A POLICY-BASED PRIVACY STORAGE APPROACH

Julien Nowalczyk and Frédérique Tastet-Cherel
Thales Communications, France

Keywords: Privacy, Security, Policies, Management, Database.

Abstract: Current information systems rely on complex databases, where information is provided and retrieved by many actors interacting with many items during many operations. Data access and transfer must be protected from external access as well as from any unauthorized internal access. We propose a light and scalable mechanism to secure the storage of such a database. First, we define storage and access control policies to record information. Then we specify a secured storage proxy that applies these policies. If required, sensitive data are transferred and stored encrypted according to the previous decided security policy.

1 INTRODUCTION

Companies databases gather an increasing amount of sensitive data coming from multiple sources. All these data can be finally stored in a back-end database system held by a third party responsible for preserving the privacy and the security of all these aggregated data. Within current database systems, the data owner usually delegates the privacy of its data to the service provider.

Most of databases integrate more and more complex security functions dealing with network security as well as data security. Yet, all these security mechanisms, such as authentication, confidentiality, integrity or access control, only relate to an organizational control over the information, and not to a personal one.

The current approaches do not directly provide privacy, i.e. freedom of choice, control and self-determination. For example, data encryption functions proposed by the most of databases allow to make the data unintelligible of an external attacker who successes to gain access to the database. But, a database administrator (DBA) could gain a complete access to these data by getting the encryption key, often stored in the database. Another measure aiming at ensuring data privacy is to maintain historical traces of data and operations, but this can only be considered as intrusion detection measure. Finally, all the proposed mechanisms often remain obsolete in comparison to the current privacy threat.

In the SMMART project (SMMART), a huge amount of data are collected through sensors and RFID networks, and finally gathered in a central back-end database ready to be processed afterwards. All these data belong to different companies and require security mechanisms that guarantee their own privacy levels. In SMMART, the service provider not only proposes a storage service, but especially a set of high-level services in order to manage the logistics supply chain (SMMART).

In order to fulfil these security requirements, the proposed policy-based privacy storage mechanism must be enough configurable to allow user to define its own data privacy policies, with no impact on the overall system.

Moreover, data security is commonly based on the definition of a secret in order to apply cryptographic functions. The main issue consists in managing this secret. In SMMART, we propose to externalize this secret by delegating it to the data owner who has a total control over its data privacy.

In this paper, we first review the existing in database security and privacy by considering what they could bring to the SMMART approach. Finally, we describe our security mechanism proposal.

Nowalczyk J. and Tastet-Cherel F. (2007).
A POLICY-BASED PRIVACY STORAGE APPROACH.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems*, pages 605-608

2 DATABASE PRIVACY ISSUES

Database privacy is an issue of a significant interest. Many works, particularly on privacy of data transmission exist. This paper only focuses on privacy of stored data.
Access control is a widespread and mandatory security mechanism to prevent access to the database from outsiders. (Oracle Database) presents a fine-grained access control (FGAC) mechanism for databases. FGAC allows to implement security policies with functions and then to use those security policies to implement row-level security on tables and views. Although access control mechanisms become more refined, they are not sufficient to guarantee data privacy from an insider. Another approach in database privacy is multi-level security (Jajodia and Sandhu). Whereas access control approach usually offers a per-table granularity, both data and entities are assigned security levels in multi-level approach. Multilevel security extends access control, but it does not provide the necessary control to the data owner over its data.
Thus, some paper opt for an individual centric privacy approach. (Marchiori) proposes a solution allowing corporations to define their own privacy policies. The organization is here considered as trustful. Therefore, considering as uncomfortable the philosophy of trusting corporations to protect privacy, (Aggarwal et al.) exports the privacy policies from the organization to the data owner, and seeks to enable the data owner to retain control over its personal information even after its release to an organization.
(Bawa et al.) adopts a completely different approach by splitting the information over several *providers*. A index allows to reconstruct the original information. The problem comes down to provide privacy-preserving search over distributed access controlled content.
Other research papers introduce statistical databases (Adam and Worthman). The main idea is to enable queries on aggregate of information without revealing individual records. This interesting approach does not fit our requirement because we need to minimize impact on existing database system.
Finally, (Adam and Worthman) explores data privacy issues in the "databases as service" (DAS) paradigm. DAS systems provide its customers seamless mechanisms to create, store, and access their databases at the host site. Data privacy is ensured by data storage in encrypted form. The main challenge, on top of the key management, is the execution of SQL queries by the service provider over encrypted database without decrypting the data. This innovative approach completely exports the database query processor to the client side.

3 SMMART SECURE STORAGE APPROACH

3.1 Overview

Providing a privacy policy-based storage mechanism is a real challenge because of the complexity of relational database and the major issue of executing SQL queries over encrypted data. This section presents the SMMART secure storage approach.
At the service level, two main entities are involved: the service provider, i.e. the company that hosts the SMMART system and all the associated services, and the client, which consumes the service. At the data level, the client uses several devices to collect its data. Vehicles of client company embed RFID and sensors networks in order to collect information, and a data concentrator unit (DCU) to gather it. Then, information is accessible either through the mobile device or through the back-end database, which synchronize with the DCU.
The figure 1 presents the overall SMMART system.

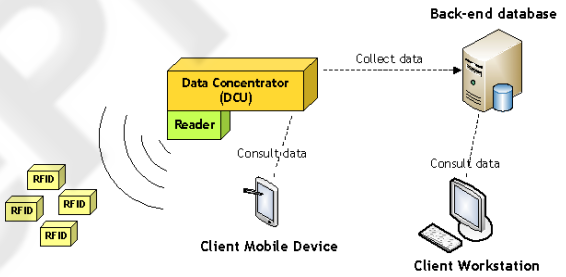

Figure 1: Simplified view of the SMMART system.

