Data confidentiality: to which extent cryptography and secured hardware can help

Nicolas Anciaux, Luc Bouganim, Philippe Pucheral

Data confidentiality: to which extent cryptography and secured hardware can help

Nicolas Anciaux*, — Luc Bouganim* — Philippe Pucheral*,****

* *INRIA Rocquencourt, SMIS Project*
Domaine de Voluceau, 78153 Le Chesnay cedex, France
{Firstname.Lastname}@inria.fr

** *PRiSM Laboratory, University of Versailles*
45, avenue des Etats-Unis, 78035 Versailles cedex, France
{Firstname.Lastname}@prism.uvsq.fr

ABSTRACT. *Data confidentiality has become a major concern for individuals as well as for companies and administrations. In a classical client-server setting, the access control management is performed on the server, relying on the assumption that the server is a trusted party. However, this assumption no longer holds given the increasing vulnerability of database servers facing a growing number of external and even internal attacks. This paper studies different alternatives exploiting cryptographic techniques and/or tamper-resistant hardware to fight against these attacks. The pros and cons of each alternative are analyzed in terms of security, access control granularity and preserved database features (performance, query processing, volume of data). Finally, this paper sketches a hybrid approach mixing data encryption, integrity control and secured hardware that could pave the way for future highly secured DBMS.*

KEYWORDS: *data confidentiality, access control management, data encryption, integrity control, tamper-resistant hardware*

Annals of Telecoms. Volume X – n° X/2005, pages 1 à X

1. Introduction

Safeguarding data confidentiality has become a primary concern for citizens and administrations. The need to gather, share and analyze personal data is manifold: increasing the quality of care thanks to Electronic Health Record (EHR) systems, simplifying administrative procedures and increasing their efficiency, personalizing the services delivered by a wide collection of smart objects in an ambient intelligent surrounding (cell phones, home networks, consumer electronics, etc) and even increasing the security of states by tracking suspect individuals (e.g., building personal profiles from government and commercial databases to fight against terrorism [EFF]). While processing personal data usually serves a societal purpose, it introduces an unprecedented threat on user's privacy[1].

All around the world, governments enact specific laws to regulate the processing of personal data, like the Federal Privacy Act in the US [Pri74] and the Data Protection Directive in the EU [Eur85]. However, translating law statements into convincing technology means is a rather difficult task. According to the Computer Security Institute and the FBI, the attacks on database servers are increasing every year despite tighter security practices, and worse, almost half of the attacks are conducted by insiders [CSI04]. This demonstrates the vulnerability of traditional security features provided by database servers.

Database security encompasses three main properties: *confidentiality*, *integrity* and *availability* [KrT]. Roughly speaking, the confidentiality property guarantees that a protected data can never be accessed by an unauthorized person or program. The integrity property guarantees that the data cannot be corrupted in an invisible way. Finally, the availability property protects the system against denial of service attacks. In this paper, we concentrate on the data confidentiality issue. However, data integrity will also be discussed every time data tampering can lead to an information leakage (e.g., data tampering is performed with the objective to mislead the DBMS access control). Preserving the data confidentiality amounts to enforce the access control policies defined on the DBMS. An access control policy, that is to say a set of authorizations, can take different forms depending on the underlying data model. For example, an authorization in a relational database is usually expressed as the grant to execute a given action (e.g., Select) on a relational table or view (i.e., a virtual table computed by a SQL query) [MeS93]. An authorization on an XML document is usually expressed as a composition of positive (resp. negative) rules selecting authorized (resp. forbidden) sub-trees in the document thanks to XPath expressions [BCF01, GaB01, DDP02]. Another dimension of the access control model is the way by which authorizations are administered, following either a Discretionary (DAC) [HRU76], Role-Based (RBAC) [SCF+96] or Mandatory (MAC) approach [BeL76].

[1] Note that data confidentiality is also an important concern for companies willing to protect their data (business strategy, know-how, customer data, etc) against industrial and commercial spying.

In this paper, we do no assumption on the way authorizations are actually expressed and administered, except for illustrative purposes.

Whatever the access control model, the authorizations enforced by the database server can be bypassed in a number of ways. An intruder can infiltrate the information system and try to mine the database footprint on disk. Another source of threats comes from the fact that many databases are today outsourced to Database Service Providers (DSP). Then, data owners have no other choice than trusting DSP's arguing that their systems are fully secured and their employees are beyond any suspicion, an assumption frequently denied by facts [HIM02]. Finally, a database administrator has enough privileges to tamper the access control definition and to spy on the DBMS behavior. While this later situation is neither new nor specific to electronic databases systems[2], it must be considered with a particular care.

The objective of this paper is to study alternatives to fight against these sources of attacks. The resort to cryptographic techniques and tamper-resistant hardware to complement and reinforce the access control has recently received much attention from the database community (e.g., [BoP02, HIL+02, PBV+01, VMS02]). In this paper, we analyze and compare different ways to combine data encryption, hashing, signatures and secured operating environments in order to increase the confidence put in database systems to preserve data confidentiality. Then, we draw from this study important research perspectives.

The paper is organized as follows. Section 2 introduces background material related to the classification of attacks and attackers and to the cryptographic and secured hardware techniques of interest. Section 3 studies to which extent data encryption participates to the preservation of data confidentiality. Section 4 deals with the protection of data embedded into secured hardware devices. Section 5 focuses on approaches mixing data encryption and secured hardware. Section 6 shows that the preceding approaches could be made more general at the price of more complex integrity controls and sketches interesting open issues. Finally, Section 7 concludes.

2. Attacks, attackers and cryptographic instruments

This section gives a classification for the different attacks threatening data confidentiality and for the attackers who may conduct them. It then presents usual instruments based on cryptographic and secured hardware techniques to counter these attacks.

[2] For example, a rule specifying that "the doctor is the only person to get full access to her patients' medical folder", is actually translated in a paper-based archive by "the doctors and the archive employees has full access to the patient's folder".

