Chip-Secured Data Access: Reconciling Access Rights with Data Encryption

Luc Bouganim* François Dang Ngoc** Philippe Pucheral*,** Lilan Wu**

* INRIA Rocquencourt
France
<Firstname.Lastname>@inria.fr

** PRISM Laboratory
78035 Versailles – France
<Firstname.Lastname>@prism.uvsq.fr

1. Introduction

The democratization of ubiquitous computing (access data anywhere, anytime, anyhow), the increasing connection of corporate databases to the Internet and the today's natural resort to Web hosting companies and Database Service Providers strongly emphasize the need for data confidentiality. Users have no other choice than trusting Web companies arguing that their systems are fully secured and their employees are beyond any suspicion [AKS02]. However, database attacks are more and more frequent (their cost is estimated to more than $100 billion per year) and 45% of the attacks are conducted by insiders [FBI02]. Therefore, no one can be fully confident on an invisible DataBase Administrator (DBA) administering confidential data.

Traditional database security policies, like user authentication, communication encryption and server-enforced access controls [BPS96] are inoperative against insider attacks. Several attempts have been made to strengthen server-based security approaches thanks to database encryption [Ora02, Mat00, HeW01]. However, as Oracle confesses, server encryption is not the expected "armor plating" because the DBA (or an intruder usurping her identity) has enough privilege to tamper the encryption mechanism and get the clear-text data.

Client-based security approaches have been recently investigated. They still rely on database encryption, but encryption and decryption occur only on the client side to prevent any disclosure of clear-text data on the server. Storage Service Providers proposing encrypted backups for personal data [Sky02] are crude representative of the client-based security approach. The management of SQL queries over encrypted data complements well this approach [HIL02]. These solutions provide a convincing way to store and query safely personal data on untrusted servers. However, sharing data among several users is not addressed. Actually, users willing to share data have to share the same encryption keys and then inherit from the same access rights on the data.

In a recent paper [BoP02], we precisely addressed this sharing issue. We proposed a solution called C-SDA (Chip-Secured Data Access), which allows querying encrypted data while controlling personal privileges. C-SDA is a client-based security component acting as an incorruptible mediator between a client and an encrypted database. This component is embedded into a smartcard to prevent any tampering to occur on the client side. This cooperation of hardware and software security components constitutes a strong guarantee against attacks and allows to reestablish the orthogonality between access right management and data encryption.

A full-fledged prototype of C-SDA has been developed with the support of the French ANVAR agency (Agence Nationale pour la VAlorisation de la Recherche). This prototype runs on an advanced JavaCard platform provided by Schlumberger. The objective of the C-SDA prototype demonstration is twofold:

- Validate the design of C-SDA by building a real-case application and showing the benefits of the approach.
- Validate the techniques C-SDA relies on by showing that they match the smartcard's hardware constraints and the user's response time expectation.

This paper is organized as follows. Section 2 introduces the hosted corporate database application which we will use for the demonstration. Section 3 presents the C-SDA design and implementation choices needed to understand the value of the demonstration. Section 4 presents the demonstration platform and the way we plan to validate our techniques.

2. The Corporate Database Demonstrator

The demonstrator selected to illustrate the properties of C-SDA relates to a corporate database hosted by a Database Service Provider (DSP). This demonstrator is representative of a growing range of real-case applications. Indeed, small businesses are today prompted to delegate part of their information system to Web-hosting companies or DSP that guarantee data resiliency, consistency and high availability [eCr02,Qck02]. Undoubtedly, the resort to a DSP is today the most cost-effective solution to make the corporate database of a small business available to its traveling salesmen and to its potential partners.

Most DSP provide wizards to create in minutes predesigned or customized business-oriented shared databases. In the same spirit, we use for our demonstrator the well-known TPC-H database schema [TPC02]. To illustrate the effectiveness of our approach, we consider different classes of users sharing the corporate database with distinct privileges. Each traveling salesman has access to all information's regarding her own clients (e.g., identity, address, orders), in a way similar to a *virtual private database* [Ora00]. Each supplier of the small business is granted the right to consult only the total amount of orders related to the products she supplies (so that she can forecast future delivery). All sensitive data (e.g., customers' information, orders, traded prices) is encrypted to prevent them from any disclosure on the server. The privileges of each user is recorded on her own smartcard and refreshed by a transparent and safe mechanism.

3. C-SDA Design and Implementation

This section recalls from [BoP02] the foundation of C-SDA as well as important technical considerations that are required to weight up the value of the demonstration.

