprise Privacy Policies; 8th European Symposium on Research in Computer Security (ESORICS 2003), LNCS 2808, Springer-Verlag, Berlin 2003, 162-180.

[2] IBM: Enterprise Privacy Authorization Language (EPAL); Submission request to W3C, http://www.w3.org/Submission/EPAL/, November 2003.

[3] S. Fischer-Hübner: *IT-Security and Privacy*. Lecture Notes in Computer Science 1958, Springer-Verlag, 2001.

[4] G. Karjoth and M. Schunter. A Privacy Policy Model for Enterprises. In *15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002.

[5] Günter Karjoth, Matthias Schunter, Michael Waidner: The Platform for Enterprise Privacy Practices - Privacy-enabled Management of Customer Data; 2nd Workshop on Privacy Enhancing Technologies (PET 2002), LNCS 2482, Springer-Verlag, Berlin 2003, 69-84.

[6] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, W3C Recommendation, 16 April 2002 (from www.w3.org/TR/2002/REC-P3P-20020416/).

PORTIA: Privacy, Obligations, and Rights in Technologies of Information Assessment

Dan Boneh
Computer Science Department
Stanford University
Stanford, CA 94350 USA
dabo@cs.stanford.edu

Joan Feigenbaum and Avi Silberschatz
Computer Science Department
Yale University
New Haven, CT 06520 USA
{jf,avi}@cs.yale.edu

Rebecca N. Wright
Computer Science Department
Stevens Institute of Technology
Hoboken, NJ 07030 USA
rwright@cs.stevens-tech.edu

Abstract

Increasing use of computers and networks in business, government, recreation, and almost all aspects of daily life has led to a proliferation of sensitive data about people and organizations. Without proper precautions, these sensitive data can be misused, misinterpreted, or mismanaged. The PORTIA project aims to develop a comprehensive, end-to-end technological infrastructure for handling sensitive data over the entire course of their lifetime.

1 Introduction

Increasing use of computers and networks in business, government, recreation, and almost all aspects of daily life has led to a proliferation of sensitive data about people and organizations. By *sensitive data*, we mean data that, if used improperly, can harm data subjects, data owners, data users, or other relevant parties. These data are stored by a multiplicity of entities, ranging from individuals to small businesses to large government agencies, and concern about the ownership, control, privacy, and accuracy of these data has become a top priority in technical, academic, business, and political circles. Social trends ranging from a fluid and unpredictable business climate (that leads to unanticipated uses and exchanges of sensitive data) to a homeland-security-focused political climate (that leads to increased use of electronic surveillance technologies) make it likely that the creation and collection of massive amounts of sensitive data and the attendant nervousness about whether they are being handled properly will both increase as time goes on. Already, large population data banks store information that many are uncomfortable with [18]. The proposed "Total Information Awareness" initiative [32] caused widespread concern about potential erosion of data subjects' rights, and thus the high-tech sector anticipates

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

growing demand for surveillance technology with some built-in privacy protections [24]. Clearly, there is an urgent need for appropriate technology and policy for sensitive-data handling.

One often hears that individuals must give up some privacy or convenience so that American society as a whole can benefit from greater security. This vague and unproven assertion is used to justify the growing number of venues in which one is asked to "provide a government-issued photo ID," the thriving background-check industry, the fingerprinting of foreign visitors to the US, and the ubiquity of surveillance cameras in public places. Yet, there is no official national-ID system with provable security properties, publicly available data feeds used in low-cost background checks are error-prone, some foreign visitors are exempt from the fingerprinting requirement, and there are neither efficient procedures for mining massive sets of images nor commonly agreed-upon social norms for the appropriate use of surveillance data. Without attention to these issues, one can imagine a world in which the misuse of sensitive data continues to grow more prevalent, individuals' civil rights are routinely violated for the sake of "security measures" that may or may not actually provide the desired security, and poor data quality results in inconvenience and worse.

Some basic technological tools exist for finding critical information and patterns in a sea of disparate data feeds, for storing huge amounts of data and accessing it efficiently for certain mission-critical purposes, and for hiding or safeguarding some private information while still accomplishing tasks such as web-shopping on a large-scale public network. Yet, there is no comprehensive, end-to-end technological and public policy infrastructure for handling sensitive data over the entire course of their lifetime.

The PORTIA project [26] is exploring the design and development of such an infrastructure. Rejecting the overly simplistic "security vs. privacy" tradeoff for sensitive data, PORTIA addresses the need for deeper understanding of both the rights and the responsibilities of a number of parties, including the data subjects, the data owners, and the data users. The difficult, but necessary, goal is to balance apparently conflicting objectives such as privacy, anonymity, authenticity and integrity, appropriate-use policies, and auditability.

In this paper, we provide brief overviews of three major themes of the PORTIA project: privacy-preserving data mining, database-policy enforcement tools, and identity theft and identity protection.

Privacy-preserving data mining seeks to satisfy the desires to disclose or discover some information while protecting the privacy of other information. For example, is it possible to satisfy law-enforcement requirements for extensive mining of diverse databases in a way that does not compromise individuals' rights? We describe our vision for privacy-preserving data mining in Section 2.

Database-policy enforcement tools can allow specification and enforcement of distributed access-control policies and integrate them with existing database systems. An important goal is to control the exposure of *derived sensitive data* that can be deduced by aggregating information from multiple agents. We describe our goals for database-policy enforcement tools in Section 3.

Identity theft is the fastest growing crime in the US [30]. Often identity theft is done by impersonating a user's digital identity on a web service such as e-Bay. Can one develop techniques that make such impersonation harder to accomplish? Challenges include preventing web-spoofing attacks designed for stealing identities, defending against fraudulent spam email that targets naive users, and protecting user identities by keeping them private. We describe our work on preventing identity theft and protecting identity privacy in Section 4.

2 Privacy-preserving Data Mining

The lives of people and organizations typically involve contacts with multiple entities, including individuals, businesses, government bodies, and law-enforcement agencies. Traces of these contacts are left in the form of electronic data records, collected for *e.g.*, billing and service in the case of commercial parties or census and taxation in case of the government. Consistent advances in databases and data mining, storage, and networking technologies, as well as a rapid growth in online data collection, has led to increasing concerns about privacy protection for individual and institutional data.

Government and commercial organizations already store, manage, and mine their information. This is the basis for the database industry that has matured over the past three decades, supported by a formal theory of data management and robust commercial systems that implement it. As concern for homeland security has grown, data-mining needs have become more distributed and more urgent. Privacy-preserving data mining can act as an enabling technology by respecting the needs of individuals, corporations, and government agencies to protect certain information, while allowing the disclosure or discovery of other information. Adding privacy protection to the list of requirements for data-management systems is desirable for at least three reasons:

- **Protection of innocent bystanders**: One should not sacrifice the privacy of individuals beyond what is necessary. For example, answering the security-relevant question of whether a *targeted* person has been at a particular location should not require surveillance systems to reveal the identities of *everyone* who has been at that location.

- **Protection of sensitive information**: Pairs of organizations, *e.g.*, airlines and law-enforcement agencies or hospitals and the Centers for Disease Control, should be able to find common elements of their databases without revealing the entire databases to each other or even revealing the specific potential matches.

- **Collaboration among disparate agencies**: As has been well documented, different federal and local agencies do not always cooperate to the degree necessary to provide the highest security. In principle, such agencies should be able to use cryptographic protocols to determine security-relevant outcomes based on their joint data, without requiring any agency to reveal its data to the others or to a trusted third party. For example, photographic databases owned by two different agencies should be matchable in such a way that neither agency needs to trust the other with all of its images.

At this point, it is appropriate to ask whether such strong requirements could ever be satisfied. Isn't the phrase "privacy-preserving data mining" (or, *a fortiori*, "privacy-preserving surveillance") an oxymoron? In fact, the cryptographic-research community has, over almost three decades, developed tools that are extremely (almost paradoxically) powerful. Computing exactly one relevant fact about a distributed data set while concealing everything else about it is precisely what cryptographic theory enables *in principle* [35, 36, 4, 8]. Government agencies should be able to use these ideas to find security-critical matches in their databases without revealing the databases to each other. Similarly, medical practioners and researchers should be able to conduct their necessary activities while protecting the privacy of the individuals involved and complying with relevant legislation such as HIPAA [17].

More generally, researchers have developed vast bodies of cryptographic techniques (*e.g.*, [23, 20, 7, 14, 6, 19]), data-perturbation and data-sanitization techniques (*e.g.*, [3, 2, 28, 31, 34, 19, 11, 12]), and policy-specification and policy-evaluation techniques (*e.g.*, [5, 13, 21, 22, 27]) that should be useful in balancing homeland-security and other goals with individual rights.

A central focus of PORTIA is the further development of techniques that are simultaneously privacy-preserving, computationally efficient, and practical. Specific agenda items in this area include:

- Develop algorithms and protocols for privacy-protecting distributed data mining, in particular for security and surveillance applications. Consider both structured and unstructured data, simple database queries, data mining, and integration of disparate public and private databases. In many cases, this will require both advances in the state of the art of data mining and new privacy-preserving data transformations. One wide-open technical area is the algorithmics of *structural* data mining, *i.e.*, the problem of discovering ordering structures from unordered data.

- Formulate new technical definitions of privacy that are weaker than those in the existing cryptographic literature (and hence may be achievable by more efficient algorithms) but that still provide meaningful and quantifiable protection.

- Fully explore the principled use of approximation as a resource. Traditionally, approximation algorithms have been viewed as a necessary evil—substitutes for exact computations that are simply too expensive. However, recent work has shown that approximation can be used to achieve privacy preservation as well as computational efficiency [14]. PORTIA goals include the study approximation in privacy-preserving use of surveillance data, census data, *etc.*

- Develop techniques for privacy-preserving data cleaning. Most databases and data warehouses suffer from the problem that data received from external sources is likely to contain errors, *e.g.*, those introduced by noisy data collection, data-entry mistakes, missing fields, and inconsistent conventions across data sources. Thus, a lot of effort goes into *data cleaning*, the task of detecting and correcting errors in data. The PORTIA project will develop and rigorously analyze techniques for validating and correcting input data before loading, with the goal of using this cleaning stage of database construction and update as an opportunity for privacy-preserving transformations.

- Integrate the technical approaches of privacy policies and privacy-preserving data mining. Ideally, each data holder would have a machine-readable privacy policy, each data-mining system would have a machine-readable specification of what it does and does not reveal about a data set, and a policy-compliance-checking component could give data holders the information they need in order to decide whether they are willing to feed data to or receive data from the system. Detailed evaluation of mismatches between privacy policies and ostensibly "privacy-preserving" data-mining algorithms could inform the development of better algorithmic tradeoffs between privacy and information-discovery.

- Explore the social implications of proposed technological solutions. Consider both the goals of upholding current social standards and enabling new standards, particularly if the new standards might have been desirable in the face-to-face world but weren't technologically possible in that world.

- Explore the usefulness of "trusted-computing platforms," such as those under development by the TCPA [33] and by Microsoft's Next-Generation Secure-Computing Base program (formerly Palladium) [25], in privacy-preserving data transformations. Trusted platforms enable a machine to attest to a remote peer that it is running specific executable code. Thus, trusted platforms can potentially be used to implement "trusted third parties" to whom an organization can outsource data mining and data cleaning. The organization is assured that its data will never become available in the clear, because the data mining machine attested to the fact that its software deletes the data after mining it [16].

Recent progress by PORTIA participants in this area includes new protocols for computing the k^{th}-largest element of the union of confidential data sets in the multiparty case [1] and for computing the intersection in the two-party case [15].

3 Database-Policy Enforcement Tools

Much of the ongoing research in databases focuses on the challenges of making them more efficient, functional, and reliable. Database security tends to focus on access-control mechanisms that allow one to state explicitly what types of operations the various users can invoke on the relations and views constituting the database. PORTIA's goal is to complement these efforts by taking a user-centric and data-centric approach to database security. Access policies and queries tend to be quite complex in database systems, and methods for enforcing them cannot interfere with stringent performance requirements.

- **Distributed Access Control via Complex Policy.** An important part of developing a comprehensive, end-to-end technological infrastructure for handling sensitive data involves defining, managing, and enforcing information-access policies. Any organization dealing with sensitive data must develop a *published* policy governing release of information that may compromise an individual's privacy. The success of companies such as Oblix, Securant, and Netegrity illustrates the growing trend of formulating complex policies and making decisions based on them. The development of XrML, an XML-based language for expressing rights policies, also illustrates a growing interest in complex policies and their use. PORTIA goals in this area include:

  - Investigate the use of trust management [5] and other authorization frameworks in the specification of enterprise-wide information-disclosure policies. Enable the specification of policies that depend on the enterprise, the individual and organizational data users, and the data subject(s). Investigate the tradeoffs that may arise between support for these policies and the very fast access to data that today's database systems are designed to provide.
  - Develop infrastructure that supports distributed use of the policy framework by multiple, heterogeneous organizations. This infrastructure may include software supporting key management and policy exchange.
  - Develop methods and tools for policy development, testing, and maintenance. For example, the author of a policy that depends on other organizations may wish to test the consequences of her policy and examine ways in which extrinsic changes affect information flow.

- **Policy enforcement.** Can one build a *trusted data-management service* (TDMS) to enforce policies specified as above? One of the main challenges is to control the exposure of *derived data*. Derived data are obtained by transforming, copying, or aggregating data obtained from multiple agents. If one ignores the issue of derived data, as most commercial DBMSs do today, then providing a trusted data service is much simpler. The PORTIA project will address this issue explicitly and pursue two implementation approaches:

  - Wrapper-based: One can start with a conventional DBMS that provides data robustness and some fixed security and privacy mechanisms and then add a wrapper that intercepts all interactions between the DBMS and the outside world (including other wrappers). Each wrapper must be able to describe its policies to its peer wrappers, so that they can share some of the data. Initially, all programs can be required to access data through the wrapper, so that policies that are not implemented by the underlying DBMS can be enforced by the wrapper. Eventually, the wrapper will allow trusted and confined programs to run directly on the DBMS.
  - Native TDMS: Starting with an open-source DBMS (like Postgress), one can extend it to provide TDMS functionality natively. This approach is more efficient, because security and privacy rules can be enforced by the DBMS directly. Also, if the DBMS has exclusive control over the storage disks, then it is possible to track how data are modified and to confine derived database values. One important question to be addressed is whether enforcement is at the lowest level (*e.g.*, the record interface), or at a higher level (*e.g.*, the SQL level), or both.