2.1. A classification of attacks and attackers

As stated in the introduction, this paper focuses on attacks threatening data confidentiality, that is to say attacks trying to read unauthorized data. However, attacks trying to modify unauthorized data must also be considered when they aim at abusively altering the user's authorizations. This can be achieved by tampering the view definitions, the privilege table or the data participating in the computation of a granted view. This leads to consider four classes of attacks:

- *Data Snooping:* an attacker examines the data and deduces an unauthorized information
- *Data Altering*: an attacker deletes or modifies (even randomly) a data
- *Data Substituting*: an attacker replaces a valid data by another valid data
- *Data Replaying*: an attacker replaces a valid data by one of its older version

Note that these attacks can be directed either against the data at rest (i.e., on disk), against the data being processed by the database system (including the server's main memory content) and against the data exchanged on the communication link between the client and the server.

These attacks can be conducted by different categories of attackers. We distinguish three classes of attackers according to their initial granted privileges:

- *Intruder:* a person with no database privilege, who infiltrates a computer system and tries to extract valuable information from the database footprint on disk. [App03, Eru01] state that the majority of computer attacks are targeting the data at rest.
- *Insider:* a person properly identified by the database server (i.e., a registered user) who tries to get information exceeding her own privileges. The owned privileges give her more abilities than the intruder to tamper the system and to deduce valuable unauthorized content. [CSI04] reported that around 50% of information theft comes from internal attacks.
- *Administrator:* a person who has enough (usually all) privileges to administer a computer system (System Administrator) or a DBMS (Database Administrator or DBA). These privileges give her the opportunity to access the database files and to spy on the DBMS behavior (e.g., main memory monitoring).

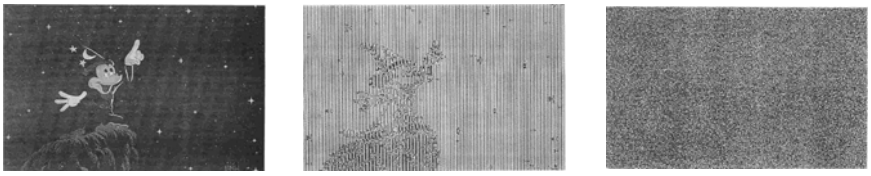
The intersection between these classes can be non-empty, considering for example that a talented intruder may be able to usurp the identity of an administrator. Actually, the objective of this classification is to attach a name to categories of threats of increasing dangerousness.

2.2. Cryptographic and secured hardware instruments

This section describes some instruments used in the literature for enforcing data confidentiality. These instruments are based on cryptographic techniques or on specific secured hardware and constitute the building blocks for tamper-resistant database systems.

- *Encryption:* the purpose of encryption is to ensure data opacity (protection against snooping attacks) by keeping the information hidden to any unauthorized persons (e.g., intruders). The plaintext version of an encrypted data cannot be recovered from the ciphertext without the decryption key. In general, and more specifically in the database context, care should be taken about the chosen encryption mode to prevent an attacker to analyze repetitive patterns in the encrypted text. Figure 1 exemplifies this concern with a 3-DES encryption of a picture using the ECB (each block is encrypted individually) or the CBC mode (the result of the encryption of a block depends on all previous encrypted blocks).

Figures taken from José Rolim and F. Schutz courses at http://cui.unige.ch/tcs/cours/crypto.



(a) Original *(b) 3-DES-ECB encryption* *(c) 3-DES-CBC encryption*

Fig. 1. Encryption opacity.

- *Cryptographic hash functions (CHF)*: a CHF is a mathematical operation which takes as input a data of arbitrary length and computes a hash value, i.e., a fixed size *digest* of that data, that is used afterwards to detect any modification in the input data. A CHF ensures that it is computationally infeasible to recover any input of the hash function given its digest output, or to find two distinct inputs having the same digest output. Popular CHF include *Message Digest 5* (MD5) [Riv92] and *Secure Hash Algorithm 1* (SHA-1) [NIS95]. The result of a hash function used to detect whether the data has been tampered is called a *Modification Detection Code* (MDC). A MDC acts like a sensor mesh coating the data, which breaks as soon as an attacker updates a single bit of the data.
- *Origin authentication [MOV97]*: when a hash function is associated to a secret key, its output is called a *Message Authentication Code* (MAC). Checking the MAC allows to prove that a data has been originated by a key owner. Combining CHF and asymmetric encryption allows proving the origin of a data (i.e., digital signature) and thus, provides the non-repudiation property. New cryptographic algorithms have been devised to achieve both encryption and

authentication. EAX [BRW04] is a representative example of this class of algorithms.

- *Transaction authentication [MOV97]*: while MDC, MAC and digital signatures may be used to establish that data was generated by a specified party at some time in the past, they provide no inherent timeliness guarantee. Therefore, these techniques alone cannot deal with data replaying attacks. Transaction authentication denotes data authentication augmented with a timeliness guarantee. The timeliness guarantee is typically provided by an appropriate use of time-variant parameters (TVP) such as random numbers in challenge-response protocols, sequence numbers and timestamps.
- *Merkle hash trees [Mer90]*: a Merkle hash tree is a virtual tree computed dynamically over encrypted data blocks (at the leaves of the tree) to attest the integrity of any granule (block or group of blocks) according to the current root hash value. As pictured in Figure 2, each intermediate hash value results from the hash of the concatenation of its children hashes. Each leaf hash value is obtained by hashing the corresponding encrypted data block. To check the integrity of a retrieved encrypted data block, the trusted party has to retrieve some hashes to re-compute the root hash value and compare it to the expected one. In the example of Figure 2, when retrieving the ciphertext C4, hash values H3, H9 and H14 are also retrieved to compute the hash value H15'=H (H14||H (H9||H (H3||H(C4))) and compare it with H15. Note that when the data contains some timeliness information, the tree additionally guarantees that the accessed data has not been replayed.

