Database Encryption - How to Balance Security with Performance

Ulf T. Mattsson
Protegrity Corp.

Abstract

Encryption can provide strong security for data at rest, but developing a database encryption strategy must take many factors into consideration. Organizations must balance between the requirement for security and the desire for excellent performance. Encryption at the database level, versus application level and file level has proved to be the ideal method to protect sensitive data and deliver performance. There are a multitude of architectures and techniques to improve performance: the alternatives fall into two broad categories – alternative topologies to decrease encryption overhead and techniques to limit the number of encryption operations. In addition, performance and security, in real-world scenarios, are complex issues and experts should be used who understand all available options and the impact for each particular customer environment.

Every organization must protect sensitive data or suffer potential legislative, regulatory, legal and brand consequences. Relying on perimeter security and database access control does not provide adequate security. Packaged database encryption solutions have proven to be the best alternative to protect sensitive data. This is a specialized and complex solution area and if internal resources don't have the cryptography expertise in relation to IT environment, outside expertise should be used to ensure superior performance. This paper reviews the performance aspects of three dominant topologies for database encryption.

Keywords: Performance, Database Security, Encryption, Privacy, VISA CISP, GLBA, HIPAA.

1 INTRODUCTION

There are many architectures, techniques, and tools available to Security and IT organizations to ensure security and performance are balanced and optimised.

Each of these approaches has its advantages and disadvantages. Database security is a wide research area [26, 23] and includes topics such as statistical database security [21], intrusion detection [34], and most recently privacy preserving data mining [22], and related papers in designing information systems that protect the privacy and ownership of individual information while not impeding the flow of information, include [22, 23, 24, 25]. Prior work [7] [2] does not address the critical issue of performance. But in this work, we have addressed and evaluated the most critical issue for the success of encryption in databases, performance. To achieve that, we have analysed different solution alternatives. Each topology effects security and performance differently and has advantages and disadvantages.

2 DIMENSIONS TO ENCRYPTION SUPPORT

There are three main dimensions to encryption support in databases. One is the granularity of data to be encrypted or decrypted. The field, the row and the page, typically 4KB, are the alternatives. The field is the best choice, because it would minimize the number of bytes encrypted. However, as we have discovered, this will require methods of embedding encryption within relational databases or database servers. The second dimension is software versus hardware level implementation of encryption algorithms. Our results show that the choice makes significant impact on the performance. We have discovered encryption within relational databases based on hardware level implementation of encryption algorithms entail a significant start up cost for an encryption operation. Each model also offers different operational performance, tuning possibilities, and encryption offloading capabilities. Only a few database brands are currently supporting a row or the page level encryption that amortizes this cost over larger data. The loss of granular protection will impact the security level. This is discussed in more detail in [18]. The third dimension is the location of the encryption service - local service, remote procedure service, or network attached service. Choosing the point of implementation not only dictates the work that needs to be done from an integration perspective but also significantly affects the overall security model. The sooner the encryption of data occurs, the more secure the environment—however, due to distributed business logic in application and database environments, it is not always practical to encrypt data as soon as it enters the network. Encryption performed by the DBMS can protect data at rest, but you must decide if you also require protection for data while it's moving between the applications and the database.

2.1 Database-level encryption

Database-level encryption allows enterprises to secure data as it is written to and read from a database. This type of deployment is typically done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. Database-level encryption protects the data within the DBMS and also protects against a wide range of threats,

including storage media theft, well known storage attacks, database-level attacks, and malicious DBAs. Database-level encryption eliminates all application changes required in the application-level model, and also addresses a growing trend towards embedding business logic within a DBMS through the use of stored procedures and triggers. Since the encryption/decryption only occurs within the database, this solution does not require an enterprise to understand or discover the access characteristics of applications to the data that is encrypted. While this solution can certainly secure data, it does require some integration work at the database level, including modifications of existing database schemas and the use of triggers and stored procedures to undertake encrypt and decrypt functions. Additionally, careful consideration has to be given to the performance impact of implementing a database encryption solution, particularly if support for accelerated index-search on encrypted data is not used. First, enterprises must adopt an approach to encrypting only sensitive fields. Second, this level of encryption must consider leveraging hardware to increase the level of security and potentially to offload the cryptographic process in order to minimize any performance impact. The primary vulnerability of this type of encryption is that it does not protect against application-level attacks as the encryption function is strictly implemented within the DBMS.