3.1. The Data Confidentiality Problem

An in-depth analysis of the respective limitations of both server-based and client-based security approaches led us to characterize the *data confidentiality problem* we are addressing by the following dimensions [BoP02].

- *Confidentiality enforcement*: data confidentiality must be guaranteed against attacks conducted by intruders and DBA (or System Administrator). This precludes server-based solutions since they are inoperative against administrator attacks [Ora00].
- *Sharing capacity*: data may be shared among multiple users having different privileges. This precludes client-based solutions where data sharing is not supported or is implemented by means of encryption keys sharing [HIL02].
- *Storage capacity*: the system must not limit the volume nor the cardinality of the database. This precludes solutions where the whole database is hosted in a secured device (e.g., secured personal folders on smartcards [PBV01]).
- *Query capacity*: any data, whatever its granularity, may be queried through a predicate-based language (typically SQL). This precludes solutions restricted to encrypted backups [Sky02].

Before discussing the technical details of C-SDA, let us outline the way C-SDA tackles the data confidentiality problem. Roughly speaking, C-SDA is a smartcard oriented client-based security approach. The benefit provided by smartcards in the approach is essential. Indeed, smartcards are extremely difficult to tamper [ScS99] and they are now powerful enough to execute complex applications developed in high-level languages like JavaCard [Sun99]. Thus, the principle of C-SDA consists in coupling a DBMS engine embedded in a smartcard with a server hosting an encrypted database. By this way, C-SDA builds a sphere of confidentiality encompassing the smartcard DBMS, the server and the communication channel linking them. This principle is illustrated in Figure 1.

The smartcard DBMS manages access rights and views (i.e., access rights are defined on SQL views), query evaluation and encryption/decryption. When the user issues a query, the smartcard DBMS first checks the user's access rights and, in the positive case, gets the data from

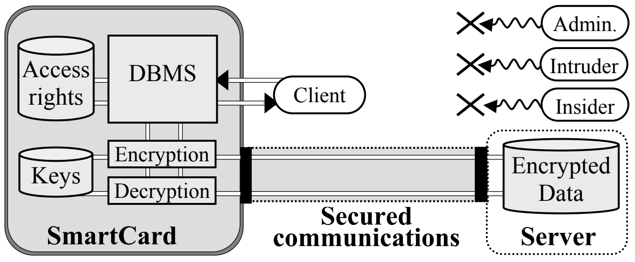


Figure 1: *C-SDA sphere of confidentiality*

the server, decrypts it, executes the query and delivers the result to the terminal. Thus, decryption and access right management are confined in the smartcard and cannot be tampered by the DBA nor by the client.

However, one may wonder whether the smartcard DBMS can conciliate complex queries, large volumes of data and performance, considering the inherent hardware constraints of a smartcard. The next sections recall the smartcard constraints of interest and then investigate the query processing issue.