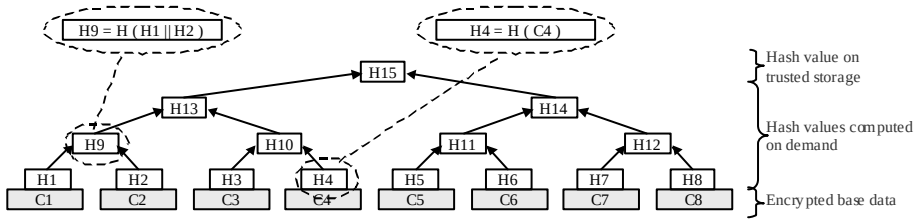


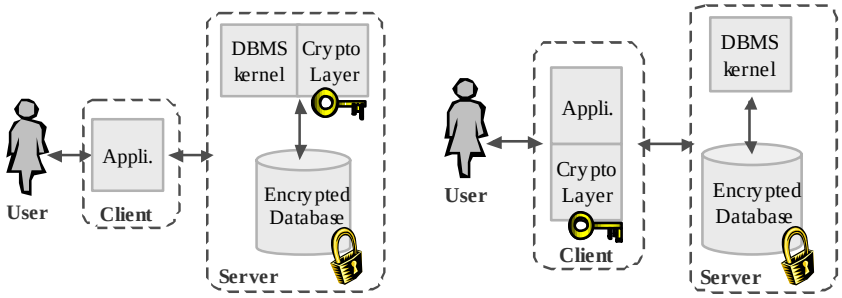
Fig. 2. Merkle Hash Tree.

- *Tamper-Resistant Hardware:* hardware protected devices guarantee that the embedded data and programs cannot be tampered. Secure co-processors and smart cards are today the most widely used hardware protected devices. They provide a very high level of security protecting their content against any form of snooping and tampering. For example, in smart cards, layers of metal are covering the chip and detect any invasion attempt, the embedded algorithms are proved secure against software attacks, the radiations produced by the processor during normal operations are limited, the power consumption is maintained roughly constant as well as the chip's temperature, fault generation is avoided thanks to on-chip sensors (low frequency and/or voltage alarms) that detect abnormal input signals and then disable the chip. Thus, the data/programs

downloaded on such devices are stored/ran as expected and are destroyed if hacked.

3. Data encryption approaches

Traditional database security policies rely on user authentication, communication encryption and server-enforced access controls [BPS96]. Unfortunately, these mechanisms are inoperative against most of the attacks identified in Section 2. Several attempts have been made recently to strengthen server-based database security policies thanks to database encryption. In this section, we discuss different techniques proposed by DBMS manufacturers and academic researchers to increase the security level of traditional client/server DBMS. These strategies have the benefit to keep the existing DBMS kernels unaffected by simply adding a security layer on top of them.



(a) Encryption at the server side. *(b) Encryption at the client side.*

Fig. 3. DBMS architectures exploiting data encryption.

The main DBMS manufacturers provide cryptographic toolkits to encrypt the database tuples. DB2 UDB includes encryption functions for string data with a column granularity [IBM03]. Oracle provides the Oracle Obfuscation Toolkit as a PL/SQL package to encrypt string and binary data [Ora01, Ora02]. Finally, Microsoft SQL Server 2000 supplies encryption for the network traffic and for metadata (e.g., views definitions). This leads to the functional architecture pictured in Figure 3.a. The encryption of the data at rest prevents intruder attacks. However, the data is decrypted on the server at runtime (encryption keys must be transmitted or kept on the server side) and then the protection against administrator attacks is rather weak. In this context, the server memory and the keys storage are assumed to be trusted, i.e. a pirate cannot gain access to keys nor data currently in memory.

In the same spirit, the study presented in [IMM+04] assumes that the server is trusted enough to proscribe performing a memory dump. Thus, the authors focus on protecting secondary storage using encryption, minimizing the degradation of the overall system performance. A new storage model inspired by PAX [ADH+01] is devised for an efficient evaluation of the result, vertically decomposing the database

tuples stored in each disk page and encrypting each (confidential) column separately. Other studies address the query response time overhead induced by data decryption and propose order-preserving encryption techniques to evaluate range selection predicates [AKS+04, DCP+04] directly on the encrypted data. These encryption strategies reduce the amount of data to be decrypted during the query evaluation, improving the performance. However, this solution suffers from the same security breach as commercial DBMSs.

Solutions complementary to database encryption have been recently investigated to guard the DBMS from the DBA. Protegrity [Mat04] introduces a clear distinction between the role of the DBA, administering the database resources, and the role of the SA (Security Administrator), administering user privileges, encryption keys and other related security parameters. This distinction is also made effective at the system level by separating the database server from the security server. The gain in confidence comes from the fact that an attack requires a conspiracy between DBA and SA. This solution can adapt to most DBMS products (Oracle, IBM DB2, SQL Server, Sybase). However, one must keep in mind that the data is still decrypted by the server at query execution time.

To circumvent this weakness, some academic proposals assume the decryption be performed on the client rather than on the server side [HIL+02, HIM02, HIM04, HIM05, OSC03, DDJ+03]. This leads to the functional architecture pictured in Figure 3.b. Thus, the cryptographic layer (and the associated keys) as well as (part of) the query processing module are shift on the client side. The query issued by the user is split on the client into two sub-queries. The first one is evaluated on the server, directly on the encrypted data, and the second one is processed on the client after decryption of the result of the first sub-query. On the server side, the processing is made possible thanks to additional fuzzy indexing information added to the encrypted database. These fuzzy indices are used by the server to select the superset of the encrypted database required to compute the final result. The client then decrypts the data and completes the query evaluation. Note that the more precise the indices on the server side, the more efficient the overall execution process. However, the precision of the indices also increases the risk of data snooping (by comparing indices, the attacker may deduce some information). A study balancing the efficiency allowed by the precision of the indices and the confidentiality of the encrypted data is conducted in [DDJ+03].