2.2 Storage-level encryption

Storage-level encryption enables enterprises to encrypt data at the storage subsystem, either at the file level (NAS/DAS) or at the block level SAN. This type of encryption is well suited for encrypting files, directories, storage blocks, and tape media. In today's large storage environments, storage-level encryption addresses a requirement to secure data without using LUN (Logical Unit Number) masking or zoning. While this solution can segment workgroups and provides some security, it presents a couple of limitations. It only protects against a narrow range of threats, namely media theft and storage system attacks. However, storage-level encryption does not protect against most application- or database-level attacks, which tend to be the most prominent type of threats to sensitive data. Current storage security mechanisms only provide block-level encryption; they do not give the enterprise the ability to encrypt data within an application or database at the field level. Consequently, one can encrypt an entire database, but not specific information housed within the database

3 CHOOSING THE TOPOLOGY FOR ENCRYPTION IMPLEMENTATION

We considered several possible combinations of different encryption approaches, namely; software and hardware level encryption, and different data granularity. We started with software encryption at field level. We then developed search acceleration support to index encrypted fields, and experienced a low performance overhead when searching on encrypted fields, including primary index fields. We also directed our experiments hardware level encryption only

for master key encryption.

3.1 Basic software level encryption

Initially we considered several encryption algorithms AES, RSA [10] and b) Blowfish [11] for the implementation. We conducted experiments using these algorithms and found that the performance and security of the AES algorithm is better than the RSA implementation and the Blowfish algorithm implementation. AES is fast, compared to other well-known encryption algorithms such as DES [12]. Detailed description of the algorithm is given in [12]. DES is a 64-bit block cipher, which means that data is encrypted and decrypted in 64-bit chunks. This has implication on short data. Even 8-bit data, when encrypted by the algorithm will result in 64 bits. We also implemented a secure method to preserve the type and length of the field after encryption, see below. The AES implementation was registered into the database as a user defined function (UDF) (also known as foreign function). Once it was registered, it could be used to encrypt the data in one or more fields - whenever data was inserted into the chosen fields, the values are encrypted before being stored. On read securely access, the stored data is decrypted before being operated upon.

3.2 Basic hardware level encryption

We studied the use of HSM FIPS-140-1 level 3 Hardware Security Modules with a mix of hardware and software keys. The master key was created and encrypted / decrypted on HSM. The master key is not exposed outside the HSM. The cost of encryption/decryption consists of start up cost, which involves function and hardware invocation, and encryption/decryption algorithm execution cost, which is depended on the size of the input data. This implies that the start up cost is paid every time a row is processed by encryption. We used specialized encryption hardware from different vendors, including IBM, Eracom, nCipher, and Chrysalis for this test. On of our test beds used the IBM S/390 Cryptographic Coprocessor, available under IBM OS/390 environment with Integrated Cryptographic Enterprise IT infrastructure component Facility (ICSF) libraries. IBM DB2 for OS/390 provides a facility called ”editproc” (or edit routine), which can be associated with a database table. An edit routine is invoked for a whole row of the database table, whenever the row is accessed by the DBMS. We registered an encryption/decryption edit routine for the tables. When a read/write request arrives for a row in one of these tables, the edit routine invokes encryption/decryption algorithm, which is implemented in hardware, for whole row. We used the DES [3] algorithm option for encryption hardware. The loss of granular column-level protection will impact the security level. This is discussed and evaluated earlier.