Finally, the goal of the privacy policy-based storage mechanism lies in providing to the client the mean to control its own data privacy. To address this issue the client defines privacy policies to attribute a privacy level to its data (section 2.2). According to this privacy level, the client's data are finally stored in the service provider backend database. Moreover, client manages its own encryption material, that means database encryption is based on user-supplied keys (section 2.3).

3.2 Policy-based Privacy

Policies allow the client to fully control its own data privacy. Figure 2 gives a very simple example of

privacy policy. .In the following example, conditions rely on the actor and the data type. The action enables (or not) data privacy by encrypting (or not) the data in the back-end database system.

#Rule	Actor	Data type	Privacy	*Action*
1	John (maintenance)	Vehicle mission datapack.	YES	*AES-256 bits*
2	Bob (human resources)	global configuration data	NO	*no encryption.*
...	...	...	...	...

Figure 2: Privacy policy example.

The *privacy policy management* module manages the privacy policies (component architecture presented in Figure 3). For each query, the *query management* module consults and applies the policies defined in the *privacy policy management* module. Finally, the main role of policy based system is finally to define the data encryption granularity in the database.

3.3 Component Architecture

The challenge in designing such a component is to allow the client, which manages all the security material, to execute SQL queries over an encrypted database. This is the role of the *Query management* module (Figure 3).

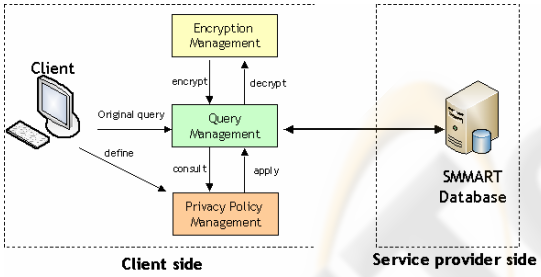

Figure 3: Architecture overview.

We illustrate this process on a specific SMMART table, which maintains information about DCU management, defining by the following statement:

```
CREATE TABLE smmart_table
      (     dcu_id integer PRIMARY KEY
            date data
            location varchar(25)
            release integer PRIVATE
      );
```

The **PRIVATE** attribute means the *release* column in the `smmart_table` intervenes in client privacy policies. The elements of that column are encrypted regarding to the privacy policy. One encryption key is defined per column, and the *Encryption Management* module maintains all these keys. That means, the module knows all the column marked as private, and all the associated cryptographic keys. To insert an new record, the client executes the following request:

```
INSERT INTO smmart_table_ VALUES (101,
2007/01/23, 'tarnos-fr', 46785123);
```

The SQL query is fully transparent for the client. The *Query management* module receives this query and interacts with the *Privacy Policies Management* module in order to consult and apply the privacy policies defined by the client. Then, it calls the *encryption* module in order to encrypt the right fields (the *release* field in the previous example)of the SQL query. The query is finally sent to the server, in the following way:

```
INSERT INTO smmart_table VALUES (101,
2007/01/23, 'tarnos-fr',ee2cbd4);
```

The back-end database stores the release number encrypted. When the client needs to access the previous release number, it issues the following request:

```
SELECT release FROM smmart_table
      WHERE dcu_id = 101;
```

The request is simply forwarded to the back-end database. When receiving the response, the client needs to decrypt the release number, which is stored encrypted in the database, with the encryption key of the release column. For that, the *Query Management* module asks the *Encryption Management* module to decrypt the release field. All the process is also transparent to the final user.

Finally, encryption process needs an entity that provides the cryptographic keys. We propose to fully delegated the key management to the client. The key distribution is centralized in order to allow group encryption. Thus, Alice and Bob can work on the same column with the same encryption keys, which are given by the key distribution module. Moreover the key distribution can rely on a certificate-based infrastructure to authenticate user during the distribution.

4 CONCLUSION

This paper presents an approach to ensure data privacy within the scope of SMMART, but many challenges still remains. This first one is the encryption cost. A trade-off needs to be made between encryption techniques (software/hardware encryption) and data granularity for encryption. The introduction of privacy policies allows to set and optimize this data granularity.

The second challenge is the execution of SQL query over encrypted data and the impact of it on the storage model. In SMMART, we try to keep the original service provider data model, by restricting SQL queries to basic selection queries. But the execution of more complex queries (with the join, grouping ,aggregation or sorting operators) raises a lot of new issues, and may imply afterwards the definition of new storage model.

Then, key management plays an important role by enabling the encryption process. The exportation of the key management to the client side increases the data privacy level.

Finally, this paper insists on the necessity for the user to keep control over its personal information in order to enable data privacy. Database infrastructures need to be improved to allow the definition of such security architectures.

REFERENCES

SMMART project website. http://smmart.iao.fraunhofer.de/plone

Oracle Database 10g Family Product www.oracle.com/technology/products/database/oracle 10g/pdf/twp_general_10gdb_product_family .pdf

S. Jajodia and R. Sandhu, Toward a multilevel secure relational data model.

M. Marchiori (Ed.), The Platform for Privacy Preferences 1.0 (P3P1.0) Specification.

G. Aggarwal, M. Bawa, H. Garcia-Molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, Y. Xu, Vision Paper: Enabling Privacy for the Paranoids.

M. Bawa, R. J. Bayardo Jr.,R. Agrawal, Privacy-Preserving Indexing of Documents on the Network.

N. R. Adam and J. C. Worthman, Security-Control Methods for Statistical Databases: A Comparative Study

H. Hacýgümüs, B. Iyer, S. Mehrotra, Providing Database as a Service.