Although these solutions provide valuable protections against some confidentiality attacks (e.g., intruder attacks), their limit is twofold. First, the relationship between indices and data gives means to the attackers to infer some information (e.g., data distribution). Moreover, a pirate obtaining a single information on the database content would easily deduce many others by disclosure propagation. Data snooping attacks could thus bypass the protection settled by encryption. Second, these solutions either do not prevent administrator attacks (Figure 3.a) or proscribes data sharing among users having different privileges (or at least makes it rather difficult to organize). Indeed, in this latter case, the privileges

of each user have to be hardcoded by the encryption process (Figure 3.b). Despite these drawbacks, these solutions are efficient and have the main advantage of being adaptable to existing database servers.

4. Secured hardware approaches

A drastic solution to enforce the security of the DBMS code and of the database consists in embedding them into a Secured Operating Environment (SOE), with the objective to inherit from the SOE its intrinsic security. Smart cards are good examples of SOE and exhibit a very high security level for the on-board data, protecting them against all the forms of attacks identified in Section 2 [ABP03b]. Embedding the DBMS code as well into the secured hardware prevents from any information disclosure at query processing time. Figure 4 pictures a secured DBMS architecture relying on secured hardware.

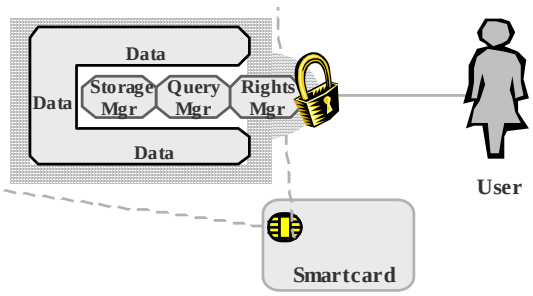


Fig. 4. DBMS architecture exploiting secured hardware.

However, secured hardware devices are usually highly limited in terms of resources to cope with strong security constraints. To illustrate this, smart card microcontrollers must fit in 25 mm². Consequently, CPU, RAM and stable storage compete on the same silicium die. This generally leads to scarce RAM resources (because of the low density of RAM cells) and to high density electronic stable memory (e.g., EEPROM, Flash) that exhibit fast read time but very slow write time. The peculiarities of these environments compel to deeply revisit existing DBMS techniques (e.g., storage and indexation models, query execution strategies). This section presents state of the art works in this domain, and then discusses the limitations induced by this approach.

The first attempt towards a DBMS embedded in a smart card is ISOL's SQLJava Machine [Car99] and the ISO standard for smart card database language, SCQL [ISOp7]. Both are addressing generation of smart cards endowed with 8 kilobytes of stable memory. SQLJava Machine and SCQL design are limited to mono-relation queries. This limitation has an impact on the possible target applications, reducing the access rights powerfulness and thus the sharing capabilities. However, this proposal exemplified the interest for dedicated smart card DBMS.

A full-fledged on-chip DBMS called PicoDBMS has been more recently designed [PBV+01] and prototyped [ABB+01], taking advantage of the increasing computing and storage resources of advanced smart card platforms. The DBMS kernel acts as a doorkeeper that authenticates the users and solely delivers the data corresponding to their privileges. In a relational DBMS, the powerfulness of the access control is directly determined by the complexity of the views that can be built. To be able to provide fine grain privileges, PicoDBMS supports complex query processing including select, project, join and aggregate operators. This study raises specific design rules required to build a full-fledged DBMS kernel complying with the smart card's constraints. Following these rules, the technical solution relies on highly compact storage structures (attributes are stored in domains, acting as a compression by dictionary), on ad-hoc compact indexation techniques (attributes possibly engaged in selections and joins are linked by rings of pointers), and on a pure pipeline query execution model consuming a minimal (bounded) amount of RAM. A mini-benchmark has also been settled [Anc04] for smart card DBMS. The metrics is comparable to those used for traditional storage device, expressed in capacity (number of storable tuples), latency (time to return the first result) and transmission rate (number of results delivered per second).

A recent study proposes specific storage techniques to manage data in Flash memory on a smart card [BSS+03]. The design is limited to mono-relation queries and is strongly impacted by the physical characteristics of the target smart card architecture. Indeed, the database is stored in NOR Flash memory generally dedicated to store programs as ROM replacement. Since updates in NOR Flash memory are very costly (updating a single data induces large and costly block erasure), the techniques are driven by update cost minimization (deleted bit and dummy records). While this study shows the impact of hardware characteristics on the DBMS internals, it does not address complex query processing, mandatory to express fine grain authorizations.

The initial application target of smart card DBMS was the management of secured personal folders (e.g., healthcare, scholarship, insurance, etc.). While this is still an important application domain and deserves a growing attention from major industrial actors (e.g., MasterCard’s Open Data Store [Mas02]), the introduction of secured chips in usual computing infrastructures paves the way for new large-scale applications. First, tamper-resistant chips should be integrated soon in any PC platforms [TCP] and consumer appliances [Sma] to prevent piracy and enforce Digital Right Management. Second, ambient intelligence is flooding many aspects of our everyday life with smart objects gathering information about our habits and preferences, threatening the citizen’s elementary right to privacy. Therefore, embedding data management techniques in secured chips can be the mean by which each individual can preserve the expected control on her own data.

To conclude, this approach provides an unequalled security level protecting the database against all the attacks depicted in Section 2. This security level is obtained thanks to (i) hardware protection and (ii) database self-administration. However, the

volume of protected data as well as the DBMS code footprint (and thus the database functionalities) are bounded by the hardware constraints. To illustrate this, the most advanced smart card prototypes are endowed with 1 megabyte of stable storage. In addition, data sharing can only be obtained by physically sharing the smart card thereby limiting the applicability of the approach. In the long term, these constraints will be alleviated, allowing embedding larger and more powerful DBMS that could even be acting as servers connected to the network. However, such an evolution would inevitably conduct to administrate the database, and then would reintroduce the threat of administrator attacks.