3.3 Encryption penalty

If we compare the response time for a query on unencrypted data with the response time for the same query over the same data, but with some or all of it

encrypted, the response time over encrypted data will increase due to both the cost of decryption as well as routine and/or hardware invocations. This increase is referred to as the encryption penalty. An observation according to recent studies is that, different fields have different sensitivity [16]. It is possible for Hybrid to support encryption only on selected fields of selected tables. Encryption, by its nature, will slow down most SQL statements. If some care and discretion are used, the amount of extra overhead should be minimal. Also, encrypted data will have a significant impact on your database design. In general, you want to encrypt a few very sensitive data elements in a schema, like Social security numbers, credit card numbers, patient names, etc. Some data values are not very good candidates for encryption -- for example booleans (true and false), or other small sets like the integers 1 through 10. These values along with a column name may be easy to guess, so you want to decide whether encryption is really useful. Creating indexes on encrypted data is a good idea in some cases. Exact matches and joins of encrypted data will use the indexes you create. Since encrypted data is essentially binary data, range checking of encrypted data would require table scans. Range checking will require decrypting all the row values for a column, so it should be avoided if not tuned appropriately with an accelerated search index.

4 PERFORMANCE OF DIFFERENT ENCRYPTION TOPOLOGIES

Three obvious topologies are: 1) Network Attached Encryption Device, 2) Software, 3) Combination of Software and a Hardware Security Module (HSM). Each of these approaches has its advantages and disadvantages. Adding only central security and encryption support is not satisfactory, since it would always penalize system performance, and more importantly, it is likely to open new security holes. Database security is a wide research area [6, 3] and includes topics such as statistical database security [21], intrusion detection [19, 4], and most recently privacy preserving data mining [13], and related papers in designing information systems that protect the privacy and ownership of individual information while not impeding the flow of information, include [13, 14, 5, 8]. Users wishing to access data will now securely access it using the privacy management infra structure instead of developing multiple customized solutions for each application and data storage system. Applications and databases would not be impacted by an application specific implementation. This would alleviate the problem of maintaining the software and administrating privacy for each specific application and database.

4.1 Performance considerations

We studied the industry standard SQL benchmark [15] as a model for workloads. Some simple sample tests on Oracle and DB2. The first benchmark was focus on a particular customer scenario. Subsequent benchmarks used a workload combined from multiple customer case studies. The technological aspects of

developing database privacy as an enterprise IT infrastructure component lead to new research challenges. First and fore-most is the issue of encryption key management. Most corporations view their data as a very valuable asset. The key management system would need to provide sufficient security measures to guard the distributed use of encryption keys. We propose a combined hardware and software based data encryption system as the solution to this problem. A distributed policy and audit capability is proposed for the control the use of different encryption keys. Detailed investigation of this solution is presented below. Since the interaction between the database and the enterprise IT infrastructure component there are potential over-heads introduced by encryption. Therefore the sources of performance degradation and its significance should be determined.

4.2 Network Attached Encryption

The Network Attached Encryption (NAED) is implemented as a Network Attached Encryption Appliance that scales with the number of Network Attached Encryption Appliances available. A NAED is a hardware device that resides on the network, houses the encryption keys and executes all crypto operations. This topology has the added security of physically separating the keys from the data. However, this added security comes with a heavy price; performance can be 10-100 times worse than alternative methods. The benchmarks showed a throughput of between 440 and 1,100 row-decryptions per second. A system with twelve database servers performed at 4,200 row-decryptions per second with five Network Attached Encryption Appliances. In prior work with IBM Research [46] we addressed some critical performance issues when using HSM support. A coming paper will address how to avoid the problems of performance and scalability when using HSM support, and also how to prevent API level attacks when using HSM support, including Network Attached Encryption Appliances.

There are three points of overhead with this topology. Let's explore a simple example to demonstrate the overhead; a user requests 500,000 rows of encrypted data:

When a user requests secured data, the security system manages the process of retrieving encrypted data from the database, ensuring that the request is from an authorized user, and performing the decryption process. In this topology, the encryption agent handles the request and retrieves the encrypted data from the database. It sends the encrypted data over the network to be decrypted by the NAED. Inside the NAED are the keys and the algorithms to decrypt the data. However once decrypted, we have clear-text information that needs to be sent back over the wire to the database server. This requires that we re-secure the information for transit, typically through a secure communication process such as SSL. When the data arrives at the agent on the database server, it has to be returned to clear-text, and then it is served up to the calling application.