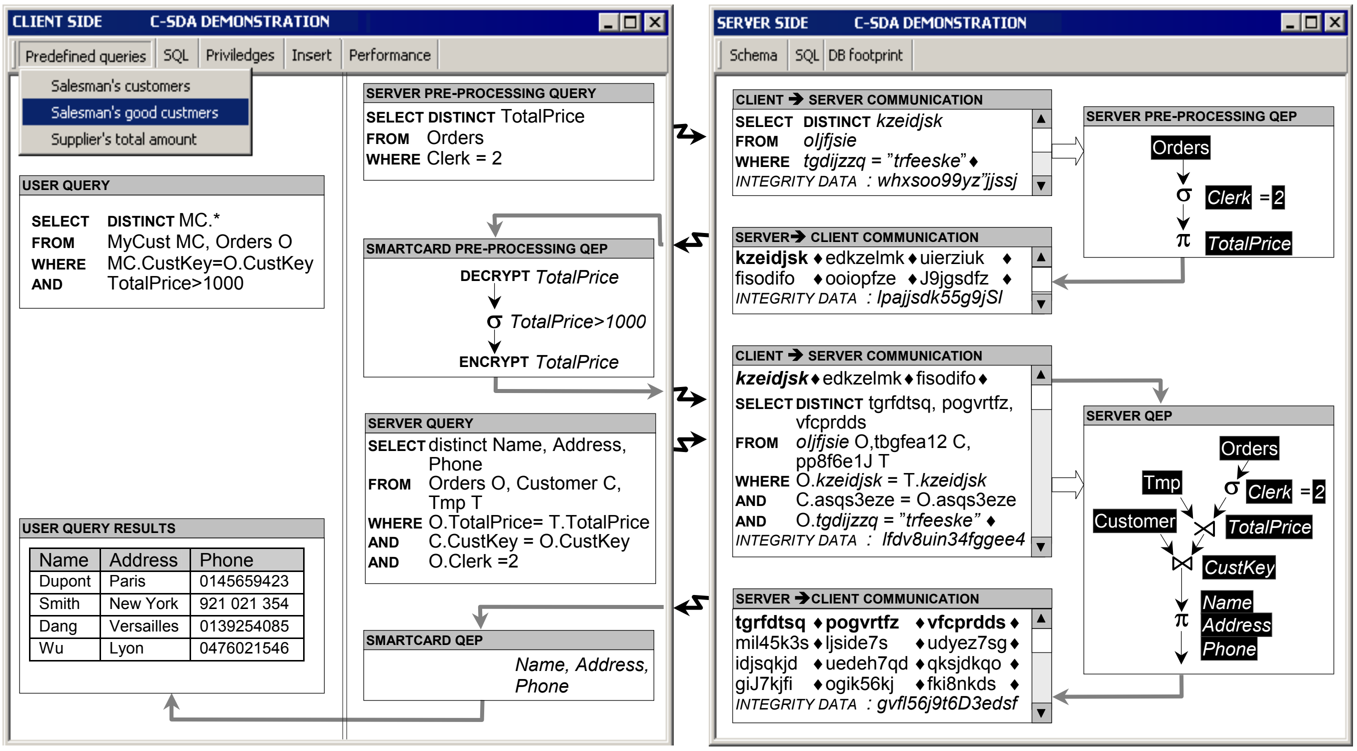
3.2. Smartcard constraints

Advanced smartcards include in a monolithic chip, a 32 bits RISC CPU, memory modules (of about 96 KB of ROM, 4 KB of RAM and 128 KB of EEPROM), a serial I/O channel (current bandwidth is around 9.6Kbps but the ISO standard allows up to 100Kbps) and security components preventing tampering [ISO98]. With respect to our study, the main constraints of smartcards are: (i) the extremely reduced size of the RAM (actually, less than 1KB of RAM is left to the application), (ii) the very slow write time in EEPROM (from 1 to 5 ms/word), and (iii) the limited communication bandwidth. On the other hand, smartcards benefit from a very high security level and from a very powerful CPU with respect to the other resources. According to the chip manufacturers the current trends in hardware advances are on augmenting the CPU power to increase the speed of cipher algorithms and on augmenting the communication bandwidth [Tua99]. These trends are partly explained by market perspectives on delivering protected multimedia flows [Sma02].

3.3. Query evaluation principle

A naive interpretation of the C-SDA architecture depicted in Figure 1 would be to consider the server as a simple encrypted repository for the smartcard DBMS. Obviously, such architecture would suffer from disastrous performance in terms of communication, I/O and local computation on the smartcard. Thus, new query evaluation strategies must be devised that better exploit the computational resources available on the server, and on the terminal, without sacrificing confidentiality. This leads to split a query Q into a composition of the form $Q_t \circ Q_c \circ Q_s$, as follows[1]:

[1] for the sake of simplicity, let assume Q be an unnested SQL query.



USER QUERY

```
SELECT   DISTINCT MC.*
FROM     MyCust MC, Orders O
WHERE    MC.CustKey=O.CustKey
AND      TotalPrice>1000
```

SERVER PRE-PROCESSING QUERY

```
SELECT DISTINCT TotalPrice
FROM   Orders
WHERE  Clerk = 2
```

SMARTCARD PRE-PROCESSING QEP

```
DECRYPT TotalPrice
σ TotalPrice>1000
ENCRYPT TotalPrice
```

SERVER QUERY

```
SELECT distinct Name, Address,
       Phone
FROM   Orders O, Customer C,
       Tmp T
WHERE  O.TotalPrice= T.TotalPrice
AND    C.CustKey = O.CustKey
AND    O.Clerk =2
```

SMARTCARD QEP

Name, Address, Phone

USER QUERY RESULTS

Name	Address	Phone
Dupont	Paris	0145659423
Smith	New York	921 021 354
Dang	Versailles	0139254085
Wu	Lyon	0476021546

CLIENT → SERVER COMMUNICATION

```
SELECT  DISTINCT kzeidjsk
FROM    oljfjsie
WHERE   tgdijzzq = "trfeeske"♦
INTEGRITY DATA  : whxsoo99yz"jjssj
```

SERVER→ CLIENT COMMUNICATION

```
kzeidjsk ♦edkzelmk♦uierziuk  ♦
fisodifo  ♦ooiopfze ♦J9jgsdfz ♦
INTEGRITY DATA  : lpajjsdk55g9jSl
```