5. Mixing encryption and secured hardware

This section discusses a hybrid approach relying on database encryption at the server and on a secured processing environment at the client side. This combination of instruments allows data sharing among several users (and devices) while protecting the database from administrator attacks.

Chip-Secured Data Access (C-SDA) [BoP02] is a client-based security component acting as an incorruptible mediator between a user and an encrypted database. At query execution time, C-SDA checks whether the user has enough privileges to issue the query. In the positive case, it participates to the query evaluation in collaboration with the server, decrypts the data and externalizes the authorized result to the client terminal. This component is embedded into a smart card to prevent any tampering to occur at the client side. This cooperation of hardware and software security components allows reestablishing the orthogonality between access control management and data encryption. The architecture of this solution is pictured in Figure 5.

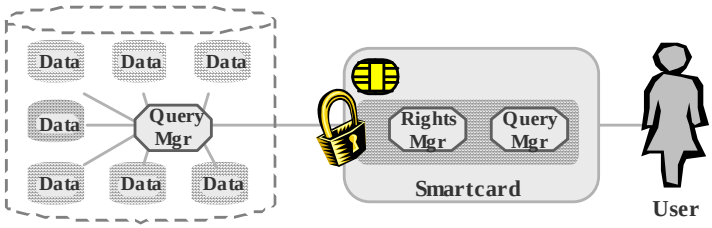


Fig. 5. Architecture mixing encryption and secured hardware.

While the proposed architecture is attractive, it seems difficult to instantiate it on current smart cards given their limited resources. Indeed, the hardware constraints would lead to poor performance when decrypting and processing large amounts of data (limited bandwidth, working memory and CPU). To tackle this issue, [BoP02] delegates to the server the part of the query processing that can be done directly on the encrypted data. The decryption and the remaining part of the processing must be done in the client tamper-resistant device in a pipeline fashion (to prevent memory overflow).

The proposed solution uses a specific encryption scheme allowing delegating projections, equi-selections, equi-joins, and grouping to the server: each column value is encrypted separately using an ECB mode algorithm (e.g., triple DES) applied on clear text values. Therefore, the equality property is preserved among encrypted data. [BoP02] shows that, thanks to this property, any SQL query can be split in two parts, the first one dealing with equality predicates on the encrypted data and the second part being calculable in a pure pipeline fashion. Optimizations based on a multi-stage cooperation between the smart card and the server have been proposed to minimize the amount of processed data and the transmission, decryption and CPU costs.

The limitations of this solution are the following. First, by preserving the equality among the encrypted data (a necessary requirement to cope with current smart card constraints), the encryption process provides a limited opacity. Thus, snooping attacks are possible using statistical analysis on the encrypted data. Second, this solution assumes no collusion between the client and the server. Indeed, to gain access to unauthorized data, a malicious client could try to modify the data on the server, even if encrypted, using substitution, random modification, etc. Thus, the system is not protected against altering, substituting and replaying attacks.

This approach is however promising. The volume of storable data is unbounded, the solution provides data sharing and a reasonable level of performance and functionality can be obtained. New techniques have to be devised in order to ensure a higher security level while delegating part of the processing to a potentially untrusted server.

6. Integrity control add-on: a step toward broader solutions

This section builds on the previous ones and adds integrity verifications within the secure operating environment to ensure that the accessed data is the original one and has not been altered, substituted or replayed by a malicious user (e.g., an insider). We first introduce a generic architecture where the server is assumed to be hosted in an untrusted environment. We then present two approaches exploiting encryption, secured hardware and integrity control techniques to protect from all the attacks of Section 2. These two approaches are however restricted in terms of query processing capabilities. Finally, as an introduction to future research directions, we sketch a possible approach allowing performing complex query processing while keeping the same security level.

6.1. Generic architecture

The reference architecture depicted in Figure 6 includes three elements, each with a different level of security assigned to it. The Secure Operating Environment

(SOE) is the unique element of trust. Within the SOE, processing and data are trusted, secured and hidden to any attacker. Conversely, the Untrusted Operating Environment (UOE) may be the focus of attackers. The UOE does not offer any security guarantee on the data it holds, neither on the processing it does, thus requiring encryption and integrity instruments (see Section 2). Finally, the Rendering Terminal (RT) is the mean by which the query result is delivered to the user. Thus, the RT has the same level of trust as the user himself; otherwise the RT could not deliver the results. The RT should not have access to unauthorized data or temporary results since the user could tamper the RT.

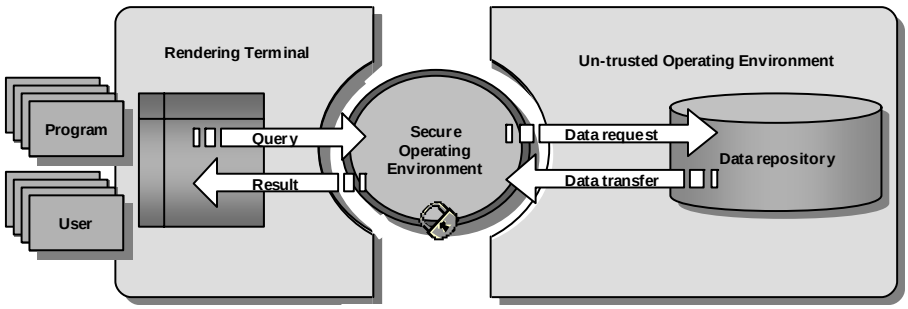


Fig. 6. Reference Architecture.