1. A NAED topology has three points of encryption versus one for other methods. In the example above, the 500,000 rows of data are sent over the wire to be decrypted at the NAED. The clear text is then encrypted using SSL to send back over the network and decrypted at the database to be served in clear text to the application.
2. Network overhead is caused by sending all 500,000 rows over the network to be decrypted by the NAED and then must return over the network to the database.
3. The NAED is a stateless device and needs to be initialised/set-up before each row is decrypted. In this simple example, the NAED is set-up 500,000 times. The set-up has a large overhead.

The Network Attached Encryption Device (NAED) topology has proven in tests, due to the three points of overhead, to perform by an order of magnitude, worse than alternative structures. Each round trip over the network is roughly 1 millisecond per row. In the example above this would be 500,000 x 1ms = 500 seconds compared to 1-25 seconds with alternative topologies.

This example debunks a well-publicized myth, that NAEDs off-load work from the database. There isn't an off-load of work since this solution must perform one encryption operation in the database, which is the same for other topologies, in addition to the encryption functions at the NAED.

4.3 The Hybrid System

This topology combines the enhanced performance of the Software structure with the added security of a hardware device. A HSM, in some situations, is an ideal way to add additional protection for the most important element of any encryption solution – the encryption keys. HSM devices are fast and tamper proof, so they make an excellent vault to store the crown jewels – the encryption keys.

The performance in this topology is essentially identical to the earlier pure software structure, with an occasional transit to the HSM to refresh and retrieve the master encryption keys. During the majority of processing time, performance is identical to the software solution. In our 500,000-row example, in contrast to the NAED structure - where all 500,000 rows flowed over the wire to the NAED - the encryption service in the database server accesses the key from the HSM one time and thereafter all crypto operations are completed in the database by the software encryption service.

The Hybrid system is implemented as distributed processes that scales with the number of processors and database server available. In the Software topology the database server becomes the platform for encryption services, removing the network and a remote device from the equation. When the application calls for

secure information, the encryption service requests the encrypted data from the database server, performs a local decryption, and returns clear-text information to the calling application. All network overhead and encryption (e.g. SSL) has been eliminated from the critical path, optimising the response time and throughput. In addition, since it is not using a separate hardware device there isn't any set-up overhead. In our example, the decryption of the 500,000 rows is handled within the database server. Due to the reduction in encryption points, elimination of network traffic, and set-up overhead, the software topology provides superior performance. In our example of 500,000 rows, the performance is greatly improved: 500,000 x .05ms = 25* seconds.

Some preliminary benchmarks with SQL Server showed a typical throughput in the range of 3,000 to 32,000 row-decryptions per second, depending on a optimised combination of column level encryption and table level encryption, and the amount of cached table data, The initial SQL Server 2000 test used a low-end test system running Windows with a 1.6 GHz processor, 1 GB Physical RAM, and 3 GB Virtual RAM.

Our DB2 benchmarks at IBM showed a typical throughput of 187,000 row-decryptions per second, with 20 concurrent users. This translates to an ability to decrypt 187,000 database table rows per second. The test tables included 80 bytes of encrypted data per row. We saturated all six RS6000 processors at 100% utilization when we tested with 1,000 concurrent users.

Some additional benchmarks with DB2, and Oracle showed a typical throughput in the range of 66,000 to 110,000 row-decryptions per second, on a two processor, 3 GHz system with 3 GB RAM, running a Windows operating system. The benchmarks also showed a linear scalability of this topology when adding additional database servers. A system with twelve database servers performed at 2,100,000 row-decryptions per second.

4.4 Additional tuning to limit the number of crypto operations

As noted in the discussion above on topology alternatives, each crypto operation adds overhead. There are many techniques, supported in mature solutions, which limit the number of operations required. Use of these techniques could mean the difference between acceptable and unacceptable performance overhead.

4.4.1 Search on encrypted data without first decrypting to clear text

Advanced techniques have been developed, similar to indexing, that enables encrypted data to be searched without the overhead of first decrypting into clear text. Only the result set is decrypted to clear text. Many vendor solutions still require the data to be decrypted before being searched. This creates a large increase in the number of crypto operations and therefore hinders performance. Additional tuning by adding an accelerated search index for encrypted columns

reduced the response-time and the number of rows to decrypt, by a factor between 10 and 30 for some of the queries in our Oracle test. This can be viewed as enabling a 'virtual throughput' in the range of 660,000 to 1,100,000 'virtual row-decryptions' per second, when comparing to a solution that is not using an accelerated search index for encrypted columns.