4 Identity Theft and Identity Protection

Identity theft is the fastest growing crime in the US [30]. Identity thieves use a number of techniques to steal sensitive information in the digital world:

- Using network-based attacks, criminals break into databases and extract sufficient customer information to impersonate those customers. Online merchants and credit-card processors frequently come under such attacks, resulting in the theft of personal data about millions of customers.

- By creating replicas (a.k.a. "spoofs") of legitimate web sites, criminals trick honest customers into revealing their passwords and other identifying information. Recent spoofs of the e-Bay web site resulted in thousands of user passwords being exposed. Spoofers can then masquerade as those users on e-Bay and deceive honest e-Bay users and merchants.

- Criminals are well aware that users tend to use the same password at many different sites. Password thieves often break into the user database of a low-security site (say, a high-school reunion site) and then try all exposed username/password pairs at e-commerce sites such as e-Bay. As a result, security at an e-commerce site can be nullified by poor password management at another site, despite good information-security practices at the e-commerce site.

- Scam artists send out millions of fraudulent spam-email messages promising to provide a service, when all they actually want is to extract bank account numbers from naive users. Nigerian email scams are known to have caused at least five million dollars in damages, as well as at least one murder.

In general, improvements in privacy and authenticity of data and data access are directly relevant to preventing these attacks. Data-mining techniques have been shown to be very effective in exposing identity theft at online financial services such as PayPal and e-Bay. Using privacy-preserving data mining, one could potentially mine databases at multiple organizations without compromising user privacy or organizational policies. Mining of multiple databases could result in faster and more accurate identification of stolen identities.

How can one defend against web-site spoofing? Consider how these attacks work. A web criminal (usually outside the US) copies web pages from a high volume site such as e-Bay. He specifically takes care to make user-authentication pages look familiar and therefore apparently authentic, with all the usual banners and logos. He then sends legitimate-looking email to lots of users inviting them to take advantage of some deal on the target web site. The email contains links to the spoofed sites. Most users simply click on the link in the email without checking the link. They *enter their identifying information on the spoofed site* and are redirected to the real site. Note that SSL server authentication provides little protection against this attack; spoofed sites either turn off SSL or have a valid certificate for their own domain.

There has been recent progress by PORTIA researchers on the web-spoofing problem. Techniques used are similar in nature to those that have proved useful in spam-email filtering. One needs an automated tool that examines three factors—the contents of a web page, user actions at the page, and how the user arrived at the page—and decides, based on these factors, whether the page is a spoof or a valid page. Many heuristics can be used here. For example, if the page contains an image resembling the e-Bay logo, the page contains a password field, but the domain is not e-Bay and, furthermore, the user reached the page by clicking on an email link, then most likely the web page is a spoof. PORTIA researchers have built a browser plug-in that implements many such rules [9]. This tool will be made available to the public and will be updated as more sample spoofed pages are collected and studied.

Questions now under investigation include: What is an acceptable false-positive rate? Can collaborative spoof detection help? Should the tool automatically alert the site being spoofed? Can a trusted online service help in determining the legitimacy of a site requesting sensitive user data? Note that, if the attacker cannot get users to visit the spoofed site, the threat is greatly reduced. Building on work on spam-email reduction using puzzles [10] and trusted platforms [16], one might be able to provide additional defense against web-spoofing attacks.

A related question is whether one can build and deploy practical user-identification mechanisms that preserve privacy. For example, are there practical mechanisms by which a user can prove membership in a group without identifying himself? Similarly, are there mechanisms that reveal sensitive data only to parties that need to use it (*e.g.*, reveal a credit-card number to a credit-card processor, but not to merchants).

The problems caused by use of one password at multiple sites appear to have a simple technical solution. Suppose a user enters his password on the `www.yyy.com` login page. Just prior to sending the password to the site, the user's browser can compute a hash of the password and the domain-name `yyy.com` (technically, the browser would compute $\text{HMAC}_{pwd}(yyy.com)$) and send the resulting hash to the site instead of sending the user's password. In this manner, one ensures that a break-in at the low-security site, which now only sees a site-specific hash, will not reveal the password for the high-security site. As one might expect, making this simple idea work on the web is not that easy. There are many hurdles to overcome [29]. PORTIA researchers are currently developing a browser plug-in for this task. The plug-in would enable each user to have just one password that he can safely use at all sites. Note that this mechanism also provides a defense against web-spoofing, because spoofed sites now only obtain a worthless hash of the user's password.

5 Conclusions

The overarching goal of the PORTIA project is to produce both a next generation of technology for handling sensitive information that is qualitatively better than the current generation's and an effective conceptual framework for policy making and philosophical inquiry into the rights and responsibilities of data subjects, data owners, and data users. Along the way, project participants hope to focus public attention on the need to think more deeply about these issues and to reject simplistic hypotheses, including the assumptions that increased security requires decreased privacy or that restricting access to personal information is necessarily more important or effective than controlling use of that information. In this paper, we have given a few examples of the recent and ongoing work of project participants.

Acknowledgements

The PORTIA project is funded by the National Science Foundation, under the Information Technology Research program. Since September 2003, it has been carried out by 10 academic PIs and co-PIs (including the four authors of this paper), 12 research partners (from the computer industry, key user communities, Washington DC-based advocacy organizations, and the law and public-policy communities), and numerous students. Detailed information can be found on the project web site [26].

We thank all project participants for their crucial input to the PORTIA vision and the description of it that we have provided here.

References

[1] G. Aggarwal, N. Mishra, and B. Pinkas, " Secure computation of the k^{th}-ranked element," to appear in *Proceedings of Eurocrypt 2004*.

[2] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in *Proceedings of the 20th Symposium on Principles of Database Systems*, ACM Press, New York, 2001, pp. 247–255.

[3] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 19th International Conference on Management of Data*, ACM Press, New York, 2000, pp. 439–450.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computing," *Proceedings of 20th Symposium on the Theory of Computing*, ACM Press, New York, 1988, pp. 1–10.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, “The Role of Trust Management in Distributed Systems Security,” in Secure Internet Programming: Security Issues for Distributed and Mobile Objects, LNCS volume 1603, Springer, Berlin, 1999, pp. 183–210.

[6] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright, “Selective private function evaluation with applications to private statistics,” in *Proceedings of the 20th Symposium on Principles of Distributed Computing*, ACM Press, New York, 2001, pp. 293–304.

[7] C. Cachin, S. Micali, and M. Stadler, “Computationally private information retrieval with polylogarithmic communication,” in *Advances in Cryptology: EUROCRYPT ’99*, LNCS volume 1592, Springer, Berlin, 1999, pp. 402–414.

[8] D. Chaum, C. Crépeau, and I. Damgård, “Multiparty unconditionally secure protocols,” in *Proceedings of 20th Symposium on the Theory of Computing*, ACM Press, New York, 1988, pp. 11–19.

[9] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, “Client-Side Defense Against Web-Based Identity Theft,” to appear in *Proceedings of the 2004 Network and Distributed System Security Symposium*.

[10] C. Dwork and M. Naor, “Pricing via Processing, Or, Combatting Junk Mail,” in *Advances in Cryptology – CRYPTO ’92*, LNCS volume 740, Springer, Berlin, 1993, pp. 139–147.

[11] A. Evfimievski, J. Gehrke, and R. Srikant, “Limiting Privacy Breaches in Privacy Preserving Data Mining,” in *Proceedings of the 22nd Symposium on Principles of Database Systems*, ACM Press, New York, 2003, pp. 211–222.

[12] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, “Privacy Preserving Mining of Association Rules,” in *Proceedings of the 8th SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, ACM Press, New York, 2002, pp. 217–228.

[13] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, “SPKI Certificate Theory,” IETF RFC 2693, September 1999.

[14] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright, “Secure multiparty computation of approximations,” in *Proceedings of 28th International Colloquium on Automata, Languages and Programming*, LNCS volume 2076, Springer, Berlin, 2001, pp. 927–938.

[15] M. Freedman, K. Nissim, and B. Pinkas, “Efficient Private Matching and Set Intersection,” to appear in *Proceedings of Eurocrypt 2004*.

[16] T. Garfinkel, M. Rosenblum, D. Boneh, “Flexible OS Support and Applications for Trusted Computing,” to appear in *Proceedings of the 2003 USENIX Workshop on Hot Topics in Operating Systems*.

[17] Health Insurance Portability and Accountability Act
`http://www.cms.hhs.gov/hipaa/`

[18] J. Kaiser, “Population Databases Boom, from Iceland to the US,” *Science*, November 8, 2002, pp. 1158–1161.

[19] M. Kantarcioglu and C. Clifton, “Privacy-preserving distributed mining of association rules on horizontally partitioned data,” in *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, ACM Press, New York, 2002, pp. 24–31.

[20] E. Kushilevitz and R. Ostrovsky, “Replication is not needed: Single database, computationally-private information retrieval,” in *Proceedings of 38th Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, 1997, pp. 364–373.

[21] N. Li, B. Grosof, and J. Feigenbaum, “Delegation Logic: A Logic-based Approach to Distributed Authorization,” *ACM Transactions on Information and System Security*, 6:128-171, 2003.

[22] N. Li and J. Mitchell, “RT: A Role-based Trust-management Framework,” in *Proceedings of the 3rd DARPA Information-Survivability Conference and Exposition*, IEEE Computer Society Press, Los Alamitos, 2003, pp. 201–212.

[23] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” *J. of Cryptology*, 15:177-206, 2002.

[24] S. Mollman, “Betting on Private Data Search,” *Wired News*, March 5, 2003.
`http://www.wired.com/news/technology/0,1282,57903,00.html`

[25] Microsoft Next-Generation Secure-Computing Base – Technical FAQ
`http://www.microsoft.com/Technet/security/news/NGSCG.asp`

[26] PORTIA: Privacy, Obligations, and Rights in Technologies of Information Assessment
`http://crypto.stanford.edu/portia/`

[27] R. Rivest and B. Lampson, “SDSI: A simple, distributed security infrastructure,”
`http://theory.lcs.mit.edu/~rivest/sdsi11.html`

[28] S. Rizvi and J. Haritsa, “Maintaining Data Privacy in Association Rule Mining,” in *Proceedings of the 28th International Conference on Very Large Data Bases*, Morgan Kaufmann, San Francisco, 2002, pp. 682–693.

[29] B. Ross and D. Boneh, “Simple password management for the web,”
`http://crypto.stanford.edu/WebPwd`

[30] R. Stana, Director Justice Issues, “Identity theft,” Statement to the Subcommittee on Technology, Terrorism and Government Information, Committee on the Judiciary, U.S. Senate, Feb. 2002.

[31] L. Sweeney, “Replacing Personally-Identifying Information in Medical Records, the Scrub System,” *J. American Medical Informatics Association*, Washington, DC: Hanley & Belfus, Inc., 1996, pp. 333–337.

[32] Total Information Awareness, `http://www.darpa.mil/iao/TIASystems.htm`

[33] Trusted Computing Platform Alliance, `http://www.trustedpc.org`

[34] J. S. Vaidya and C. Clifton, “Privacy preserving association rule mining in vertically partitioned data,” in *Proceedings of the 8th SIGMOD International Conference on Knowledge Discovery and Data Mining*, ACM Press, New York, 2002, pp. 639–644.

[35] A. C. Yao, “Protocols for secure computation,” in *Proceedings of the 23rd Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, 1982, pp. 160–164.

[36] A. C. Yao, “How to generate and exchange secrets, in *Proceedings of the 27th Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, 1986, pp. 162–167.

Digital Rights Protection *

Mikhail J. Atallah Sunil Prabhakar Keith B. Frikken Radu Sion
Department of Computer Science,
Center for Education and Research in Information Assurance and Security (CERIAS)
Purdue University
Email: {mja, sunil, kbf, sion}@cs.purdue.edu

Abstract

Digital Rights Protection (DRP) is the broad class of technological, legal, and other regulatory means used to protect the rights of the owners of digital content, while simultaneously protecting the usage rights and the privacy of the users. This article briefly discusses the technological aspect of the issue.

1 Introduction

The ability to perfectly (and cheaply) replicate digital objects (software, data, etc.) has been one of the greatest advantages of the digital format over analog. While this property enables mass reproduction at low costs, it proves to be a double-edged sword since it also allows perfect reproduction for purposes of piracy. Piracy of digital works such as music and software programs is already commonplace today and results in significant losses for the owners of the digital content due to their inability to collect payment for the illegal use of pirated copies. In addition to the ease of copying digital objects, the extensive use of the Internet as a distribution medium allows widespread sharing of illegal copies. The goal of Digital Rights Protection (DRP) is to protect the Intellectual Property rights of owners of digital content. In addition to protecting the rights of owners of digital content DRP also aims to protect the usage rights and privacy of the (legitimate) consumers of digital content. The ability to share information in digital format over the Internet is highly attractive from the viewpoint of ease, speed and cost. However, the fear of piracy prevents many individuals and corporations from fully embracing this alternative. Sound DRP techniques are essential in removing this fear and encouraging more effective use of the Internet. But it is all too easy for DRP technologies to encroach on the user's rights, such as privacy (if the technologies allow tracking of the user's access to the data), or "fair use" (such as making a private copy for convenience, e.g., for use on another platform owned by the user).