6.2. Existing approaches

Two solutions, called Trusted-DataBase (TDB) [VMS02, MVS00] and GnatDB [Vin02], have been proposed. These solutions target emerging applications requiring running trusted programs on untrusted hosts. For instance, to enforce Digital Right Management (DRM) policies, distributors of digital content need to control the usage of the supplied content made by consumers on multimedia appliances. The terms of a DRM policy are implemented by a program (i.e., a license interpreter) that must run in a secured environment to be reliable.

The first solution, TDB, targets an architecture endowed with a large SOE like a secured co-processor (powerful CPU, boosted cryptographic performance, megabytes of RAM and FLASH memory) within a tamper-responding enclosure like the IBM 4758 [DLP+01]. The second solution, GnatDB, is a limited version of TDB addressing secured architectures endowed with smaller SOE like smart cards or secure tokens.

In TDB, the data is stored encrypted on the UOE into large chunks (100 bytes to 100 kilobytes) stored in a log fashion. The integrity of each partition is protected by a Merkle Hash Tree stored within a location map (also organized as a hierarchy). Encryption keys and the hash roots of each tree are stored within the SOE. The data is encrypted/decrypted within the SOE, its integrity is checked using the Merkle Hash Tree, and replay attacks are prevented using increment-only counters associated to the data. TDB implements some transactional features (a set of updates can be declared to be atomic). Supported queries are equivalent to relational mono-

table selections (TDB has an object-oriented interface). The performance analysis of the system (TPC-B benchmark) shows the feasibility of the approach.

GnatDB is the light version of TDB, fitting into a smart card and complying with the smart card constraints. GnatDB ensures the same high security level as TDB. GnatDB does not rely on Merkle Hash Trees to reduce the code footprint, minimizes the RAM consumption within the SOE, and provides a light transactional kernel (atomicity and durability only). The location map is not stored as a hierarchy but within an array (again to minimize the code footprint), written sequentially at each update. This technique limits the scalability of this system, but provides transaction atomicity. In this proposal, performance is not the primary concern, and query processing is not addressed.

These solutions successfully rely on a secure operating environment, on encryption and on integrity techniques to ensure a very strong level of security, actually much higher than the one provided by the preceding approaches (except those introduced in Section 4). Also, the main DBMS transactional functionalities are supported. However, the query processing capabilities are limited, thus proscribing applications that rely on fine grain privileges. Thus, these works show the benefit of combining SOE, encryption and integrity control and pave the way for more powerful approach dealing with real query processing.

6.3. A step toward broader solutions

Considering query processing, and thus fine grain privileges, raises two important questions: (i) which part of the query processing could be delegated to untrusted parties without reducing the security level and (ii) how to cope with the constrained resources of the SOE while processing the remaining part of the query. These questions are addressed in the following and some perspectives are drawn.

Compared with an architecture where the DBMS code and the database are entirely hosted in a SOE (e.g., PicoDBMS), the security level is obviously reduced as soon as part of the system stands on an UOE, potentially controlled by an attacker (e.g., the UOE administrator). Thus, externalized data and externalized processing (i.e., processing done on the UOE and messages exchanged between the SOE and the UOE) must be opaque and tamper resistant.

While cryptographic techniques may provide such opacity and tamper resistance for the externalized data, providing opacity and tamper resistance for externalized processing is today an open problem. Indeed, tamper resistance means that any externalized processing should be checked afterward in the SOE. Actually, such verification might be as expensive as the processing itself. Some techniques indeed exist for simple selections [GKM+04] but the problem remains open for more

complex operations[3]. Thus, the tamper resistance of externalized processing seems difficult to attain. Externalized processing opacity seems also unreachable since delegating processing always discloses some information (even if it is done on encrypted data), which can be exploited by an attacker. As observed in [KaC05], providing total opacity on externalized processing appears to be a Private Information Retrieval (PIR) [CGK+95] problem.

A possible, but drastic, solution to cope with these issues is to reduce externalized processing to its strict minimum. This can be achieved by reducing the UOE to a server of encrypted blocks, each one characterized by an address and a size. The advantages of this option are to obtain the highest level of opacity and to make the problem of tamper resistance tractable. Indeed, it only discloses the accesses to the data blocks and the number and size of the exchanged messages. Moreover, checking the external processing integrity simply amounts to verify that each retrieved data block has not been altered (data integrity), substituted (the block originates from the expected address with the expected size) or replayed (this is the latest version of this block).

Query processing is thus confined within the SOE, except the retrieval of external data, which is necessarily delegated to the UOE. The execution strategy is therefore highly impacted by the data access cost and the limited amount of RAM available in the SOE. Indeed, the external data must be decrypted and its tamper-resistance must be checked within the SOE, thereby significantly increasing the data access cost (two orders of magnitude more expensive than accessing the internal memory of the smart card). This leads to strongly minimize access to irrelevant data. This could be achieved by an extensive use of indices, vertical fragmentation and fine grain access to data (and thus fine grain encryption, authenticity, etc.)

The limited amount of RAM has also a large impact on query execution. Indeed, techniques as the one proposed in PicoDBMS or [ABP03a] cannot be used since they incur numerous iterations on the data. Aggressive indexation is clearly required in this context but do not solve totally the problem unless every possible view result is materialized, thereby hurting data and access right dynamicity.

A possible approach is to use the small amount of available RAM for query processing and to rely on swapping on the UOE or RT in order to adapt state-of-the-art algorithms (e.g., sort-merge join). Obviously, swapped data should have the same opacity and integrity characteristics as the base data. Thus, swapping in and out data roughly doubles the data access cost. To maximize the usefulness of the

[3] In [GKM+04], the author use Merkle hash trees to authenticate the result of simple selections produced by an untrusted server (but on plaintext data). Roughly, the idea is to sort the data according to the desired criteria (e.g., the age of a person when the predicate is age $\in$ [30-40]) and to provide hash values for the data that are not in the result (e.g., less than 30 and more than 40). These hash values combined with the computed hash of the result should be equal to the hash of the queried relation. This interesting result is however difficult to adapt to all operators of the relational algebra. Moreover, in our case, the computation must be done on encrypted data!