Searching for an exact match of an encrypted value within a column is possible, provided that the same initialisation vector is used for the entire column. On the other hand, searching for partial matches on encrypted data within a database can be challenging and can result in full table scans if support for accelerated index-search on encrypted data is not used. Encrypted columns can be a primary key or part of a primary key, since the encryption of a piece of data is stable (i.e., it always produces the same result), and no two distinct pieces of data will produce the same cipher text, provided that the key and initialisation vector used are consistent. However, when encrypting entire columns of an existing database, depending on the data migration method, database administrators might have to drop existing primary keys, as well as any other associated reference keys, and re-create them after the data is encrypted. For this reason, encrypting a column that is part of a primary key constraint is not recommended if support for accelerated index-search on encrypted data is not used. Since primary keys are automatically indexed there are also performance considerations, particularly if support for accelerated index-search on encrypted data is not used. A foreign key constraint can be created on an encrypted column. However, special care must be taken during migration. In order to convert an existing table to one that holds encrypted data, all the tables with which it has constraints must first be identified. All referenced tables have to be converted accordingly. In certain cases, the referential constraints have to be temporarily disabled or dropped to allow proper migration of existing data. They can be re-enabled or recreated once the data for all the associated tables is encrypted. Due to this complexity, encrypting a column that is part of a foreign key constraint is not recommended, if automated deployment tools are not used. Unlike indexes and primary keys, though, encrypting foreign keys generally does not present a performance impact.

Indexes are created to facilitate the search of a particular record or a set of records from a database table. Plan carefully before encrypting information in indexed fields. Look-ups and searches in large databases may be seriously degraded by the computational overhead of decrypting the field contents each time searches are conducted if accelerated database indexes are not used. This can prove frustrating at first because most often administrators index the fields that must be encrypted – social security numbers or credit card numbers. New planning considerations are needed when determining what fields to index; if accelerated database indexes are not used. Indexes are created on a specific column or a set of columns. When the database table is selected, and WHERE conditions are provided, the database will typically use the indexes to locate the records, avoiding the need to do a full table scan. In many cases searching on an

encrypted column will require the database to perform a full table scan regardless of whether an index exists. For this reason, encrypting a column that is part of an index is not recommended, if support for accelerated index-search on encrypted data is not used.

4.4.2 Limit the types and amount of data (e.g. column level) that are encrypted

An important step in policy definition is identifying the data that needs to be protected and the data that can remain in clear text. Many organizations skip this indispensable step and encrypt more data than required. Although it may seem harmless, and provide added protection, encrypting more data than absolutely needed will come with a price paid in performance. Column level encryption should be implemented to support the organization's data security policy.

4.4.3 Perform operations without decrypting the data

Similar to searching on encrypted data, advanced solutions perform operations such as joins on encrypted data, without the requirement to convert to clear text, therefore preserving the performance of the system.

4.4.4 Query rewrite to improve encryption overhead

We implemented limited support for rewrite of a query, and experienced significant optimisation capabilities when searching on encrypted columns. A method for common sub-expression elimination (CSE) needs to be applied to expensive user defined functions for a query. Common sub-expression detection and elimination are well known in compiler optimisation [1] [9]. An occurrence of an expression is a common sub-expression (CS) if there is another occurrence of the expression whose evaluation always precedes this one in execution order and if the operands of the expression remain unchanged between the two evaluations [9].