DRP is challenging because it is not sufficient to defeat an average attacker, or even most attackers – rather one has to protect against the best attackers. This is because even a single compromise of the protection mechanism can lead to an unprotected copy of the digital object which can then be widely distributed. Moreover,

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Portions of this work were supported by Grants EIA-9903545, IIS-0325345, IIS-0219560, IIS-0312357, IIS-9985019 and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

the ability to easily share the instructions for compromising the protections over the Internet renders each user potentially as powerful as the most skilled attacker that breaks the protection system and shares the methodology (often through point-and-click automated scripts).

A distinction between DRP and access control can be made based upon the location of the protected object: In the case of access control, the digital object resides in the content owner's (server's) space; In the case of DRP, the object resides in the client's space. Since objects residing in the user's space can be freely accessed by the user, DRP faces unique challenges in preventing unauthorized access to and use of the data. DRP is also different from the related field of Privacy Preserving data access/sharing and Secure Multi-Party Computations. The goal of privacy preserving data access is to ensure that only desired aspects of the data are shared while hiding other aspects. For example, in privacy preserving data mining across multiple domains that do not fully trust each other, the goal is to discover interesting patterns over the union of the data without revealing the data owned by another participant to any of the participants.

2 Current Approaches

Current approaches to DRP can be divided into two broad categories based upon the nature of the protected digital object: Software or Data. Software DRP deals with ensuring that programs are not illegally replicated and disseminated. Data DRP deals with rights protection for digital objects such as images, video, and databases. Enforcing DRP for data can often be achieved through controls placed on the software. For example, videos can be distributed in a format recognized only by special software. Ensuring that this software is used only for authorized playing of these videos through software DRP methods results in protection for the media.

We now briefly discuss some of the mechanisms for protecting digital rights of copyright owners.

2.1 Watermarking

Digital watermarking aims to protect digital content by enabling provable ownership over content. This is achieved by modifying the digital objects such that identification information is embedded in the object itself. With regards to the resilience of the watermark, there are two categories: *robust* and *fragile*. A robust watermark must be resilient to attack, i.e. it must not be easy to remove the watermark without significantly destroying the object itself and thereby rendering it useless. On the other hand a fragile watermark is easily destroyed by even minor alterations to the data. Each type of watermark serves its own purpose. In addition to proving ownership of content, watermarking can also be employed to trace copyright violations and detecting modification.

An essential property of a watermark is that it modifies the digital object. It is important that the modification have minimal impact (if any) on the use of the digital object. The degree of modifications allowed depends on the nature of the object and its application. For multimedia objects such as images, acceptable change is invariably defined by the inability of the human sensory system to distinguish between the original and watermarked objects. For the case of software, the modification must ensure that the resulting code perform equivalent computation to the original object (at least with respect to the desired functionality).

Attacking a watermarked objects entails further modification of the object such that the watermark is no longer detectable. As with the insertion of the watermark, any attempt at deleting the watermark can only be considered successful if in the process the usability of the object is not destroyed. Attacks techniques include *additive* attacks that insert more data, *subtractive* attacks that throw away a large portion of the data, the insertion of another watermark, and modifications. Modifications to the object are most effective if the attacker determines what parts of the object have been modified to insert the watermark. Consequently, many watermarking schemes take great pains to hide this information through the use of secret keys. If the attacker is unable to precisely identify the location of the watermark then the modifications must necessary be random, which increases the likelihood of undesirable destruction of the value of the object. A good watermarking scheme should therefore try to maximize this likelihood,

Digital watermarks are most often used for establishing proof of ownership by inserting a robust watermark into the digital object. Ownership of an object is established through the detection of the watermark by using a secret (possibly the same secret key used in the generation of the watermark). In order to be convincing, the watermark should be such that it would be highly unlikely that the watermark could occur by accident. A good watermarking method should be immune to the "torturing the data" defense. This line of defense claims that the alleged owner of the object has exhaustively searched this digital object using various keys and has thereby "discovered" this watermark. Such claims can be countered by publishing a one-way cryptographic hash of the secret key in a time-stamped manner (such as in a newspaper).

Inserting different watermarks in different copies of the same object can be used to record information about each copy, in addition to information about the owner of the copyright. This additional information can include the identity of the purchaser of that copy. Such watermarks can be used as *fingerprints* in order to trace copyright violations. A pirated copy would bear a watermark that uniquely identifies the initial purchaser of this copy. A down-side to fingerprinting is that it is vulnerable to a special kind of attack – the "diff" attack. In his attempt to remove a watermark, an adversary can be much more effective if he can identify the locations of the watermark. These locations are typically hidden through the use of secret keys during embedding. With the availability of multiple watermarked versions of the same data, a simple comparison easily identifies potential locations of the watermark. These can then be altered yielding the watermark ineffective without having to make major modifications to the object. Although fingerprinting can help identify the original recipient (and thereby the source of the piracy) the culpability of the recipient is a different matter – e.g. one could claim that the object was stolen and pirated by the thief.

The ability of robust watermarks to survive significant modifications can be used to serve other purposes that require tamper-resistance. For example, a watermark that attests to some fact can serve as a credential. The content of the watermark text can be used to embed control information such as restrictions on the use of the data. For example, a media player may only play a file if it contains a watermark with the appropriate permissions embedded in it. On the other hand, the intolerance of a fragile watermark to even minor changes can be used as a means for establishing the integrity of the data. The absence of the watermark is an indication that some unauthorized modification of the file has taken place.

The most common application of digital watermarking has been for the domain multimedia data such as images and video. The watermark is typically embedded in either the original domain of the object, such as by modifying the low order bits of pixel values, or in more sophisticated methods, in the corresponding frequency domain. In frequency domain watermarking, the object is first converted to the frequency domain through transformations such as the Fast Fourier Transform (FFT) or the Discrete Cosine Transform (DCT). The watermark is embedded in this domain through minor modifications, and then the watermarked object is generated through the corresponding inverse transformation back to the original domain. Due to the reliance of these methods on the relative insensitivity of the human sensory system, and the regular structure of media data, these watermarking methods are typically not applicable to other domains such as software and data. Watermarking has also been applied to software, natural language text, and more recently, structured and semi-structured data. Watermarking of software is achieved through modifications to the structure of the code (such as IBM's use of a very specific register loading order) or its behavior. Watermarks that modify behavior may be detectable at run-time only if a secret key is provided.

Watermarking structured data, such as a relational database is an interesting problem that has received little attention thus far. A unique feature of this domain is that the issue of acceptable change is not as clear as in other domains. Unlike the multimedia domain, the insensitivity of the human sensory system is not available since even minor changes in some critical data values can be important. More importantly, acceptability of changes to data values is highly application dependent.

Overall, watermarking is an important tool for Digital Rights Protection since it enables proof of ownership of content, traceability of pirated copies, integrity testing, and authentication.

2.2 Code Obfuscation

In instances where the intellectual property to be protected lies in the algorithms or data structures in a piece of software, it is necessary to ensure that malicious users are not able to reverse engineer the code. Code Obfuscation can be used to achieve this purpose. Code Obfuscation is the process of mangling code and data structures used in the code so that it becomes difficult to understand the functioning of the program. These methods are also valuable in making protection mechanisms in the code harder to defeat or circumvent. For example, a simple implementation of a software check that requires a password or key to work correctly typically contains a conditional jump statement that makes the critical decision of whether or not the password or key is valid. If an attacker is able to discover this statement (for example using code cracking tools such as those discussed below), the check can easily be defeated by changing the jump to an unconditional jump or other similar change. Code obfuscation cannot guarantee that the attacker will not be able to reverse engineer the obfuscated code. However, it can make the task highly non-trivial rendering it difficult for a novice attacker to succeed, and at least significantly slowing down expert adversaries.

In order to be effective (i.e. resilient to attack) and efficient, Code Obfuscation has to meet certain requirements, including: 1) the original functionality of the software must remain unchanged; 2) the modifications should not be easy to remove using automatic tools; 3) the resulting performance degradation should be acceptable; and 4) the modifications should not be removed by a compiler during compilation. Obfuscation techniques fall into one of the following categories:

1. *Data* obfuscation reorganizes data structures and the code that uses them so as to obscure their role. Examples include merging of unrelated data, or taking apart data that are related.

2. *Control* obfuscation aims to hide the true control flow of the program by entwining it with segments of code that have no actual influence on the control structure of the program. The modification must make it difficult to identify these irrelevant portions of the resulting program. Naturally, control obfuscation must survive the compilation process – changes that the compiler is able to discard due to their irrelevance (such as a function that is never called) will not work.

3. *Layout* obfuscation deals with the rewriting and rearranging of the source code of a program to make it harder to read. Examples of layout obfuscation include the removal of comments, variable substitutions, and altering the formatting of statements.

An important tool for obfuscating segments of code is the use of *opaque predicates*. An opaque predicate is a conditional test that will always generate the same answer although it would be very difficult to arrive at this conclusion simply through an examination of the code. Opaque predicates can be inserted in code to give the appearance of being integral parts of the code whose "role" is difficult to discern.

Methods of Attack We briefly mention some of the methods that can be employed to defeat rights protection mechanisms discussed above. An important step in circumventing software protections is to first understand the functioning of the code. Commonly available tools for debugging, decompiling or disassembling tools can be used to trace the control flow during execution of the program, locate interesting events in the code, and generate "cleaner" versions of the code. Many protection related segments of code perform encryption. Performance profilers and pattern matching can be used to identify such pieces of code, which can then be subjected to attack.

2.3 Other Methods

Encryption Wrappers One way to hide code or data is to store it in encrypted format using a secret key. This data is automatically decrypted when it is needed. Since the data is to be decrypted on the user's computer, an attacker would be able to obtain the decrypted version of the data. Thus the key to this technique is to prevent the user from easily obtaining the decrypted version. This is achieved through techniques such as preventing the use of tools that can dump the decrypted code to disk or debugging the program at run-time. Alternatively, the software decrypts the program in small chunks thereby making it difficult to obtain a snapshot of the entire program at any one time. Thus an attacker must obtain numerous snapshots and integrate them to obtain a copy of the entire encrypted program. A weakness of encryption wrappers is that the decryption key is contained within the encrypted data and thus can be discovered through an analysis of the program.

Hardware Techniques Although software techniques for DRP have been studied and used extensively, it has been established that it is impossible to have completely secure protection using software alone. This makes hardware protection highly desirable if strong guarantees of protection are needed. Unlike software protections, defeating hardware protections is much harder and may require special equipment. In addition, even if a hardware protection mechanism is defeated by a determined and expert attacker, it may not be as easy to share the compromise with others.

Hardware-based DRP methods make use of devices such as trusted hardware processors or peripheral devices (often called Dongles). Typically, dongles are small devices that are connected to the client computer and interact with the protected program. The role played by the dongle can vary from simply establishing the right to use a digital object by their presence, to playing an essential role in the program. The use of dongles is expensive for the producer of the digital content and may be inconvenient for the user since a dongle is needed for each copy of protected software. Trusted processors provide a means for executing only software that has not been corrupted (such as bypassing a critical authorization check) – this includes all types of code run on the machine including the operating system. The extensive use of encryption that goes along with the use of these trusted processors can be prohibitive for a low-end market. In addition, it can lead to limitations on the use of the system for other purposes that are not related to the legitimate protection of digital rights.

2.4 DRP for Databases

The problem of protecting ownership rights over collections of data such as in a database, or XML format has recently begun to draw the attention of researchers. It has become clear that existing DRP techniques cannot directly be ported to this domain. Consider the case of watermarking wherein the notion of acceptable change is highly dependent upon the intended use of the data. An encrypted version of the data with minor changes may be acceptable change for some applications while not for others. The current results on watermarking of relational data are only the first steps towards enabling robust DRP solutions for structured data.

3 Conclusion

The ease of replication and distribution of digital media make it attractive for content owners as a means of cheap, rapid and widespread distribution. However, these very same properties make it difficult to assert ownership rights and collect payment over the same objects. Digital Rights Protection mechanisms therefore play an important role in enabling content owners to prove ownership of digital content and to limit illegal distribution and use of digital content. Both software and hardware techniques have been developed that provide a wide variety of protections. Software only solutions cannot guarantee protection – and may eventually be compromised by a determined and expert adversary. Thus these solutions achieve the purpose of being deterrents and delaying the adversary. The focus of DRP solutions has been software and media objects such as images. The problem of protecting ownership rights over collections of data such as in a database, or XML format has recently begun to draw the attention of researchers. It has become clear that existing DRP techniques cannot directly be ported to this domain. The current results on watermarking of relational data are only the first steps towards enabling robust DRP solutions for structured and data.

References

[1] R. Agrawal and Jerry Kiernan. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Databases VLDB*, 2002.

[2] R. Anderson *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wily, John & Sons, Inc. February 2001, p. 413-452.

[3] R. Anderson "TCPA/Palladium Frequently Asked Questions" *www.cl.cam.ac.uk/ rja14/tcpa-faq.html*.

[4] R. Anderson "Security in Open versus Closed Systems - Dance of Blotzmann, Coase, and Moore" *www.ftp.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf*.

[5] R. Anderson, M. Kuhn "Tamper Resistance - a Cautionary Note" Proceedings of Second Usenix Workshop on Electronic Commerce, pp. 1–11, November 1996.

[6] R. Anderson, M. Kuhn Low Cost Attacks on Tamper Resistant Devices Proceedings of the 1997 Security Protocols Workshop, Paris, April 7–9, 1997.

[7] W. Arbaugh, D. Farber, and J. Smith "A Secure and Reliable Bootstrap Architecture" Proceedings of the IEEE Symposium on Security and Privacy, 1997.

[8] Mikhail Atallah, Keith Frikken, Carrie Black(*), Susan Overstreet, and Pooja Bhatia "Digital Rights Management" *Practical Handbook of Internet Computing*, Munindar P. Singh (Ed.), CRC Press, 2004.