RAM, the following principle could be adopted: at every step during query processing, the SOE should only retrieve the subset of data strictly necessary to perform that step. While this seems natural, it is opposed to classical execution techniques where caching and prefetching are used to increase performance.

We are currently investigating the approach proposed above and believe that it is a first step towards a highly secure DBMS providing query processing features, views and powerful privileges. However, many problems remain open and deserve further investigation.

7. Conclusion

This paper studied and compared alternatives to enforce data confidentiality, by means of cryptographic techniques and/or specialized secured hardware.

Approaches based solely on data encryption provide a security and functionality level that depends on the place encryption/decryption is actually performed. Decryption at the server side exposes the system to administrator attacks. Moving it to the client side increases the security at the price of loosing data sharing capabilities, thereby limiting the applicability of the approach to the management of private databases.

An unequalled security level can be obtained by embedding the DBMS and the database into a secure operating environment. Approaches based on smart cards have been explored so far. While limited by the smart card hardware constraints, these solutions are well suited for handling secured personal folders (e.g., healthcare folder) as well as for a broader range of applications dealing with smart objects in an ambient intelligence surrounding. Indeed, a strong protection is required for these highly personal and sensitive data. The smart card limitations should be alleviated in the mid-term thus increasing the applicability of this approach.

Combining cryptographic techniques and secure operating environment allows enforcing confidentiality while preserving important DBMS properties like data sharing, unbounded data volume and performance. Few works have been conducted in this area. C-SDA is one of them. It uses a specific encryption scheme allowing delegating part of the processing to the server on the encrypted data, thus allowing efficient processing. As a side effect, this encryption scheme allows attackers to perform snooping attacks. This limits the scope of this solution to scenarios where there is no collusion between the client and the server.

Finally, we presented an approach where the server is hosted in an untrusted environment and integrity techniques are added to prevent from altering, substituting or replaying the data. Two systems representative of this approach already exist but they do not provide real query facilities and then implement coarse grain privileges. The problem becomes much harder as soon as query processing is considered. Preliminary ideas have been sketched in this paper to support complex

query processing without entailing a reduction of security. These ideas could pave the way for future highly secured DBMS providing query processing facilities, views and fine grain privileges and deserve further investigations.

8. References

[ABB+01] Anciaux N., Bobineau C., Bouganim L., Pucheral P., « PicoDBMS: Validation and Experience », *International Conference on Very Large Databases (VLDB)*, demo session, 2001.

[ABP03a] Anciaux N., Bouganim L., Pucheral P., « Memory Requirements for Query Execution in Highly Constrained Devices », *International Conference on Very Large Databases (VLDB)*, 2003.

[ABP03b] Anciaux N., Bouganim L., Pucheral P., « Database Components on Chip », *ERCIM news*, 2003.

[ADH+01] Ailamaki A. G., DeWitt D. J., Hill M. D., Skounakis M., « Weaving Relations for Cache Performance », *International Conference on Very Large Data Bases (VLDB)*, 2001.

[AKS+04] Agrawal R., Kiernan J., Srikant R., Xu Y, « Order-Preserving Encryption for Numeric Data », *ACM International Conference on Management of Data (SIGMOD)*, 2004.

[Anc04] Anciaux N., « Database Systems on Chip », PhD thesis, University of Versailles, 2004.

[App03] Application Security Inc., « Encryption of Data at Rest - Database Encryption », White Paper, 2002. http://www.appsecinc.com

[BCF01] Bertino E., Castano S., Ferrari E., « Securing XML documents with Author-X », *IEEE Internet Computing*, 2001.

[BDP04] Bouganim L., Dang Ngoc F., Pucheral P., « Client-Based Access Control Management for XML documents », *International Conference on Very Large Databases (VLDB)*,2004.

[BeL76] Bell D. E., LaPadula L. J., « Secure computer systems: Unified exposition and multics interpretation », Technical Report ESD-TR-73-306, The MITRE Corporation, 1976.

[BoP02] Bouganim L., Pucheral P., « Chip-Secured Data Access: Confidential Data on Untrusted Servers », *International Conference on Very Large Databases (VLDB)*, 2002.

[BPS96] Baraani A., Pieprzyk J., Safavi-Naini R., « Security In Databases: A Survey Study », 1996. citeseer.nj.nec.com/baraani-dastjerdi96security.html

[BRW04] Bellare M., Rogaway P., Wagner D., « The EAX Mode of Operation », *Fast Software Encryption (FSE)*, 2004.

[BSS+03] Bolchini C., Salice F., Schreiber F., Tanca L., « Logical and Physical Design Issues for Smart Card Databases », *ACM Transactions on Information Systems (TOIS)*, 2003.

[Car99] Carrasco L. C., « RDBMS's for Java Cards ? What a Senseless Idea ! », 1999. http://www.sqlmachine.com

[CGK+95] Chor B., Goldreich O., Kushilevitz E., Sudan M., « Private information retrieval », *Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

[CSI04] Computer Security Institute. « CSI/FBI Computer Crime and Security Survey », 2004. http://www.gocsi.com/forms/fbi/pdf.html.

[DCP+04] Damiani D., De Capitani Di Vimercati S., Paraboschi S., Samarati P., « Computing range queries on obfuscated data », *Information Processing and Management of Uncertainty in Knowledge-Based Systems* (*IPMU)*, 2004.

[DDJ+03] Damiani E., De Capitani Vimercati S., Jajodia S., Paraboschi S., Samarati P., « Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs », *ACM Conference on Computer and Communications Security (CCS)*, 2003.

[DDP02] Damiani E., De Capitani di Vimercati S., Paraboschi S., Samarati P., « A Fine-Grained Access Control System for XML Documents », *ACM TISSEC*, vol. 5, n. 2, 2002.