5 ENCRYPTION KEY MANAGEMENT

One of the essential components of encryption that is often overlooked is key management - the way cryptographic keys are generated and managed throughout their life. Because cryptography is based on keys that encrypt and decrypt data, your database protection solution is only as good as the protection of your keys. Security depends on two factors: where the keys are stored and who has access to them. When evaluating a data privacy solution, it is essential to include the ability to securely generate and manage keys. This can often be achieved by centralizing all key management tasks on a single platform, and effectively automating administrative key management tasks, providing both operational efficiency and reduced management costs. Data privacy solutions should also include an automated and secure mechanism for key rotation,

replication, and backup. Today's complex and performance sensitive environments require the use of a combination of software cryptography and specialized cryptographic chipsets, HSM, to balance security, cost, and performance needs. One easy solution is to store the keys in a restricted database table or file. But, all administrators with privileged access could also access these keys, decrypt any data within your system and then cover their tracks. Your database security in such a situation is based not on industry best practice, but based on an honour code with your employees. If your human resources department locks employee records in filing cabinets where one person is ultimately responsible for the keys, shouldn't similar precautions be taken to protect this same information in its electronic format? All fields in a database do not need the same level of security. With tamper-proof hardware and software implemented, the encryption being provided by different encryption processes utilizing at least one process key in each of the categories master keys, key encryption keys, and data encryption keys, the process keys of different categories being held in the encryption devices; wherein the encryption processes are of at least two different security levels, where a process of a higher security level utilizes the tamper-proof hardware device to a higher degree than a process of a lower security level; wherein each data element which is to be protected is assigned an attribute indicating the level of encryption needed, the encryption level corresponding to an encryption process of a certain security level. With such a system it becomes possible to combine the benefits from hardware and software based encryption. The software-implemented device could be any data processing and storage device, such as a personal computer. The tamper-proof hardware device provides strong encryption without exposing any of the keys outside the device, but lacks the performance needed in some applications. On the other hand the software-implemented device provides higher performance in executing the encryption for short blocks, in most implementations [26], but exposes the keys resulting in a lower level of security.

5.1.1 Secure encryption of short blocks

When using CBC (Cipher Block Chaining) mode of a block encryption algorithm, a rotating, or random generated initialisation vector is used to provide secure encryption of short blocks and must be stored for future use when the data is decrypted. The IV works like a second key, but does in most cases not need to be kept secret and can be stored in clear in the database. If the application requires having an IV per column, which can be necessary to allow for searching within that column, the value can be stored in a separate table. For a more secure deployment, but with limited searching capabilities if support for accelerated index-search on encrypted data is not used, an IV can be generated per row and stored with the data. In the case where multiple columns are encrypted, but the table has space limitations, the same IV can be reused for each encrypted value in the row, even if the encryption keys for each column are different, provided the encryption algorithm and key size are the same.

5.2 Controlling the use of encryption keys

Current commercial RDBMSs support many different kinds of identification and authentication methods, password-based authentication [32], host-based authentication [24, 32, 31], PKI (Public Key Infrastructure) based authentication [39], third party-based authentications such as Kerberos [37], DCE (Distributed Computing Environment [43]) and smart cards [40]. Essentially, all methods rely on a secret known only to the connecting user. It is vital that a user should have total control over her/his own secret. For example, only she/he should be able to change her/his password. Other people can change a user's password only if they are authorized to do so. In a DB system, a DBA can reset a user's password upon the user's request, probably because the user might have forgotten her/his password. However the DBA can temporarily change a user's password without being detected and caught by the user, because the DBA has the capability to update (directly or indirectly) the system catalogs. This is discussed in more detail in [18].

5.3 A separated security policy

A data directory consists of many catalog tables and views. It is generally recommended that users (including DBAs) do not change the contents of a catalog table manually. Instead, those catalogs will be maintained by the DB server and updated only through the execution of system commands. However, a DBA can still make changes in a catalog table if she/he wants to do so. To prevent unauthorized access to important security-related information, we introduce the concept of security catalog. A security catalog is like a traditional system catalog but with two security properties: It can never be updated manually by anyone, and its access is controlled by a strict authentication and authorization policy. This is discussed in more detail in [18].