[9] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang *On the (Im)possibility of Obfuscating Program* Electronic Colloquium on Computational Complexity, Report No. 57, 2001.

[10] Elisa Bertino, M. Braun, Silvana Castano, Elena Ferrari, and Marco Mesiti. Author-x: A java-based system for XML data protection. In *IFIP Workshop on Database Security*, pages 15–26, 2000.

[11] T. Budd "Protecting and Managing Electronic Content with a Digital Battery" *Computer*, vol. 34, no. 8, pp. 2-8, Aug 2001.

[12] A. Caldwell, H. Choi, A. Kahng, et. al. "Effective Iterative Techniques for Fingerprinting Design IP" UCLA Computer Science Department, 1999.

[13] J. Camenisch "Efficient Anonymous Fingerprinting with Group Signatures" IBM Research, Zurich Research Laboratory, 2000.

[14] C. Collberg, C. Thomborson, and D. Low *A taxonomy of obfuscating transformations*. Tech. Report # 148, Department of Computer Science, University of Auckland, 1997.

[15] C. Collberg and C. Thomborson *Watermarking, Tamper-Proofing, and Obfuscation Tools for Software Protection*. IEEE Transactions on Software Engineering. Vol. 28 No. 8 pages 735-746, 2002.

[16] M. Jakobsson and M. Reiter "Discouraging Software Piracy Using Software Aging" Digital Rights Management Workshop 2001: 1-12.

[17] A. Kahng, J. Lach, W. Mangione-Smith, S. Mantik, I. Markov "Watermarking Techniques for Intellectual Property Protection" UCLA Computer Science Department, 1998.

[18] R.J. Lipton, S. Rajagopalan, D.N. Serpanos "Spy: a method to secure clients for network services" IEEE Distributed Computing Systems Workshops 2002:23-28.

[19] T. Maude and D. Maude "Hardware Protection Against Software Piracy" *Comm. of ACM*, vol. 27, no. 9, pp. 950-959, Sep 1984.

[20] J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang. Experience with software watermarking. In *Proceedings of ACSAC, 16th Annual Computer Security Applications Conference*, pages 308–316, 2000.

[21] B. Pfitzmann and A. Sadeghi "Coin-Based Anonymous Fingerprinting" Universitaet des Saarlandes, Fachbereich Informatik. Saarbrcken, Germany. 1999.

[22] B. Pfitzmann and M. Schunter "Asymmetric Fingerprinting" Universitaet Hildesheim, Institut fr Informatik. Hildesheim, Germany. 1996.

[23] G. Qu and M. Potkonjak "Fingerprinting Intellectual Property Using Constraint-Addition" Computer Science Department, University of California Los Angeles, 2002.

[24] G. Qu, J. Wong, and M. Potkonjak. "Optimization-Intensive Watermarking Techniques for Decision Problems"

[25] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons Inc., 2000.

[26] M. Seadle, J. Deller, and A. Gurijala "Why Watermark? The Copyright Need for an Engineering Solution" JCDL. July 2002.

[27] R. Sion, M. Atallah, S. Prabhakar *Rights Protection for Relational Data* ACM International Conference on Management of Data, SIGMOD 2003,

[28] C. Wang, J. Hill, J. Knight and J. Davidson *Software Tamper Resistance: Obstructing Static Analysis of Programs* PhD Dissertation, University of Virginia.

[29] G. Wroblewski *General Method of Program Code Obfuscation*, PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002

Databases that tell the Truth: Authentic Data Publication*

Michael Gertz[1] April Kwong[1] Charles U. Martel[1] Glen Nuckolls[2]
Premkumar T. Devanbu[1] Stuart S. Stubblebine[3]
[1] University of California at Davis, CA
[2] University of Texas at Austin, TX
[3] Stubblebine Research Labs, Madison, NJ

Abstract

The publication of high-value and mission critical data on the Internet plays an important role in the government, industry, and health-care sectors. However, owners of such data are often not able or willing to serve millions of query requests per day and furthermore satisfy clients' data requirements regarding the integrity, availability, and authenticity of the data they manage in their databases.

In this article, we give an overview of our work on authentic publication schemes in which a data owner employs a (possibly untrusted) data publisher to answer queries from clients on behalf of the owner. In addition to query answers, publishers provide clients with verification objects a client uses to verify whether the answer is the same as the owner would have provided. We consider two popular types of database systems, those managing relational data and those managing XML data in the form of XML repositories.

1 Introduction

In today's information-centric society, the reliable dissemination of information over the Internet plays a crucial role, in particular when the data is vital to high-value decisions in financial, health-care, and government sectors. Owners of such data must provide mechanisms that ensure integrity, authenticity, and non-repudiation requirements of data consumers. However, providing such a protection over public networks is an expensive matter. In addition to maintaining a secure computing and database infrastructure that ensures the availability of data and query services, owners have to digitally sign results of queries submitted by clients to ensure non-repudiation of the results. Data owners, however, may not be willing or able to maintain an infrastructure that protects signing keys against attacks and can furthermore manage millions of queries a day.

A possible solution to this problem is that the data owner uses third-party data publishers that provide for a secure and efficient computing infrastructure. A data owner periodically distributes the data to publishers, which then answer queries from clients on behalf of the data owner. Although this approach is much more scalable, a client querying a data publisher might worry that the publishers engages in deception and does not give the same result to a query as the data owner would have given. The client would also have to trust the key management of the publisher. In general, clients simply might not trust data publishers.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Funding for this work was provided by the NSF ITR award 0085961.

In this paper, we present an approach called *authentic publication scheme* where a data owner employs one or more *untrusted data publishers*. In this approach, which is illustrated in Figure 1, the data owner employs digests that are bottom-up hashes computed recursively over tree-type structures representing the entire set of objects in the owner's database. These digests are distributed to all clients interested in the owner's data. This is done in a secure fashion by using, for example, signing keys. The owner then distributes his data to data publishers and finally goes off-line. When a clients now issues a query against a data publisher, he receives a query result and a *verification object (VO)*. This VO is generated by the publisher and is based on the bottom-up hash scheme previously used by the data owner. The client then uses the VO, previously distributed digests, and the query result to verify that the result is correct, i.e., the same result would have been computed by the owner. The VO in combination with the digest realize a hard to forge proof the client uses to verify the answer. If the publisher provides the client with an incorrect query result or forged verification object, the client is able to detect such cases of possible deception.

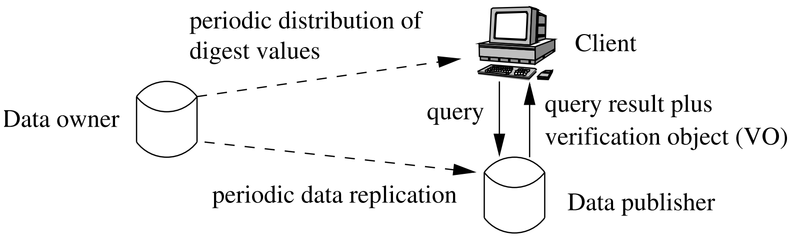

Figure 1: Authentic Publication Scheme

Our approach is founded on cryptographic assumptions regarding the security of hash-functions and public key systems. Compared to related approaches, there are several advantages of our authentic publication scheme. First, a client only has to trust the digests distributed by the data owner; he does not have to trust the publisher. Second, data owners only have to periodically distribute the digest of their data to clients and their data to data publishers, and remain off-line otherwise. Third, verification objects are nearly linear in the size of the answer to a query.

In the following section, we present the authentic publication scheme in the context of relational databases and describe how data owner and data publishers compute digests and verification objects for different types of queries, respectively. In Section 3, we illustrate how the authentic publication scheme is used in the context of databases that manage XML data. We conclude in Section 4 with a summary and outline of future work.

2 The Relational Case

In the following, we present the authentic publication scheme for the case where data owner and publisher employ a relational database. We first present the concept of Merkle-Hash Trees owner and publisher use to compute digest values and verification objects for query results, respectively. We then illustrate how results for different types of queries against a relational database can be verified by clients. Details of this approach can be found in [3, 5].

2.1 Merkle-Hash Trees

The authentic publication scheme is based on digests (distributed by data owners) and verification objects (computed by data publishers) that are determined from tree-type structures over sets of data. For this, we adopt a Merkle-Hash Tree approach [10]. Assume a relation R with attributes $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$. A Merkle-Hash

Tree over R, denoted $MHT(R, \mathcal{A})$, is a binary tree with $|R|$ leaf nodes and hash value $h(i)$ associated with each node i of the tree as follows.

1. First, for each tuple $t \in R$, a tuple hash $h_t = h(h(t.A_1)||h(t.A_2)|| \ldots ||h(t.A_n))$ is computed using a collision resistant hash function (|| denotes the concatenation operator). For a hash length of 128 bits, for example, the probability of random collisions of hash values is close to 2^{-128}.
2. Next, assume a total order on R, based on the primary key in $\mathcal{A}$ or any other sequence of attributes in $\mathcal{A}$ (which might include the primary key as last sorting criterion). The hash value $h(i)$ for a node $i \in MHT(R, \mathcal{A})$ is computed as follows.
   (a) If i is a leaf node, then $h(i) = h_t(t_k), k = 1, \ldots, |R|$.
   (b) If i is an internal node, then $h(i) = h(h(l), h(r))$, where l and r are the left and right children of node i.

The root hash, denoted $h_{R,\mathcal{A}}$ is the *digest* of all values in the Merkle-Hash Tree $MHT(R, \mathcal{A})$. Figure 2 illustrates the computation of the digest for a sorted relation. An important property of this approach is that if the hash

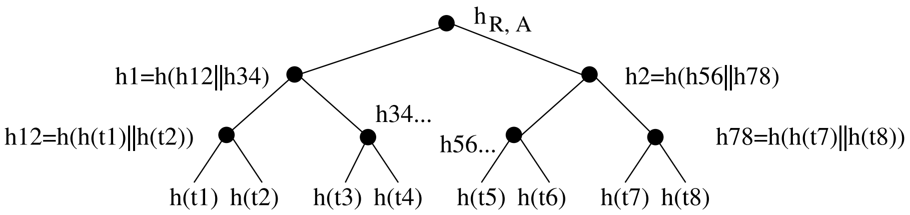

Figure 2: Computation of digest using Merkle-Hash Tree on a sorted relation with tuples $t1, \ldots, t8$

value for a node is known, then the hash values of its children (and values of tuples in the case of leaf nodes) cannot be forged. This property is based on cryptographic assumptions of collision-resistant hash functions and has initially been used by Naor and Nissim in their work on revocation and updates of certificates [13].

In our approach, the data owner computes digests for different relations and sorting criteria used in client applications, and securely distributes the digests to clients. The data publisher maintaining a copy of the owner's data uses the Merkle-Hash Tree scheme to compute verification objects for query results. That is, the hash function used by the owner for computing digests is known to clients and publisher. An important concept in computing verification objects is that of a *hash path*. A hash path describes nodes and node values in a Merkle-Hash Tree that are necessary and sufficient to recompute the root digest and thus to verify the existence (or absence) of tuples in a relation.

Definition 1: Let i be the leaf node corresponding to a tuple $t \in R$ in a Merkle-Hash Tree $MHT(R, \mathcal{A})$. The hash path for i, denoted $path(i)$, comprises all nodes necessary to compute the root digest $h_{R,\mathcal{A}}$. Such a hash path always has the length $\lceil \log_2(|R|) \rceil$ and comprises $2 * \lceil \log_2(|R|) \rceil - 1$ nodes where exactly two nodes are leaf nodes. Of these, only $\lceil \log_2(|R|) \rceil + 1$ are needed to recompute $h_{R,\mathcal{A}}$. Hash paths can also be provided for non-leaf nodes.

Example 1: Consider the tuple $t6$ in Figure 2. In order to recompute $h_{R,\mathcal{A}}$, the tuple $t5$ (or its hash value $h(t5)$) and the values $h78$ and $h1$ are necessary and sufficient. If a client knows the digest $h_{R,\mathcal{A}}$ and the tuple $t6$ is in R, then any incorrect value for $t6$ or nodes in $path(t6)$ should result in a hash value different from the digest. Now assume one wants to show that there is no tuple t' in R such that $t5 < t' < t6$ (given the criterion on which R is sorted in $MHT(R, \mathcal{A})$). The two tuples $t5$ and $t6$, and the node values $h78$ and $h2$ would suffice to prove this case. The two tuples $t5$ and $t6$ can be considered *witnesses* for the absence of t'. Any other combination of tuples or incorrect (forged) hash values for inner nodes would again result in a digest different from $h_{R,\mathcal{A}}$.

The nodes in $path(i)$ constitute a *verification object (VO)*, showing that tuple t_i is (not) in relation R with digest $h_{R,\mathcal{A}}$. Verification objects are computed by data publishers and passed on to clients in addition to the result of a query. Clients take the query result and the VO and recompute the digest as illustrated above (details of a respective protocol are given in [3]). If the recomputed digest is the same as the digest previously distributed by the data owner, the client can be assured that (1) no qualified tuple has been left out from the result, and (2) the result only contains qualified tuples. If the digests do not match, then either the query result is incorrect (i.e., different from what the data owner would have computed) or hash values have been forged.

The above approach can be easily extended to provide a proof for the existence of a sequence of tuples in a relation R. Recall that we always assume a total order among tuples representing the leaves in a Merkle-Hash Tree $MHT(R, \mathcal{A})$. To prove the existence of a certain sequence s of tuples, there are always two tuples that are not included in s, and they describe the greatest lower bound $glb(s)$ and lowest upper bound $lub(s)$ of s, respectively. If the tuple $glb(s)$ ($lub(s)$) does not exist in R, its value is determined based on the smallest (largest) tuple in s. Now let $cover(s)$ denote the inner node in $MHT(R, \mathcal{A})$ that is the root node of the minimal subtree in $MHT(R, \mathcal{A})$ having all tuples in s as leaf nodes. Obviously, for the leaf nodes $glb(s), lub(s)$ and the inner node $cover(s)$, there always exists a lowest common ancestor node, denoted $lca(s)$, which in the extreme case is the root of the Merkle-Hash Tree.