CLIENT → SERVER COMMUNICATION

```
kzeidjsk♦edkzelmk♦fisodifo♦
SELECT DISTINCT tgrfdtsq, pogvrtfz,
       vfcprdds
FROM   oljfjsie O,tbgfea12 C,
       pp8f6e1J T
WHERE  O.kzeidjsk = T.kzeidjsk
AND    C.asqs3eze = O.asqs3eze
AND    O.tgdijzzq = "trfeeske" ♦
INTEGRITY DATA  :  lfdv8uin34fggee4
```

SERVER →CLIENT COMMUNICATION

```
tgrfdtsq ♦pogvrtfz  ♦vfcprdds♦
mil45k3s♦ljside7s   ♦udyez7sg♦
idjsqkjd  ♦uedeh7qd ♦qksjdkqo ♦
giJ7kjfi  ♦ogik56kj  ♦fki8nkds ♦
INTEGRITY DATA  : gvfl56j9t6D3edsf
```

Figure 2: *Demonstration graphical interface*

- *Server subquery (Q_s):* to gain performance, any predicate that can be evaluated on the encrypted form of the data must be pushed down to the server. Therefore, the scope of Q_s is determined by the data encryption policy. Let us consider below that the encryption algorithm E preserves the following property: $\forall d_i,d_j$, $E(d_i) = E(d_j) \Leftrightarrow d_i = d_j$. From this assumption, we infer that equi-selection, equi-join and group by predicates are part of Q_s.
- *Smartcard subquery (Q_c):* Q_c filters the result of Q_s to evaluate all predicates that cannot be pushed down to Q_s and that cannot be delegated to the terminal (the data flow resulting from Q_s may go beyond the user's access rights). Under the preceding assumption, inequi-selections, inequi-joins and aggregation have all to be evaluated on the smartcard.
- *Terminal subquery (Q_t):* Q_t is restricted to the part of the query for which the evaluation cannot hurt confidentiality, namely the result presentation (Order by clause).

Obviously, other encryption policies, like the one proposed in [HIL02], will lead to different query decompositions. The more opaque the encryption policy, the less work in Q_s and then the less performance. The challenge is that Q_c must accommodate the smartcard's hardware constraints, whatever its complexity. To this end, the evaluation of Q_c must preclude the generation of any intermediate results since: (i) the RAM capacity cannot accommodate them, (ii) RAM cannot overflow into EEPROM due to the dramatic cost of EEPROM writes and (iii) intermediate results cannot be externalized to the terminal without hurting confidentiality. In [BoP02], we proposed an algorithm that evaluates Q_c in a pure pipeline fashion, consuming one tuple at a time from Q_s and requiring a single buffer to cache the tuple of Q_c under construction. We shown that the computation of Q_c is not CPU bound (powerful processor, low algorithm complexity) nor memory bound (one tuple at a time) but communication bandwidth bound. This led us to devise new optimization techniques to tackle the situations where the ratio $|Q_c|/|Q_s|$ is low, where $|Q|$ denotes the cardinality of Q's result.

To illustrate this situation, let us consider a query retrieving the *Customers* having placed an *Order* with a *TotalPrice* greater than a given value and let assume that only 1% of Order tuples satisfy this selection criteria. This means that 99% of the tuples resulting from Q_s (Customers $\bowtie$ Order) and sent to the smartcard are irrelevant, generating a bottleneck on the smartcard communication channel. The solution proposed relies on a multi-stage cooperation between the smartcard and the server. For each inequality predicate, the smartcard does the following pre-processing. As pictured in Figure 2, the smartcard gets from the server the collection of encrypted values on which the inequality predicate applies, decrypts them, evaluates the predicate and sends back the matching values in their encrypted form to the server. On the server side, this result is integrated in the initial query thanks to a semi-join operator.

4. C-SDA Demonstration

In this section, we present our demonstration platform and describe how we will demonstrate the principles of C-SDA and its performance, through a scenario illustrating a corporate database hosted by a DSP. To make the demonstration user-friendly and easy to follow, we use graphical tools that help understand the behavior of C-SDA.

4.1. Demonstration platform

The demonstration platform includes the C-SDA prototype, a JDBC driver, a traditional database server materializing the services provided by the DSP, and two graphical interfaces devoted to the client and the server (see Figure 2). Our C-SDA prototype is written in JavaCard 2.1 and runs on a smartcard platform provided by Schlumberger (4KB of RAM, 32 KB of EEPROM).