6 AUDITABILITY

Technically, if we allow a DBA to control security without any restriction, the whole system becomes vulnerable because if the DBA is compromised, the security of the whole system is compromised, which would be a disaster. However if we have a mechanism in which each user could have control over their own secrecy, the security of the system is maintained even if some individuals do not manage their security properly. Access control is the major security mechanism deployed in all RDBMSs. It is based upon the concept of privilege. A subject (i.e., a user, an application, etc.) can access a database object if the subject has been assigned the corresponding privilege. Access control is the basis for many security features. Special views and stored procedures can be created to limit users' access to table contents. However, a DBA has all the system privileges. Because of their ultimate power, a DBA can manage the whole system and make it work in the most efficient way. However, they also have the capability to do the most damage to the system. With a separated

security directory the security administrator sets the user permissions. Thus, for a commercial database, the security administrator (SA) operates through separate middle-ware, the Access Control System (ACS), used for authentication, verification, authorization, audit, encryption and decryption. The ACS is tightly coupled to the database management system (DBMS) of the database. The ACS controls access in real-time to the protected fields of the database. Such a security solution provides separation of the duties of a security administrator from a database administrator (DBA). The DBA's role could for example be to perform usual DBA tasks, such as extending table spaces etc, without being able to see (decrypt) sensitive data. Thus, it is important to further separate the DBA's and the SA's privileges. For instance, if services are outsourced, the owner of the database contents may trust a vendor to administer the database. The DBA role then belongs to an external person, while the important SA role is kept within the company, often at a high management level. Thus, there is a need for preventing a DBA to impersonate a user in an attempt to gain access to the contents of the database. The method comprises the steps of: adding a trigger to the table, the trigger at least triggering an action when an administrator alters the table through the database management system (DBMS) of the database; calculating a new password hash value differing from the stored password hash value when the trigger is triggered; replacing the stored password hash value with the new password hash value. Similar authentication verification may also be implemented if VPN (Virtual Private Network) based connection and authentication is used.

7 CONCLUSION

We addressed performance as a particularly vital problem and evaluated different solutions for database encryption. In this paper, we discussed the Hybrid, a database privacy solution built on top of all major relational databases. The Hybrid model introduces many significant challenges primary of which are the additional overhead of searching on encrypted data an infrastructure to guarantee data privacy, and management of such an enterprise IT infrastructure component. We have addressed these issues. Our experiments using several benchmarks showed that the overhead is tolerable when using suitable encryption architecture. The Hybrid model implements a scalable approach for data privacy and security in which a security administrator protecting privacy at the level of individual fields and records, and providing seamless mechanisms to create, store, and securely access databases. Such a model alleviates the need for organizations to purchase expensive hardware, deal with software modifications, and hire professionals for encryption key management development tasks. We proposed, implemented, and evaluated different encryption schemes. We showed the drastic decrease in query execution times from distributed software level encryption. We believe, from our experience, database privacy as an infrastructure service is a viable model and has a good chance of emerging as a successful offering for most applications.

References

[1] A. Aho, S. Johnson, and J. Ullman. Code generation for expressions with sub-expressions. Journal of ACM, Jan., 1977.

[2] G. Davida, D. Wells, and J. Kam. A database encryption system with subkeys. ACM Transactions on Database Systems, 6(2), 1981.

[3] DES. Data encryption standard. FIPS PUB 46, Federal Information Processing Standards Publication, 1977.

[4] T. F. Lunt. A survey of intrusion detection techniques. Computer & Security, 12(4), 1993.

[5] Agrawal,J.Kiernan,R.Srikant,andY.Xu.Implementing P3P using database technology. In Proc. of the 19th Int'l Con-ference on Data Engineering, Bangalore, India, March 2003.

[6] G. Hamilton and R. Cattell. JDBC: A Java SQL API. http://splash.javasoft.com/jdbc/.

[7] J. He and M. Wang. Encryption in relational database management systems. In Proc. Fourteenth Annual IFIP WG 11.3 Working Conference on Database Security (DBSec'00), Schoorl, The Netherlands, 2000.

[8] R.Agrawal,J.Kiernan,R.Srikant,andY.Xu.AnXPath-based preference language for P3P. In Proc. of the 12th Int'l World Wide Web Conference, Budapest, Hungary, May 2003.

[9] S. Muchnick. Advanced Compiler Design and Implementa-tion. Morgan Kaufmann Publishers, 1997.

[10] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[11] B. Schneier. Description of a new variable-length key, block cipher (blowfish), fast software encryption. In Cambridge Security Workshop Proceedings, pages 191–204, 1994.

[12] B. Schneier. Applied Cryptography. John Wiley & Sons, Inc., 1996.