To prove the existence of s, the VO then basically comprises (1) the hash path for $lca(s)$, and (2) the hash paths for $glb(s)$ and $lub(s)$. Given the sequence s of tuples, this information is necessary and sufficient to show that s is in R. The same approach is used to prove that a sequence is empty (using $glb(s)$ and $lub(s)$ as witness tuples), thus showing that a tuple does not exist in R. In the following, we call VOs for cases where witness tuples $glb(s)$ and $lub(s)$ are used to "surround" (non-existing) sequences of tuples as *boundary case VOs.*

Example 2: Consider the sequence $s = \langle t6, t7 \rangle$ in Figure 2. We have $glb(s) = t5$ and $lub(s) = t8$ as "near miss" tuples, and $cover(s) = h2$. The node $h2$ also happens to be $lca(s)$. In this case, the VO for s includes hash paths for $glb(s)$ and $lub(s)$ to the node $h2$, and the hash path for $h2$. Using the values for $s6$ and $s7$, the digest of the Merkle-Hash Tree can be recomputed.

2.2 Verification of Answers to Queries

In the following, we illustrate the computation of verification objects (VOs) in the context of different types of queries formulated in relational algebra. The types of queries are known to the data owner and publisher. This is a reasonable assumption since queries are typically embedded in client applications and not ad hoc, and clients are interested in obtaining proofs for results to their queries. We assume that respective digests on Merkle-Hash Trees that support such query patterns have been distributed by the data owner to clients and that the data publishers knows about these Merkle-Hash Trees structures.

Selection. Suppose a client issues selection queries of the form $\sigma_{A_i \Theta C}(R)$ against the publisher's database, and a result set $q \subseteq R$ is returned by the publisher. Tuples in the Merkle-Hash Tree supporting the computation of VOs are sorted on A_i. The following cases are considered for Θ being the equality predicate: (1) if A_i is the primary key in R and $q = \{t\}, t \in R$, then there exists exactly one tuple satisfying the query. In this case, the VO is simply the hash path for t. (2) if A_i is the primary key in R and $q = \emptyset$, then the VO is based on the two witness tuples that would "surround" the non-existing result tuple. That is, the VO is a boundary case VO as illustrated at the end of Section 2.1. (3) A_i is not a primary key and $q \neq \emptyset$. In this case, the result set builds a contiguous sequence s of tuples in $MHT(R, A_i)$. Also in this case, a boundary case VO is constructed. In case A_i is not the primary key and $q = \emptyset$, the same approach as for case (2) is applied. For selection conditions based on a predicate $\Theta \in \{\neq, <, >\}$, results to a query typically comprise one or two sequences of tuples (leaf nodes in $MHT(R, \mathcal{A})$) that satisfy the condition. Thus, again boundary case VOs are used to show the correctness of respective answer sets.

In all the above cases, the size of the VO is linear in the size of the query result and $\log(|R|)$. The VO plus query answer are sufficient to show that the answer to a selection query provided by the publisher is exactly the same as the data owner would have computed.

Projection. Computing VOs for projection queries of the form $\pi_{\mathcal{A}'}(R)$, where $\mathcal{A}'$ is a proper subset of attributes in R, involves two components. Assume a Merkle-Hash Tree in which the tuples (leaf nodes) are sorted based on $\mathcal{A}'$. Each tuple t in an answer q to a projection query potentially results from a set of tuples $S(t) \subset R$, where tuples in $S(t)$ have identical values for the attributes $\mathcal{A}'$. To show that t is in the answer set, the hash path for a witness tuple from $S(t)$ is provided as part of the VO. To show that there are no missing tuples in between two tuples t and t' $(t, t' \in q)$, the VO also has to provide information that the sets $S(t)$ and $S(t')$ are consecutive sequences in $MHT(R, \mathcal{A}')$. This again is done using boundary case VOs.

Joins. We illustrate the case for the most common type of join, the natural join $R \bowtie S$. The Merkle-Hash Tree supporting the computation of respective VOs is constructed as follows. Assume attribute A occurring in both R and S is the join attribute. Using the full outer join $R ⟗ S$, three groups of leaf nodes are obtained for the Merkle-hash Tree: (1) tuples that result from the natural join $R \bowtie S$, (2) right null-padded tuples from R for which no matching tuples in S exist, and (3) left null-padded tuples from S for which no matching tuples in S exist. That is, the data publisher materializes potential results to join queries. Such an approach is reasonable since our scheme works under the assumption that data publishers can provide the computing and database infrastructure necessary to efficiently answer millions of queries a day. The data owner, on the other hand, has to materialize the respective Merkle-Has Tree only once to determine and distribute digests to clients.

VOs for answers to natural join queries obviously employ boundary case VOs, as described in Section 2.1. It should be noted that the materialization of $R ⟗ S$ has another advantage, namely that VOs for answers to queries that have conditions on the non-existence of matching tuple can be easily computed.

Complex Queries. We conclude this section with a brief outline of how queries are handled that contain nested operators, in particular those that resemble select-from-where queries used in SQL. The key idea, which is detailed in [3], is to use so-called *multi-dimensional range trees*, a data structure used in computational geometry [1] to deal with sets of points in a multi-dimensional space. In this approach, an initial Merkle-Hash Tree is computed, say for a complete and sorted relation. This tree is used to computer partial VOs for outer query components, e.g., a final selection of tuples that satisfy a (join) condition. Each inner node of that tree (a node covering a sequence of leaf node tuples) is linked to another Merkle-Hash Tree. This tree then supports the computation of VOs for subsequent query component on only the covered tuples, e.g., a join of only these tuples with tuples from another relation or another selection condition. The construction and maintenance of such "cascading Merkle-Hash Tree" relies on the assumption that the patterns of queries submitted by clients to the data publisher are known and that the publisher has sufficient resources to constructs such tree structures based on to the queries patterns and the data owners relations.

3 The XML Case

Recently there have been several advancements in using the eXtensible Markup Language (XML) for the exchange, management, and querying of information on the Internet. Several applications settings in industry and government already employ XML to exchange and manage document data in a flexible and standardized fashion. In these settings, XML Repositories (based on native XML or relational database architectures) are built that contain collections of XML documents and data. The repository data is made available to client applications, where clients request documents or data that satisfy specified conditions based on path or tree pattern queries.

Analogous to the publication of data residing in relational databases, the integrity, authenticity, and non-repudiation of XML repository data are important requirements clients would like to see satisfied by the repository's owner when querying the repository. To address this issue in cases where XML repository owners are unable or unwilling to provide clients with a respective computing infrastructure, we extend the authentic publication scheme introduced in the previous section to XML repositories. In the following, we first outline the XML data model and query approach underlying our scheme. We then illustrate how digests for an XML repository are constructed by the owner, and how verification objects for query results are computed by repository publishers. More details and different uses of this approach are given in [2, 4, 9].

3.1 XML Data Model and Path Queries

We assume a general XML data model in which XML data is represented as an ordered, node-labeled tree. Nodes in the tree are labeled with element names and leaf nodes carry text data. To motivate the basic idea of our approach, in the following, we do not consider attributes, comments, entities, and processing instructions. We assume that a complete XML repository is modeled as a single tree and that a schema in the form of a Document Type Definition (DTD).

We use simple path queries to query an XML repository. The language underlying these queries is a subset of XPath [15]. It supports the child axis ("/"), descendant axis ("//"), wildcard ("$*$"), and conditions on text values of selection nodes in a query. We currently do not consider tree pattern queries, i.e., queries that include branching ("[]"). This is not a limitation of the approach but is related to scalability issues since the number of possible path queries against an XML repository is much smaller than the number of possible tree pattern queries.

3.2 XML Repository Digests and Certified Query Results

In order for a data owner to employ our authentic publication scheme, a model is needed that allows the computation of digest values for XML repository data. The data publisher maintaining a copy of the repository then uses the same model to compute verification objects for answers to path queries posed by clients. There are several approaches to hash and sign XML data, such as DOMHASH [6] and the XML Digital Signature Recommendation [14], respectively. All these models provide a means to perform a recursive, bottom-up hashing over tree structured data. In particular, the XML Digital Signature scheme describes a single signature over an entire XML repository to validate the authenticity of any subtree in the repository. Although in principle this approach can be used by a publisher to prove to a client that every subtree returned as part of a query result is in fact in the XML repository, it does not provide a means to prove that all qualified subtrees are included in an answer. Our approach eliminates this limitation.

We start with a solution to compute digests and verification objects for an XML repository and query results, respectively. Again, our aim is to find a Merkle-Hash Tree-like structure on data objects that supports these types of computations. Suppose a DTD D is associated with the owner's XML repository. We assume that D is non-recursive (although this assumption can be weakened by considering a fixed recursion depth). Based on D, a so-called *XTrie* is computed. An XTrie is a rooted, node-labeled tree, and it enumerates all admissible rooted paths, i.e., sequences of element names, in a repository.

Given the owner's XML repository R conforming to the DTD D and a node i in the Xtrie, one can identify all subtrees in R whose rooted path corresponds to the path leading to node i. Let $s_1, s_2, \ldots, s_n$ denote the subtrees in R, in repository order, that are rooted at node i. Each subtree s_i is hashed using a secure hash-function (see, e.g., DOMHASH [6]); let $h(s_i)$ denote the computed hash value. The hash values $h(s_1), h(s_2), \ldots, h(s_n)$ now build the leaf nodes of a Merkle-Hash Tree introduced in Section 2.1. Thus, a digest for the n subtree structures is determined and associated with the node i in the Xtrie. This procedure is repeated for every node in the Xtrie. Thus, for each node in the Xtrie, a digest is computed based on subtree structures in the repository R that are rooted at that node.

Eventually, an Xtrie is obtained where each node carries a digest of all subtree structures in the XML repository that are rooted at that node. Such an XTrie, although not a binary tree (in the Xtrie, a node can have more or less than two children), can be Merkle-hashed bottom-up, resulting in a digest for all subtree structures in the repository. This digest is securely distributed to clients by the repository owner. The final step for the repository owner is to provide a publisher with an exact copy of the XML repository.

We now consider the processing of client queries and the computation of verification objects at the publisher site. We assume that clients know the Xtrie structure, for example, as schema for formulating queries (note that an Xtrie is less expressive than a DTD). A path query submitted to a publisher is processed against the Xtrie to find the root of subtrees whose path matches the path query. With each such path query, zero, one or more nodes in the Xtrie can be associated, depending on the pattern used in the path query. Assume one node in the Xtrie matches the query. As answer to the query, the publisher will provide the client with the sequence of subtrees from the XML repository whose root node matches the query in the Xtrie. Furthermore, the publisher computes a verification object that comprises the following components: (1) the rooted path of the node matching the query, and (2) the hash path for that node (see Definition 1). The client uses this information to recompute a digest. If it matches the digest previously distributed by the owner, the client can be assured that the publisher provided all matching subtrees that can be found in the XML repository and no subtree has been left out from the result.

The above scheme easily extends to cases where more than one node in the Xtrie matches the path query or no node matches the query. Note that the client, having information about the Xtrie, can easily verify how many matches there have to be and, thus, how many VOs the publisher has to provide. In both scenarios, boundary case VO, as illustrated in the previous section, are computed by the publisher. An interesting feature of the authentic publication scheme for XML repositories is that basically only one digest is necessary, namely the digest over the complete XML repository as described above. Recall that in the case of relational databases, digests are fairly specific to the types of queries issued by clients and therefore require more digests and query specific Merkle-Hash Tree structures for tuple data. The approach for computing a digest for an XML repository and verification objects associated with answers to path queries discussed above does not address path queries that contain condition on text values and text nodes (leaves) of an XML repository. A respective extension to the above approach as well as complexity results for the authentic publication of XML data are given in [3].

4 Conclusions and Future Work

Authentic publication schemes provide data owners with an effective means to employ possibly untrusted data publishers to answer queries from clients on behalf of the data owner. The schemes outlined in this article focus on two popular types of databases as they can be found in practice, relational databases and XML repositories. The proposed schemes require only a minor effort by the data owner for computing digests on the data objects managed in the database and periodically distributing these digests to clients interested in the owner's data. In particular, our schemes satisfy important security requirements, namely that a client can verify – based on verification objects associated with query answers – that the query result computed by a data publisher is the same as the owner would have provided. The techniques for computing digests and verification objects, of course, add to the complexity of evaluating queries. However, since we assume mostly static data, i.e., data that is only updated once week or month, this is not disadvantage, in particular since the owner (with constrained resources) has to compute digests only periodically.

More general approaches for authenticating data structures than those presented in this article have been proposed by Goodrich et al. [8] and Martel et al. [11]. Another extension allows multiple owners to jointly certify a digest for a data set that combines their individual data sets [12]. We are currently studying several alternatives for implementing our authentication schemes, with a primary focus on relational databases. Using the GiST package [7], we are evaluating different types of index structures that can be used for both answering queries and "encoding" Merkle-Hash Tree structures and hash values in particular. Our focus is also on developing lightweight software components clients can plug into their applications to automatically perform the verification of VOs against digests distributed by the data owner. The purpose of these components is to make the verification process for queries fully transparent to applications querying a publisher's database.

References

[1] de Berg, M., Schwarzkopf, O., van Kreveld, M., Overmars, M.: *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[2] Devanbu, P., Gertz, M., Kwong, A., Martel, C., Stubblebine, S.: Flexible Authentication of XML Documents. To appear in *Journal of Computer Security*.

[3] Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.: Authentic Data Publication over the Internet. In *Journal of Computer Security*, 11(3), 291–314, 2003.