The client graphical interface is divided in two windows. The first window is used to issue SQL queries and visualize the resulting tuples. The second window shows the run time query decomposition performed by the smartcard DBMS. On the server side, the graphical interface displays the query executed on the encrypted data, both in its SQL and operator tree representation. To ease the understanding, names and literals appear in clear-text in the operator tree while they are actually encrypted. The encoding of the communication flow is also pictured, highlighting integrity data required to secure the communication.

4.2. Demonstrating the corporate DB application

Our demonstration consists in various scenarios based on a TPC-H like corporate database assumed to be hosted by a DSP. After explaining how data are encrypted in the database, we will demonstrate the ability of C-SDA to manage powerful and flexible access rights on encrypted data. To this end, we will exercise the application with different smartcards representing different users having their own privileges on the database (e.g., salesmen, suppliers). Among others, we will grant a *Select* privilege to a user on an aggregate value without granting her the right to see the elementary data participating in this aggregation. We will also show that two different salesmen can have a restricted access to their own customers while some customers are shared by both salesmen. While these situations are quite common in traditional databases, they are precluded in other client-based security approaches.

Then, we will explain how the persistent metadata hosted by a smartcard are automatically refreshed when the cardholder's privileges are updated. To this end, we will successively add and remove a right to a user, running a query after each modification to illustrate the effectiveness of the update mechanism.

Finally, attacks on the communication channel between the smartcard and the DSP server will be simulated to demonstrate that data cannot be modified nor disclosed by a third-party.

4.3. Demonstrating the C-SDA performance

The critical part of the C-SDA query execution model is on evaluating subquery Q_c. As discussed in Section 3.3, the computation of Q_c is not CPU bound nor memory bound but communication bandwidth bound. First, this statement will be illustrated by executing different types of SQL queries involving equality and inequality predicates both on encrypted and clear-text attributes (only the sensitive data are encrypted in the database).

Then, the benefit of the pre-processing optimization sketched in Section 3.3 will be assessed. To this end, a query involving a selective inequality predicate will be processed with and without pre-processing optimizations, demonstrating the dramatic performance improvement allowed by pre-processing.

References

[AKS02] A. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Hippocratic Databases", *Int. Conf. on Very Large Data Bases*, 2002.

[BoP02] L. Bouganim, P. Pucheral, "Chip-Secured Data Access: Confidential Data on Untrusted Servers", *Int. Conf. on Very Large Data Bases*, 2002.

[BPS96] A. Baraani, J. Pieprzyk, R. Safavi-Naini "Security In Databases: A Survey Study", 1996.

[eCr02] The eCriteria Database Service Provider http://www.ecriteria.net/

[FBI02] Computer Security Institute, "CSI/FBI Computer Crime and Security Survey" http://www.gocsi.com/forms/fbi/pdf.html

[HeW01] J. He, M. Wang, "Cryptography and Relational Database Management Systems", Int. Database and Engineering and Application Symposium, 2001.

[HIL02] H. Hacigumus, B. Iyer, C. Li, S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model", ACM SIGMOD, 2002.

[ISO98] International Standardization Organization (ISO), *Integrated Circuit(s) Cards with Contacts – Part 1: Physical Characteristics*, ISO/IEC 7816-1, 1998.

[Mat00] U. Mattsson, *Secure.Data Functional Overview*, Protegrity Technical Paper TWP-0011, 2000. http://www.protegrity.com

[Ora02] Oracle Corp., *Advanced Security Administrator Guide*, Release 9.2, 2002.

[PBV01] P. Pucheral, L. Bouganim, P. Valduriez, C. Bobineau, "PicoDBMS: Scaling down Database Techniques for the Smartcard", *VLDB Journal (VLDBJ)*, 10(2-3), 2001.

[Qck02] The Quickbase Database Service Provider https://www.quickbase.com/

[ScS99] B. Schneier, A. Shostack, "Breaking up is hard to do: Modeling Security Threats for Smart Cards", *USENIX Symposium on Smart Cards*, 1999.

[Sky02] SkyDesk : @Backup (Storage Service Provider) http://www.backup.com/index.htm

[Sma02] SmartRight: protecting content in the digital age http://www.smartright.org/

[Sun99] Sun Microsystems, *JavaCard 2.1 API Specification*, JavaSoft documentation, 1999.

[TPC02] Transaction Processing Performance Council, http://www.tpc.org/

[Tua99] J.-P. Tual, "MASSC: A Generic Architecture for Multiapplication Smart Cards", *IEEE Micro Journal*, N° 0272-1739/99, 1999.