[13] R. Agrawal and J. Kiernan. Watermarking relational databases. In 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.

[14] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In Proc. of the 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.

[15] SQL. Benchmark Specification. http://www.tpc.org.

[16] A. F. Westin. Freebies and privacy: What net users think. Technical report, Opinion Research Corporation, http://www.privacyexchange.org/iss/surveys/sr990714.html, 1999.

[17] N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. ACMComputing Surveys,21(4):515– 556, Dec. 1989.

[18] Mattsson, Ulf T.,' A DATABASE ENCRYPTION SOLUTION ', LinuxSecurity.com, 28 July 2004, http://www.linuxsecurity.com/content/view/116068/65/

[19] Mattsson, Ulf T., Social Science Research Network, 'A Real-time Intrusion Prevention System for Commercial Enterprise Databases', http://papers.ssrn.com/sol3/papers.cfm?abstract_id=482282
[20] Mattsson, Ulf T., Search Security and Techtarget http://searchsecurity.techtarget.com/whitepaperPage/0,293857,sid14_gci1014677,00.html
[21] N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. ACMComputing Surveys, 21(4):515– 556, Dec. 1989.
[22] R. Agrawal and J. Kiernan. Watermarking relational databases. In 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.
[23] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In Proc. of the 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.
[24] Agrawal,J.Kiernan,R.Srikant,andY.Xu.Implementing P3P using database technology. In Proc. of the 19th Int'l Con-ference on Data Engineering, Bangalore, India, March 2003.
[25] R.Agrawal,J.Kiernan,R.Srikant,andY.Xu.AnXPath-based preference language for P3P. In Proc. of the 12th Int'l World Wide Web Conference, Budapest, Hungary, May 2003.
[26] D. E. Denning. Cryptography and Data Security. Addison-Wesley Publishing Company, Inc., 1982.
[27] T. Dierks and C. Allen. The TLS Protocol - Version 1.0, Internet-Draft. November 1997.
[28] A. Freier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0, Internet-Draft. November 1996.
[29] S. Garnkel and G. Spa ord. Web Security & Commerce. O'Reilly & Associates, Inc., 1997.
[30] S. B. Guthery and T. M. Jurgensen. Smart Card Developer's Kit. Macmillan Technical Publishing, 1998.
[31] Informix. Informix-Online Dynamic Server Administrator's Guide, Version 7.1. INFORMIX Software, Inc., 1994.
[32] G. Koch and K. Loney. Oracle8: The Complete Reference. Osborne/McGraw-Hill, 1997.
[33] J. C. Lagarias. Pseudo-random number generators in cryptography and number theory. InCryptology and Computational Number Theory, pages 115{143. American Mathematical Society, 1990.
[34] T. F. Lunt. A survey of intrusion detection techniques. Computer & Security, 12(4), 1993.
[35] National Bureau of Standards FIPS Publication 180. Secure Hash Standard, 1993.
[36] National Bureau of Standards FIPS Publication 46. Data Encryption Standard (DES), 1977.
[37] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. IEEE Communications, 32(9):33{38, 1994.
[38] San Jose Mercury News. Web site hacked; cards being canceled, Jan. 20, 2000.

[39] Oracle Technical White Paper. Database Security in Oracle8i, November 1999.
[40] W. Rankl and W. E_ng. Smart Card Handbook. John Wiley & Sons Ltd, 1997.
[41] R. Rivest. The MD5 Message-Digest Algorithm, RFC1321 (I). April 1992.
[42] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public key cryptosystems. Communications of the ACM, 21:120{126, February 1978.
[43] W. Rosenberry, D.Kenney, and G. Fisher. Understanding DCE. O'Reilly & Associates, Inc.,1992.
[44] A. Shamir. How to share a secret. Communication of the ACM, 22(11):612{613, 1979.
[45] D. R. Stinson. Cryptography; Theory and Practice. CRC Press, Inc., 1995.
[46] M. Lindemann and SW Smith, Improving DES Hardware Throughput for Short Operations, IBM Research Report, 2001, http://www.research.ibm.com/secure_systems_department/projects/scop/papers/rc21798.pdf