[DLP+01] Dyer J. G., Lindemann M., Perez R., Sailer R., Doorn L. van, Smith S. W., Weingart S., « Building the IBM 4758 Secure Coprocessor », *IEEE Computer*, 2001.

[EFF] Electronic Frontier Foundation, « Unintended Consequences: Five Years under the DMCA ». http://www.eff.org/IP/DMCA/

[Eru01] Eruces Inc., « Securing Data Storage: Protecting Data at Rest », In Dell Power Solutions magazine, Issue 4, 2001. http://ftp.dell.com/app/4q01-Eru.pdf

[Eur85] European Directive 95/46/EC, « Protection of individuals with regard the processing of personal data », Official Journal L 281, 1985.

[GaB01] Gabillon A., Bruno E., « Regulating access to XML documents », *IFIP Working Conference on Database and Application Security*, 2001.

[GKM+04] Gertz M., Kwong A., Martel C., Nuckolls G., Devanbu P., Stubblebine S., « Databases that tell the Truth: Authentic Data Publication », *Bulletin of the Technical Committee on Data Engineering*, 2004.

[HIL+02] Hacigümüs H., Iyer B., Li C., Mehrotra S., « Executing SQL over Encrypted Data in the Database-Service-Provider Model », *ACM International Conference on Management of Data (SIGMOD)*, 2002.

[HIM02] Hacigümüs H., Iyer B., Li C., Mehrotra S., « Providing Database as a Service », *International Conference on Data Engineering (ICDE)*, 2002.

[HIM04] Hacigümüs H., Iyer B., Mehrotra S., « Efficient execution of aggregation queries over encrypted relational databases », *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2004.

[HIM05] Hacigümüs H., Iyer B., Mehrotra S., « Query Optimization in Encrypted Database Systems », *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2005.

[HRU76] Harrison M. A., Ruzzo W. L., Ullman J. D. « Protection in Operating Systems », *Communication of the ACM*, 19(8):461-471, 1976.

[IBM03] IBM corporation, « IBM Data Encryption for IMS and DB2 Databases v. 1.1 », 2003. http://www-306.ibm.com/software/data/db2imstools/html/ibmdataencryp.html.

[IMM+04] Iyer B., Mehrotra S., Mykletun E., Tsudik G., Wu Y., « A Framework for Efficient Storage Security in RDBMS », *International Conference on Extending Database Technology*, 2004.

[ISOp7] International Standardization Organization, Integrated Circuit(s) Cards with Contacts - Part 7, ISO/IEC 7816-7, 1999.

[KaC05] Kantarcioglu M., Clifton C., « Security Issues in Querying Encrypted Data », *IFIP Working Conference on Database and Applications Security (DBSec)*, 2005.

[KrT] Krause M., Tipton H. F., « Handbook of Information Security Management », Auerbach Publications, CRC Press LLC. http://www.cccure.org/Documents /HISM/ewtoc.html

[Mas02] MasterCard, « MasterCard Open Data Storage (MODS) », 2002. https://hsm2stl101.mastercard.net/public/login/ebusiness/smart_cards/one_smart_ card/biz_opportunity/mods.

[Mat04] Mattsson U., « Transparent Encryption and Separation of Duties for Enterprise Databases -A Solution for Field Level Privacy in Databases », Protegrity Technical Paper, 2004. http://www.protegrity.com/whitepapers /TRANSPARENT-ENCRYPTION-FOR-ENTERPRISE-DATABASES.pdf

[Mer90] Merkle R., « A Certified Digital Signature », *Advances in Cryptology (Crypto'89)*, LNCS, vol.435, Springer--Verlag, 1990.

[MeS93] Melton J. and A. Simon R, « Understanding the new SQL: A Complete Guide », Morgan Kaufmann, 1993.

[MOV97] Menezes A., Van Oorschot P., Vanstone S., « Handbook of Applied Cryptography », CRC Press, 1997. www.cacr.math.uwaterloo.ca/hac.

[MVS00] Maheshwari U., Vingralek R., Shapiro W., « How to build a trusted database system on untrusted storage », *Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[NIS95] NIST, « Secure hash standard », FIPS Publication 180-1, 1995.

[Ora01] Oracle Corporation, « Database Encryption in Oracle9i », 2001. otn.oracle.com/deploy/security/oracle9i.

[Ora02] Oracle Corporation, « Oracle Advanced Security - Administrator's Guide », Release 2 (9.2), Part No. A96573-01, 2002

[OSC03] Ozsoyoglu G., Singer D., Chung S. S., « Anti-Tamper Databases: Querying Encrypted Databases », *IFIP Working Conference on Database and Applications Security (DBSec)*, 2003.

[PBV+01] Pucheral P., Bouganim L., Valduriez P., Bobineau C., « PicoDBMS: Scaling down Database Techniques for the Smart card », *Very Large Data Bases Journal (VLDBJ)*, 2001.

[Pri74] The Privacy Act, 5 U.S.C. § 552a, 1974. http://www.usdoj.gov/04foia/privstat.htm.

[RIV92] Rivest R.L., « The MD5 message-digest algorithm », RFC 1321, 1992.

[SCF+96] Sandhu R., Coyne E. J., Feinstein H. L., Youman C. E., « Role-based access control models », *IEEE Computer*, 29(2):38-47, 1996.

[Sma] SmartRight Technical white paper. http://www.smartright.org/images/SMR/content/SmartRight_tech_whitepaper_jan28.pdf

[TCP] Trusted Computing Platform Alliance, http://www.trustedcomputing.org/.

[Vin02] Vingralek R., « Gnatdb: A small-footprint, secure database system », *International Conference on Very Large Databases (VLDB)*, 2002.

[VMS02] Vingralek R., Maheshwari U., Shapiro W., « TDB: A Database System for Digital Rights Management », *International Conference on Extending Database Technology (EDBT)*, 2002.