[4] Devanbu, P., Gertz, M., Kwong, A., Martel, C., Stubblebine, S.: Flexible Authentication of XML Documents. In *Proc. of the 8th International ACM Conference on Computer and Communications Security (CCS-8)*, ACM, 136–145, 2001.

[5] Devanbu, P., Gertz, M., Martel, S., Stubblebine S.: Authentic Third-party Data Publication. In *14th IFIP 11.3 Working Conference in Database Security (DBSec'00)*, Kluwer Academic Publishers, 101–112, 2000.

[6] Maruyama, H., Tamura, K., Uramoto, N.: Digest Values for DOM (DOMHASH). RFC2803, www.faqs.ord/rfcs/rfc2803.html, April 2000.

[7] The GiST Indexing Project. gist.cs.berkeley.edu.

[8] Goodrich, M.T., Tamassia, R., Triandopoulos, N., Cohen, R.: Authenticated Data Structures for Graphs and Geometric Searching. In *Topics in Cryptography CT-RSA 2003*, LNCS 2612, 295–213, 2003.

[9] Kwong, A., Gertz, M.: Authentic Publication of XML Document Data. In *Proc. of the 2nd International Conference on Web Information Systems Engineering (WISE)*, IEEE Computer Society, 331–340, 2001.

[10] Merkle, A.C.: A Certified Digital Signature. In *Advances in Cryptology – CRYPTO '89, 9th Annual International Cryptology Conference*, LNCS 435, Springer, 218-238, 1989.

[11] Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.: A General Model for Authentic Data Publication. To appear in Algorithmica (Springer).

[12] Nuckolls, G., Martel, C., Stubblebine, S.: Certifying Data from Multiple Sources. In *Proceedings of the 17th IFIP WG 11.3 Working Conference on Database and Applications Security*, 2003.

[13] Naor, N., Nissim, K.: Certificate Revocation and Certificate Update. In *Proceedings of the 7th USENIX Security Symposium*, 1998.

[14] XML-Signature Syntax and Processing. W3C Recommendation, www.w3c.org/Signature, February 2002.

[15] XML Path Language (XPath) 2.0. W3C Working Draft, www.w3c.org/TR/xpath20/, Nov. 2003

Enabling Secure Data Exchange

Gerome Miklau
University of Washington
gerome@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

1 Introduction

The emergence of diverse networked data sources has created new opportunities for the sharing and exchange of data. In support of this, a fruitful line of research has resulted in distributed data processing and integration systems [19, 17, 29, 30, 3]. However in practice, fear of unauthorized disclosure or malicious tampering requires that data stay safely behind firewalls or remain protected by secure servers. Our goal is to overcome these limitations and enable secure data exchange and sharing in distributed integration scenarios. Such scenarios are characterized by many interacting data sources and many data consumers. Primary sources create and publish data; intermediate sources combine, extract, and modify the data for further dissemination; data consumers query it. This paper describes issues in secure data exchange, and illustrates some solutions proposed in the authors' own work.

The basic requirements of secure data exchange are *confidentiality* and *integrity*. Confidentiality means that unauthorized parties are prevented from *reading* data. In data exchange, confidentiality is provided through encryption and managing keys that allow access. Confidentiality benefits data sources who need to protect data. Integrity (in its basic form) means that unauthorized parties are prevented from *modifying* data. In data exchange, integrity is provided through digital signatures and data certification techniques. Integrity benefits both data sources (who need to make sure data attributed to them is not modified) and data consumers (who need guarantees that the data they use has not been tampered with).

Confidentiality and integrity are distinct goals and the tools for each are different. In particular, techniques for providing confidentiality do not by themselves provide integrity. Participants can guarantee both properties by combining techniques. We describe the basic features of our envisioned framework for secure data exchange below:

The data Base data is annotated with *security metadata*. We use XML data since it is the preferred format for data exchange. For confidentiality, the metadata contains information about access control requirements and encryption algorithm details, while the base data is protected by encryption. For integrity, the security metadata contains evidence of authenticity (in the form of signatures), key references and algorithm descriptions. The resulting data can be exchanged freely, without relying on secure communication channels or secure servers.

Key management Confidentiality of the data depends on the secure distribution of keys to intended recipients, while verification of integrity depends on authentic retrieval of a data source's public keys. From a pessimistic point of view, it may seem we have merely transferred problems of confidentiality and integrity from the data

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

onto the keys. For instance, without a signature the data consumer may doubt the authenticity of the data. With a signature the data consumer can trust the data, but only if she trusts the source providing the public key. But, as many cryptographic protocols have proven, there is practical value in consolidating the challenges of ensuring confidentiality and authenticity on a few keys rather than on the data itself [28]. The techniques used to secure keys (like public key infrastructures and trusted certificate authorities) are definitely relevant to secure data exchange but are out of our scope here. We assume the parties involved employ these techniques and focus instead on the range of challenges that remain.

Processing the data Intermediate sources and data consumers process the data by applying keys to decrypt the data, by verifying integrity of the data by evaluating digital signatures, and by computing query results over base data (although these are not necessarily performed in distinct stages). In addition, sources construct new data with proper confidentiality and integrity properties by encrypting and signing.

In the next section we motivate the goal of secure data exchange with three applications, highlighting the security requirements of each. We discuss confidentiality in Section 3 and integrity in Section 4. We identify research challenges in Section 5 and then conclude.

2 Applications of Secure Data Exchange

Scientific data management, e-business, and personal identity databases are each compelling applications for secure data exchange.

Scientific data exchange As a representative scientific domain we consider the field of molecular biology. From a few primary sources, containing original experimental data, hundreds of secondary biological sources [2] are derived. The secondary sources export views over primary sources and/or other secondary sources, and usually add their own curatorial comments and modifications. These databases are often published on the Web as large XML documents – not stored in proprietary systems or servers that can provide security guarantees. The data consumers are scientists, and a significant fraction of research takes place in so-called "dry" laboratories using data collected and curated by others.

The risk of malicious tampering with the data is usually not the primary concern in this setting. Instead, the main issues are attributing and retaining authorship and avoiding the careless modification of data through integrity controls. In addition, although many of these databases are released to the public, some forms of confidentiality may nevertheless be important in some cases: when scientific data includes fields that identify individuals, when the data owners wish to audit their data usage by requiring keys for access, or when scientists wish to temporarily limit the release of raw experimental data.

E-business data exchange Today's business transactions very often require data exchange across corporate organizational boundaries, between many parties, and are increasingly performed by machines unattended by humans. The data exchanged may contain financial terms, detailed technical data about products, proprietary corporate information, or personal information about customers. Both confidentiality and integrity of e-business data exchange are critical for ensuring business continuity, compliance with regulations, and protection of corporate assets.

It is worth noting that while there are well-developed technologies for authenticated, confidential point-to-point messaging between two counter-parties in a transaction, that authentication usually happens once for all transmitted data. Even if the authentication information is recorded, often it is not associated directly with the data, or cannot be associated with specific parts of the data.

Personal identity databases A large class of databases, which we call "personal identity databases", have in common the fact that they contain personally identifying information about individuals (e.g. census data, medical databases, organizations' member lists, business customer data). Such databases can be viewed as intermediate sources that collect data from primary source individuals who donate their personal data. The secondary sources may disseminate or further integrate this identity data.

For example, individuals present their personal data to a hospital database when admitted. Hospitals add treatment data and send patient records to an insurance company that integrates it with data of patients from other hospitals. Due to privacy regulations, confidentiality of such databases is critical. But integrity is equally important. If an individual applies for insurance coverage, the insurance company will evaluate the cost of insuring the individual based on the data in its database. An individual should have the right to verify the accuracy of that data. This can be accomplished if the data carries integrity metadata, and the insurance company is required to present the data and its evidence of integrity. To continue the example, some of the individual's personal data will be signed by the individual, some will be signed by individual's doctors etc. Regulations need to be in place that make it illegal for the insurance company to base coverage decisions on data that is not checked for integrity.

3 Providing Confidentiality

Confidentiality is an assurance that unauthorized parties are prevented from accessing data. For exchanged data, confidentiality is provided by encryption. A few recent projects [5, 4, 25] have encouraged data sharing and exchange by allowing a single published XML document to ensure confidentiality for authors in the context of a set of data consumers with different access rights. We elaborate below on some distinguishing features of our framework [25], and also mention some of our recent work on information disclosure in data exchange [26].

3.1 Confidentiality for Published Data

We have developed a framework that allows a data owner to restrict access to published XML data through encryption. The XML document is partially encrypted (and sometimes super-encrypted) to support a declared access control policy. Then the data owner publishes it on the Web, and anyone can download it, process it, and/or re-publish it or fragments thereof. The encryption ensures that only users having specific keys can access restricted data. In addition, we enforce *conditional* access to data, granting access to users who can supply certain data values contained in the database. Conceptually this is related to binding pattern limitations on a query interface. This feature may, for example, be used to permit any medical professional who knows a patient's social security number and date of birth to access certain fields of that patient's medical record. The main components of our framework are the following:

1. A query language for expressing policies – The data owner expresses access control policies using extensions to XQuery. For example consider the following policy query:

   ```
   SUFFICIENT
   FOR      $x in /doc/subjects/subject
            $y in /doc/psychs/psych
   WHERE    $x/examining-psych/id = $y/id
   KEY      getKey( $y ) keyChain("psych")
   TARGET   $x
   ```

   It specifies that a psychologist examiner is allowed to see all subjects he examined. A new key is generated by the function `getKey( $y )` for each psychologist `$y` (or retrieved, if it already exists), in the key chain named "`psych`", and that key grants access to all subjects examined by that psychologist. Notice that if a

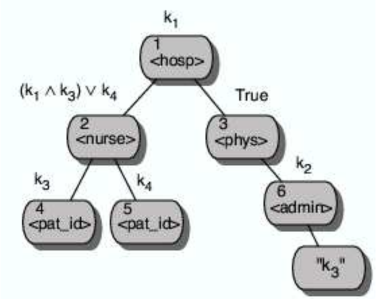

Figure 1: A tree protection.

subject was examined by multiple psychologists[1] then each of them has access to that subject. In addition, other policy queries may impose other restrictions to subject elements, to their sub-elements, or to their parents.

2. A logical model for protecting XML document – We describe a logical model for protecting an XML document tree, in which nodes are labeled with logical expressions over keys. An example is shown in Fig 1. Access to a subtree requires possession of keys that satisfy the logical expression "guarding" the subtree. For example, a user holding the keys k_1 and k_4 will have access to the nodes 1, 2, 3, and 5, and to no other node. If, in addition, the user holds key k_2, then she can unlock node 6, then access its child and acquire key k_3: this in turn gives her access to node 4.

   All policy queries that the data owner specifies are executed over the XML document and result in one logical protection of the XML document. This protection model enables rewriting and optimization at the logical level (to avoid size blow-up due to super-encryption), before the protected tree is represented physically as a partially-encrypted document.

3. Techniques for generating protected XML – Once the logical protection is optimized, it is used to generate a single partially-encrypted XML document that enforces the access control policies. In [25] we validate experimentally the efficiency of generating such XML documents, as well as the positive impact of the logical optimizations and other improvements like compressing data before encrypting.

3.2 Information Disclosure in Data Exchange

A common approach to hiding confidential data in databases is to create views that simply omit the secret items. This is also applied in data sharing. The data owner decides what views to share with each user, then uses physical protection mechanisms to enforce that each user gets only the views he is entitled to. However further dissemination of these views, or collusion between users, may compromise confidentiality. While nothing can be done after the fact, the data owner may wish to understand the amount of confidential information that leaks under different collusion scenarios. To address this problem we have studied in [26] the *query-view security problem*: given a set of views $V_1, \ldots, V_n$ and a secret query S, do the views leak any information about the secret query? To formalize information leakage we have adapted Shannon's definition of perfect secrecy: the query S is secure w.r.t. the views $V_1, \ldots, V_n$ if the probability of an attacker guessing the answer to the secret query remains the same before and after learning the view answers. Among other results, we have shown that it is possible to decide query-view secrecy for the case of conjunctive queries, and that the query complexity is Π_2^p-complete.

[1] This happens when subject has more than one examining-psych sub-elements.

4 Providing Integrity

Integrity, in its most basic form, is an assurance that unauthorized parties are prevented from modifying data. Unauthorized modification of data may be malicious, or simply careless. Integrity is more complex than confidentiality, and we distinguish between *data integrity* and *operation integrity* below. We describe here the integrity problem in data exchange, mention some of the existing techniques, and describe some future directions we hope to pursue in supporting integrity properties.

Data Integrity Data integrity can include one or more of the related properties of *origin authenticity*, *temporal commitment*, *non-repudiation*, and *freshness*. Origin authenticity is an assurance that the data comes from the attributed source (this implies that in addition it has not been modified from its original state). Temporal commitment is an assurance that the data was not modified after the time of receipt of signature. Non-repudiation is an assurance that the source of the data cannot deny authorship, while freshness is an assurance that the data is not a copy of data signed by the source, cached by an unauthorized party, and later presented as authentic. (This is commonly called a replay attack [20].)

Data integrity is provided using digital signatures, usually based on public key cryptography [20]. Any digital signature can provide origin authenticity and temporal commitment, the first two properties above. Additional features can be added to a digital signature to provide properties of non-repudiation and freshness as well. An XML Recommendation [15] describes a schema for signing data and representing signature metadata, and recent work has integrated signatures into XML processing [7, 6].

Query integrity Query integrity is an assurance that a query over a database has been performed accurately, which is a consideration distinct from the integrity of the individual data elements returned by the query. For example, data integrity techniques may make it difficult for the source to falsely introduce unauthorized data, but the source may still omit pieces of data or compute query incorrectly.

Research into consistent query protocols [18, 27, 13, 12] can provide query integrity in some cases. The techniques are based on Merkle trees [21, 22] and allow signing of a database D such that given a query Q and an possible answer x to Q, a verification object can be constructed which proves that $x = Q(D)$. This means that if a database is signed, then for some queries (like simple selections or range queries) it is possible to provide evidence of query integrity, relative to a prior signature over the database. The important point here is that the entire database is signed once, and this single certificate can later be used to verify the integrity of any query. It should be noted that there are limitations to the efficiency and privacy of these techniques – the verification objects can be large, and may reveal data elements that are not in the query output.

4.1 Managing integrity

Consider the simplest data exchange scenario: a transformation is applied to a data source to compute a view, which is exchanged with a partner. The integrity properties of the input must be propagated through the query or view to the output. This is related to some extent with managing data provenance [9, 10, 11, 8] which involves tracing and recording the origin of data and its movement among databases. In our case, however, the "integrity provenance" constitutes hard-to-forge evidence of origin (i.e. digital signatures) rather than a mere claim of origin. In data exchange the transformation querys are repeated and amplified many times. The main types of querys that impact the management of integrity provenance are:

Object fusion We construct a new object by integrating data from two different sources. For example a source such as `dblp` provides bibliographic information about a publication, which we merge with information about the number of citations obtained from, say, `citeseer`. The new object needs to carry integrity provenance that specifies that it was obtained by combining data from both sources.

Choice We wish to construct a new piece of data, based on evidence found at two (or more) different data sources. Each piece of evidence is sufficient by itself to support our new data item. We need to be able to express this as integrity provenance. Notice how this differs from object fusion: there a user needs to trust *both* sources in order to trust the fused object, while here it suffices if she trusts only one source.

Extraction We wish to publish a piece of data that is a fragment of a larger item which has a signature. Here the integrity provenance needs to include the signature on the larger object, and also an indication of how the fragment was extracted. Merkle trees address a special case of extraction, when a single tuple is extracted from a certified database.

Combination A new piece of data is constructed by combining together several data items collected from several sources. In addition to the integrity provenance of each item, now we need to include the signature of the author who performs the combination.

4.2 Querying integrity provenance

Integrity provenance can interact with a query language in different ways. First, the query language may require users to place restrictions on the data items sought, based on their integrity provenance. For example, the user may want to specify that she trusts only sources S_1, S_2, S_3, and ask the system to retrieve only data items that can be fully certified solely by these sources. Second, when the data is transformed or integrated with a query language, the integrity provenance information needs to be propagated in the output data. As suggested above, the meaning of this propagation is not obvious. One possibility is to allow users to specify policies on how the propagation is being performed. Consider the union of non-disjoint data elements. A policy may require each signature to be trusted (corresponding to logical AND), or may simply require the presence of one trusted signature (logical OR), or may always prefer one signature (when available) over another. Third, once a query is evaluated, the user may request a proof of the validity of the answer. This proof may range from a simple enumeration of all signatures used in the data items returned, to a complex expression indentifying all signatures that have been used in different parts of the data source that affected the query's answer.

5 Research Challenges

Data enriched with confidentiality and integrity metadata poses some special challenges, enumerated below.

Query language challenges Physically, the data and the security metadata may often be stored together. But logically, they should be separated. A query language should have abstractions that allow users to refer to the security metadata separately from the data. Users should never have to express queries over the mixed schema. In addition, query answers may need to contain explanations of their result since, for example, the answer to a query could be empty because evaluation on the base data resulted in an empty answer, or because the confidentiality or integrity conditions were not met.

Query processing challenges Managing the metadata together with the data during query processing will pose challenges for the system. Some accesses to the metadata will be cheap, but most often they require expensive decryption or signature verification. The query planner should favor plans that defer these expensive operations as much as possible. There are some connections here to research into query execution with expensive predicates [16], in which the customary pushing of selections is not always beneficial.

Schema challenges The base data exchanged in our scenarios will usually conform to some agreed-upon schema. However, as we have described, the base data may be transformed into partially-encrypted elements, signatures may be "wrapped" around the base data, or both. While XML schema have been defined for encryption [14] and digital signatures [15], the secured data instances will not generally conform to the base data schema or the security metadata schema. This complicates querying and validating of the data, as well as schema evolution.

User assistance and deployment In order to generate data instances with proper security metadata it is essential to have machine assistance. Signatures and encryption keys consist of random strings that cannot be manipulated in any reasonable way by humans. While a scientist may have used a text editor in the past to update a data entry, now that scientist needs more advanced tools to add signatures and perform encryption. Users also need "key chains" to manage keys given to them or granted by them.

Proving security Claims of confidentiality and integrity require formal proofs of security. Since our data instances result from the application and combination of cryptographic primitives, proofs of security are hard to generate. In the case of confidentiality, we described a logical language for protected XML trees and a process for generating encrypted data from the logical model. Ideally, we would like proofs of security to follow from the soundness of a construction in the logical model. Recent work [1, 23] from the cryptographic community addresses precisely this goal and needs to be extended and applied to our setting.

6 Concluding Remarks

Providing confidentiality and integrity is critical to facilitating sharing and exchange of data. Since data is exchanged beyond domains of influence of data authors, we can't depend on secure systems to enforce confidentiality and integrity, but must rely on techniques of cryptography. Yet, while there are many compelling cryptographic primitives available, applying and adapting them to complex data management is a major challenge. For one, in data exchange, we are concerned not with blocks of data or messages treated in isolation, but with databases which are structured collections of elements, to which confidentiality and integrity may apply at various levels of granularity. Second, communication is not point-to-point but involves many parties in a complex network as described in our application scenarios. We have described our view of the key problems, referred to existing work, and proposed future directions. The interested reader may consult [24] for a more detailed version of this paper.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP Int. Conference on Theoretical Computer Science*, Sendai, Japan, 2000.

[2] Andreas D. Baxevanis. Molecular biology database collection. Nucleic Acids Research, available at www3.oup.co.uk/nar/database/, 2003.

[3] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The hyperion project: from data integration to data coordination. *SIGMOD Rec.*, 32(3):53–58, 2003.

[4] E. Bertino, B. Carminati, and E. Ferrari. A temporal key management scheme for secure broadcasting of xml documents. In *Conference on Computer and communications security*, pages 31–40. ACM Press, 2002.

[5] E. Bertino, S. Castano, and E. Ferrari. Securing xml documents with author-x. *IEEE Internet Computing*, 5(3):21–31, 2001.

[6] E. Bertino, G. Mella, G. Correndo, and E. Ferrari. An infrastructure for managing secure update operations on xml data. In *Symposium on Access control models and technologies*, pages 110–122. ACM Press, 2003.

[7] L. Bull, P. Stanski, and D. M. Squire. Content extraction signatures using xml digital signatures and custom transforms on-demand. In *Conference on World Wide Web*, pages 170–177. ACM Press, 2003.

[8] P. Buneman. Curated databases, November 2003. personal communication.

[9] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[10] P. Buneman, S. Khanna, and W. C. Tan. On propagation of deletions and annotations through views. In *PODS '02*, pages 150–158, 2002.

[11] Y. Cui and J. Widom. Practical lineage tracing in data warehouses. In *ICDE*, pages 367–378, 2000.

[12] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. G. Stubblebine. Flexible authentication of xml documents. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 136–145. ACM Press, 2001.

[13] P. T. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In *IFIP Workshop on Database Security*, pages 101–112, 2000.

[14] D. Eastlake and J. Reagle. Xml encryption syntax and processing. http://www.w3.org/TR/xmlenc-core, 3 October 2002. W3C Proposed Recommendation.

[15] D. Eastlake, J. Reagle, and D. Solo. Xml signature syntax and processing. http://www.w3.org/TR/xmldsig-core, February 12 2002. W3C Recommendation.

[16] J. M. Hellerstein and M. Stonebraker. Predicate migration: Optimizing queries with expensive predicates. In *SIGMOD Conference*, pages 267–276, 1993.

[17] Z. G. Ives, A. Y. Levy, J. Madhavan, R. Pottinger, S. Saroiu, I. Tatarinov, S. Betzler, Q. Chen, E. Jaslikowska, J. Su, and W. Yeung. Self-organizing data sharing communities with SAGRES. In *SIGMOD '00*, page 582, 2000.

[18] J. Killian. Efficiently committing to databases. Technical report, NEC Research Institute, February 1998.

[19] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996. VLDB Endowment, Saratoga, Calif.

[20] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[21] R. C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[22] R. C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[23] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.

[24] G. Miklau. Research problems in secure data exchange. University of Washington Technical Report 04-03-01, March 2003. Available at www.cs.washington.edu/homes/gerome.

[25] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, September 2003.

[26] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. to appear SIGMOD '04, June 2004.

[27] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs on a committed database.

[28] B. Schneier. *Applied Cryptography, Second Edition*. John Wiley and Sons, Inc., 1996.

[29] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 1996.

[30] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. L. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Rec.*, 32(3):47–52, 2003.

Enabling Collaborative Administration and Safety Fences: Factored Privileges in SQL Databases

Arnon Rosenthal
The MITRE Corporation
202 Burlington Rd
Bedford, MA, USA
arnie@mitre.org

Edward Sciore
Computer Science Dept
Boston College
sciore@bc.edu

Abstract

An access policy has many aspects, concerning both information and physical execution. Agglomerating them into a single SQL grant makes policies much harder to administer, especially at enterprise scale where administrators need to collaborate. We present a way to specify policies as a conjunction of factors, in a simple, regular way. Each factor decision poses a simple question, and when a circumstance changes, only the relevant factor needs to be revisited. Factors are also help for establishing "safety fences" on an administrator's work, and for separating (global) information privileges to enable more powerful inference rules. To ease integration into existing systems, factor privileges employ the same interfaces and rules as ordinary SQL privileges, rather than multiple new top-level, awkwardly-interacting constructs (such as "autonomy" and "prohibition").

1 Introduction

The SQL access control model was designed in the 1970s for databases to be used within a closed organization. Either the DBA or the data owner could be relied on to deal with all aspects of a permission decision. In contrast, today's enterprise (and virtual enterprise) database has many stakeholders. For example, data is distributed across multiple platforms and networks, each with their own security, performance, and charge-back concerns. DBAs are typically concerned with giving applications the data they want; security officers wish to minimize the privileges granted to ensure confidentiality, integrity and availability; privacy officers want traditional security, but also want to ensure that individuals are guaranteed appropriate controls over data that describes them (e.g., to know about and reply to unfavorable information).

Today, interaction among these administrators occurs informally or via memoranda. The DBMS access-control system neither documents the stakeholders' separate policies nor enforces them. Instead, each grant requires the consideration of all aspects; when circumstances change, all aspects must be reviewed.

Today, administrators who receive *grant option* are not expected to be guided by whim, or to judge each situation from scratch. They should operate within bounds given by organizational policy. Enterprises may give their members guidance (generalized policies and principles) that they use to make specific decisions [Bark03].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Higher administrators may turn the guidance into boundaries (i.e., *safety fences*) that others should not cross. Unfortunately SQL gives no automated support for such fences.

Many authors have sought to meet this pressing need with "negative privileges" (also known as "prohibitions") [BJS99, BJWW02]. Such mechanisms are quite controversial. For systems of moderate size, opponents argue that they require administrators to flip between positive and negative, and provide unpredictable behavior, especially when overrides are needed. Enterprises have further cause for concern. Most proposals include new features that are not orthogonal to existing ones (notably with respect to views and autonomy), and there is no easy way for many administrators to work together in setting the fences.

Our research goal is to extend the SQL privilege model and services in order to better support collaborative, decentralized administration. In this paper we discuss the part of our approach that involves *factoring* privileges into smaller, easier to use granules. We then show how the use of factor granules can solve the above problems.

2 Factor Granules

The decision to grant a privilege (i.e., for an action on an object) is usually the conjunction of several, smaller decisions. We represent the types of decisions as a *factor tree*, where each factor represents an aspect to be considered in a privilege decision. The root of the tree denotes the complete, full privilege; each leaf of the tree denotes an atomic decision; and an intermediate node denotes an aggregate decision (in effect, the conjunction of all its leaves). A *factor granule* is a "partial privilege" associated with a node of the factor tree, and denotes that from that aspect, the privilege is appropriate.

We extend SQL so that granules (and not privileges) are the unit of granting. A user receives a privilege when it obtains granules for each of the privilege's factors. Granules can be granted at any level; granting a non-leaf granule is equivalent to granting all of the granules in its subtree. Granting the root granule is thus equivalent to granting the privilege.

Factor granules can be independently granted, revoked, and inferred on views (just like SQL privileges), but are better units of management and collaboration. The idea is that different administrators will have responsibility over different factors. Obtaining a granule can be thought of as having the appropriate granule administrator "sign off" on the privilege.

Figure 1 depicts an example of a factor tree that an enterprise might adopt.[1] The root has two children. One child heads the subtree of *information factors*, which represent decisions about the information content of the protected object, independent of a particular physical implementation. The other child heads the subtree of *execution factors*, which may represent decisions about the physical object's environment (e.g., should the grantee be running on this machine? on this copy of the data?).

When an object is created, we grant administrative rights on all information factors to the object creator, and execution factors to the DBA of the DBMS in which it is created. This initial split reduces the threat from either of the two insiders. The SQL model (applied to factors) lets them further delegate their rights to administrators for each factor, ideally using a script that runs automatically upon creation.

2.1 Information Factors

The *information content* of the database expresses "external world" facts (e.g., that John Smith receives penicillin), not the contents of individual memory cells on specific machines. Information factors specify whether a privilege is appropriate, independent of any physical representation. Figure 1 depicts two kinds of information factor. An *ordinaryInfo* factor granule is granted by a business expert, and denotes that it is reasonable (from the point of view of the business) for the requestor (user or roles) to access the specified information. OrdinaryInfo

[1] Our full model (forthcoming) allows a different factor tree for each operation, and includes ways to create and delete factors transparent to other administrators.

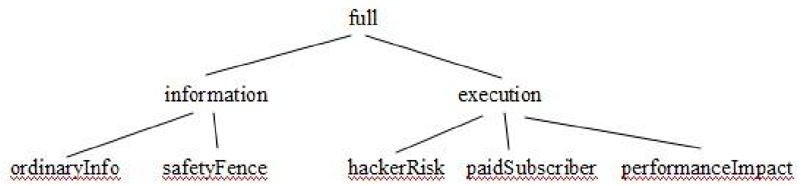

Figure 1

granules should typically be fine-grained, and the right to grant them may be widely delegated among many business experts. SafetyFence granules (Section 2.3) are granted by the security administrator, and denote those users who satisfy general access criteria.

2.2 Execution Factors

All decisions influenced by the physical system are called *execution factors*. Most execution factors permit administrators to control what operations may execute at a particular service locus (e.g., site, machine, DBMS, application server, output device and time, location). Figure 1 depicts three useful kinds of execution factors, which represent the physical-access limitations of allowing only paid subscribers, keeping potential hackers off the machine, and controlling workload. The *paidSubscriber* factor might be managed by a sales administrator, the *hackingRisk factor* by a system manager or security officer, and the *performanceImpact* factor by a DBA.

2.3 Safety Fence Factors

Many business experts and DBAs value functionality over security, and give users what they want [Wi01]. Security-conscious administrators (such as a data provider, local system administrator, or security officer) need to bound the full privileges that a class of users may receive. A *safetyFence* granule can be used for this purpose. The security administrator grants safetyFence granules only to the ”acceptable” user population. Since a user cannot receive a privilege without obtaining all of its granules, the system can guarantee that a user will never receive a privilege without approval from the security administrator. SafetyFence granules protect the system from errors and abuses, and also allow veto power by a site's security administrator. They can also enforce general organizational guidelines, such as “only managers and analysts can see product profitability figures.”

One can insert as many safety fence factors as are convenient, e.g., on either an execution or an information factor. They are similar in spirit to the *prohibitions* of [Bert99], but phrased positively. The advantages (versus prohibitions) are:

- The semantics are inherited from the general concept “factor” - there is no need for administrators to learn new rules. Furthermore, factors behave *exactly* like full privileges in their interactions with other model features (e.g., views, restrictive predicates, ...).
- One can insert as many safety fence factors as there are independent administrators who can veto a privilege. This greatly simplifies coordination. (There is no separate “override” concept. To give an administrator the power to override a safety fence factor, one gives them the Grant privilege on that factor).

We anticipate that SafetyFence grants will often be to high level roles (e.g., “any EMPLOYEE can execute on the internal portal”) and cover large sets of objects. Due to large granularity, we expect that their administration will not be a severe burden.

3 Benefits of Factoring

Factor granules improve privilege administration in several important ways:

- They allow collaboration among administrators.
- They reduce the complexity of administration.
- They allow information privileges to be inferred from view privileges, even if the query does not reference the view.
- They guide security models for virtual and materialized views.
- They fit elegantly info the SQL model.

We discuss each benefit in turn.

3.1 Collaboration

Factor granules make it easy for multiple administrators to collaborate. The factor hierarchy for a privilege explicitly shows the decisions that are required to grant the privilege. The responsibility for each decision can be assigned to an administrator having the appropriate expertise (e.g. business expert, security officer, system administrator, etc.). Each administrator can grant factor granules independently and autonomously, with a privilege being given to a user only if all administrators agree.

Because access-control authority is distributed, there is no need for any one administrator to be the ”super-user”. Instead, each administrator has local authority over his particular factor(s) but no others. For example, many system administrators will no longer require the ability to see the data in order to do their job, and the DBMS will enforce this.

3.2 Less Complexity

Each leaf node in the factor hierarchy denotes a single decision. The question of whether to grant that factor granule is therefore focused and concrete, and its answer typically depends on a narrow slice of technical or domain knowledge. In contrast, a privilege requires consideration of the issues motivating every factor. Consequently, it is easier to determine the correctness and appropriateness of granting a factor granule than it is for granting a privilege. Although there are more granules to administer, each granule is more easily understood, and some factors can be granted to broad roles (e.g. EMPLOYEE) or large sets of objects. The hope is that the number of decisions increases very little, with each decision becoming easier to make and maintain. Testing this hypothesis is an important open problem.

3.3 Implicit Privileges

For information factors, the nature of the computation is not relevant - only the result matters. In [RoSc00], we used this to infer privileges because a computation *could* have been accomplished employing only views for which the user had adequate privileges. The next item further exploits this observation.

3.4 Materialized Views

Explicit management of information factors clarifies inference of privileges on replicated data (and by extension, materialized views). The information privileges are independent of replication (assuming replication delays do not change sensitivity). Autonomy difficulties do not apply to information privileges.[2] The following table summarizes the privileges given to creators of base tables, views, and materialized views. The difference between these three constructs is easily seen when information factors are considered separately from execution factors.

	information	execution
Base Table	all	granted by System Administrator
View	inferred from its definition	inferred from its definition
Materialized View	inferred from its definition	granted by System Administrator

3.5 Elegance

Factor granules behave exactly the same as privileges. Granules apply to both base and derived objects, and have all the administration operations (Grant, Revoke, grant option) identical to privileges. In fact, a privilege is just the special case of a granule for the factor called "full". This elegance means that our model retains the simplicity of SQL, without the need to expose administrators to special cases or entirely-new constructs (e.g., "inference", "autonomy", "privilege override") as is done elsewhere [CVS97, GuOl99, BJWW02]. In addition, previous distributed data security models mostly concern federations. We achieve most of their goals, extending to for *any* derived data, in *any* distributed architecture.

4 Summary and Future Work

In this paper, we proposed splitting privileges into smaller, factor granules. Although this extension to SQL is conceptually simple, it has far-reaching benefits: It simplifies privilege administration, helps administrators to collaborate, and allows policies to be more accurately specified and more easily understood. Moreover, functionality (such as "prohibitions" and "autonomy") that previously had required significant data model extensions, can be expressed straightforwardly using this single mechanism. In work to be reported elsewhere, we integrate factors smoothly into the privilege model for derived data.

There remain many open problems. Theoretical problems include: a formal comparison between prohibition models (with and without override) versus safety fences; should negatives be supported (just) as a user interface option? We need metrics for administration complexity, and performance models that can aid designers. We also need models for specifying and reasoning about privileges at different granularities.

Systems tasks include efficient enforcement of factor assertions at different granularities (perhaps by caching inferred privileges), plus environments for administering access policies in distributed systems. Most enforcement needs to be done in DBMSs (for efficiency and trust), but any large system will include DBMSs that differ in their capabilities. Therefore one may want a "permission manager" that installs privileges for enforcement in the component DBMSs.

Administration methodologies are perhaps the biggest challenge. They need to be flexible and yet easy to learn. Administrators need clear explanations of both the decisions required, and the guidance in making them. Perhaps a fixed (but subset-able) set of factor types can satisfy most needs? How should safety fences be managed, among multiple administrators? What are the needs for switching between positive and negative viewpoints?

[2] [GuOl] also had this insight, but formalized it less thoroughly. Instead of separating information from execution issues, they had "global" privileges. These presumably would be granted only by federation administrators, and (depending on power relationships) could be overridden by local system administrators.

5 References

[Bark03] W. Barker, “Guide for Mapping Types of Information and Information Systems to Security Categories”, National Institute of Standards and Technology, 2003. http://csrc.nist.gov/publications/drafts.html

[BJS99] E. Bertino, S. Jajodia, P. Samarati, “A Flexible Authorization Mechanism for Relational Data Management Systems”, *ACM Trans. Information Systems*, April 1999.

[BJWW02] Claudio Bettini, Sushil Jajodia, Sean Wang, Duminda Wijesekera, “Provisions and obligations in policy rule management and security applications,” *VLDB Conference*, 2002.

[CVS97] S. De Capitani di Vimercati, P. Samarati, “Authorization Specification and Enforcement in Federated Database Systems”, *Journal of Computer Security*, vol. 5, n. 2, 1997, pp. 155-188.

[GuOl99] E Gudes and MS Olivier, “Security Policies in Replicated and Autonomous Databases,” in S. Jajodia (ed), *Database Security XII: Status and Prospects*, 93-107, Kluwer, 1999.

[RoSc00] A. Rosenthal, E. Sciore, “View Security as the Basis for Data Warehouse Security ”, *CAiSE Workshop on Design and Management of Data Warehouses*, Stockholm, 2000.
http://www.mitre.org/tech/itc/staffpages/arnie/index.html

[Wi01] G. Wiederhold: “Collaboration Requirements: A Point of Failure in Protecting Information”; *IEEE Transactions on Systems, Man and Cybernetics*, Vol.31 No.4, July 2001, pages 336-342.

ICDE 2005 TOKYO
International Conference on Data Engineering

The 21st International Conference on Data Engineering (ICDE 2005)

April 5-8, 2005
National Center of Sciences, Tokyo, Japan
Sponsored by
The IEEE Computer Society
The Database Society of Japan (DBSJ)
Information Processing Society of Japan (IPSJ) (pending final approval)
The Institute of Electronics, Information and Communication Engineers (IEICE) (pending final approval)
http://icde2005.is.tsukuba.ac.jp/ http://icde2005.naist.jp/ (mirror)

Call for Papers

Scope

Data Engineering deals with the use of engineering techniques and methodologies in the design, development and assessment of information systems for different computing platforms and application environments. The ICDE 2005 International Conference on Data Engineering provides a premier forum for:

- sharing research solutions to problems of today's information society
- exposing practicing engineers to evolving research, tools, and practices and providing them with an early opportunity to evaluate these
- raising awareness in the research community of the problems of practical applications of data engineering
- promoting the exchange of data engineering technologies and experience among researchers and practicing engineers
- identifying new issues and directions for future research and development work.

Welcome to Tokyo

The conference will be held in Tokyo, the capital city of Japan. Tokyo is also the most populous metropolitan area in the world. From its establishment in 1603 to the end of the 19th century, Tokyo prospered as a castle town called Edo. Tokyo still remains an exciting and attractive city that is constantly developing. Yet despite its complex urban landscape and impressive architecture, this city abounds with parks and gardens. Particularly in this season, blooming cherry blossoms will welcome you to Tokyo. Well-known sites close to Tokyo include the Great Buddha statue at Kamakura and scenic Mt. Fuji.

Topics of Interests

ICDE 2005 invites research submissions on all topics related to data engineering, including but not limited to those listed below:

- Database System Internals and Performance
- Query Processing and Optimization
- Data Warehouse, OLAP, and Statistical Databases
- Mining Data, Text and Web
- Semi-structured Data and XML
- Middleware, Web Services, and Workflow
- Heterogeneity, Semantic Web, and Metadata
- Privacy and Security
- Stream Processing, Continuous Queries, and Sensor Databases
- Database Applications and Experiences
- Temporal, Spatial, Scientific, and Biological Databases
- Distributed, Parallel, Deep Web, and P2P Databases
- Data Management for Ubiquitous and Mobile Computing

Conference Officers

Honorary Chair: The Late Yahiko Kambayashi (Kyoto University, Japan)

Organizing Committee Chair: Yoshifumi Masunaga (Ochanomizu University, Japan)

General Chairs: Rakesh Agrawal (IBM Almaden Research Center, USA)
Masaru Kitsuregawa (University of Tokyo, Japan)

Program Chairs: Karl Aberer (EPF Lausanne, Switzerland)
Michael Franklin (UC Berkeley, USA)
Shojiro Nishio (Osaka University, Japan)

Executive Committee Chair: Jun Adachi (National Institute of Informatics, Japan)

Panel Chairs: Umeshwar Dayal (HP Labs, USA)
Hongjun Lu (Hong Kong Univ. of Science & Technology, China)
Hans-Jörg Schek (UMIT, Austria, and ETH Zurich, Switzerland)

Tutorial Chairs: Michael J. Carey (BEA Systems, USA)
Stefano Ceri (Politecnico di Milano, Italy)
Kyu-Young Whang (KAIST, Korea)

Industrial Chairs: Anand Deshpande (Persistent Systems, India)
Anant Jhingran (IBM Silicon Valley Lab, USA)
Eric Simon (Medience, France)

Demo Chair: Ling Liu (Georgia Institute of Technology, USA)

Local Arrangement Chair: Haruo Yokota (Tokyo Institute of Technology, Japan)

Workshop Chair: Masatoshi Yoshikawa (Nagoya University, Japan)

Proceedings Chair: Motomichi Toyama (Keio University, Japan)

Publicity Chair: Hiroyuki Kitagawa (University of Tsukuba, Japan)

DBSJ Liaison: Hiroshi Ishikawa (Tokyo Metropolitan University, Japan)

Treasurer: Miyuki Nakano (University of Tokyo, Japan)

Program Committee Area Chairs

Anastassia Ailamaki (Carnegie Mellon University, USA)
Gustavo Alonso (ETH Zurich, Switzerland)
Phillip Gibbons (Intel Research, USA)
Takahiro Hara (Osaka University, Japan)
Jayant R. Haritsa (IISc Bangalore, India)
Alfons Kemper (University of Passau, Germany)
Sharad Mehrotra (UC Irvine, USA)
Wolfgang Nejdl (University of Hannover, Germany)
Jignesh M. Patel (University of Michigan, USA)
Evaggelia Pitoura (University of Ioannina, Greece)
Jayavel Shanmugasundaram (Cornell University, USA)
Kyuseok Shim (Seoul National University, Korea)
Kian-Lee Tan (National University of Singapore, Singapore)

Important Dates

Abstract deadline:	June 24(THU), 2004 2pm PST
Submission deadline:	July 1(THU), 2004 2pm PST
Notification date:	September 13(MON), 2004

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398