PROVIDING DATABASE ENCRYPTION AS

A SCALABLE ENTERPRISE INFRASTRUCTURE SERVICE

Protecting against External and Internal Threats

Ulf T. Mattsson, CTO Protegrity

Abstract: As databases become networked in more complex multi-tiered applications, their vulnerability to external attack grows. We address scalability as a particularly vital problem and propose alternative solutions for data encryption as an enterprise IT infrastructure component. In this paper, we explore a new 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. Although access control has been deployed as a security mechanism almost since the birth of large database systems, many still look at database security as a problem to be addressed as the need arises – this is often after threats to the secrecy and integrity of data have occurred. Instead of building walls around servers or hard drives, a protective layer of encryption is provided around specific sensitive data items or objects. This prevents outside attacks as well as infiltration from within the server itself. This also allows the security administrator to define which data stored in databases are sensitive and thereby focusing the protection only on the sensitive data, which in turn minimizes the delays or burdens on the system that may occur from other bulk encryption methods. Encryption can provide strong security for data at rest, but developing a database encryption strategy must take many factors into consideration. Different stored data encryption strategies are outlined, so you can decide the best practice for each situation, and each individual field in your database, to handle different security and operating requirements. Application code and database schemas are sensitive to changes in the data type and data length. the paper presents a policy driven solution that allows transparent data level encryption that does not change the data field type or length.

Keywords: Isolation, Intrusion Tolerance, Database Security, Encryption, Privacy, VISA CISP, GLBA, HIPAA.

1. INTRODUCTION

Although access control has been deployed as a security mechanism almost since the birth of large database systems, for a long time security of a DB was considered an additional problem to be addressed when the need arose, and after threats to the secrecy and integrity of data had occurred [23]. Now many major database companies are adopting the loose coupling approach and adding optional security support to their products. You can use the encryption features of your Database Management System (DBMS), or perform encryption and decryption outside the database. Each of these approaches has its advantages and disadvantages. Adding security support as an optional feature 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 [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].

2. DATA PRIVACY

Encryption is the perfect technique to solve this problem. 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. There are two 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].

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. How about while being processed in the application itself? particularly if the application may cache the data for some period. Sending sensitive information over the Internet or within your corporate network as clear text, defeats the point of encrypting the text in the database to provide data privacy. Good security practice protects sensitive data in both cases – as it is transferred over the network (including internal networks) and at rest. Once the secure communication points are terminated, typically at the network perimeter, secure transports are seldom used within the enterprise. Consequently, information that has been transmitted is in the clear and critical data is left unprotected. This is discussed in more detail in [18].

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.

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.

2.1 Encryption scheme alternatives

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.

2.2 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.

2.3 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.

2.4 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.

2.4.1 Query rewrite to improve encryption overhead

We implemented limited support for rewrite of a query, and experienced significant optimization 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 optimization [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].

3. SCALABILITY OF DIFFERENT ENCRYPTION ARCHITECTURES

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.

3.1 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.

3.2 Network Attached Encryption

The Network Attached Encryption is implemented as a Network Attached Encryption Appliance that scales with the number of Network Attached Encryption Appliances available. The benchmarks showed a throughput of between 440 and 1,100 row-decryptions per second. The benchmarks showed a linear scalability of this topology when adding additional database servers. 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.

3.3 The Hybrid System

The Hybrid system is implemented as distributed processes that scales with the number of processors and database server available. The Hybrid solution can also utilize an optional HSM in a way that allows the total encryption system to scale with the number of processors available on the database servers. 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.
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.

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 mix 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. Additional details from the ongoing benchmarks will be discussed in a coming paper.

3 POLICY MANAGEMENT

Current commercial RDBMSs support many different kinds of identification and authentication methods, among them are:

- 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].

3.4 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].

4. 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 tablespaces 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 a 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. A similar authentication verification may also be implemented if VPN (Virtual Private Network) based connection and authentication is used.

The first security-related component in an RDBMS (and actually in most systems) is user management. A user account needs to be created for anyone who wants to access database resources. However, how to maintain and manage user accounts is not a trivial task. User management includes user account creation, maintenance, and user authentication. A DBA is responsible for creating and managing user accounts. When a DBA creates an account for user Alice, they also specify how Alice is going to be authenticated, for example, by using a database password. The accounts and the related authentication information are stored and maintained in system catalog tables. When a user logs in, they must be authenticated in the exact way as specified in their account record. However, there is a security hole in this process. A DBA can impersonate any other user by changing (implicitly or explicitly) the system catalogs and they can do things on a user's behalf without being authorized/detected by the user, which is a security hole. A DBA's capability to impersonate other users would allow them to access other users' confidential data even if the data are encrypted.

Searching for an exact match of an encrypted value within a column is possible, provided that the same initialization 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. One approach to performing partial searches, without prohibitive performance constraints and without revealing too much sensitive information, is to apply an HMAC to part of the sensitive data and store it in another column in the same row, if support for accelerated index-search on encrypted data is not used. For example, a table that stores encrypted customer email addresses could also store the HMAC of the first four characters of each email address. This approach can be used to find exact matches on the beginning or end of a field. One drawback to this approach is that a new column needs to be added for each unique type of search criteria. So if the database needs to allow for searching based on the first four characters as well as the last five characters, two new columns would need to be added to the table. However, in order to save space, the HMAC hash values can be truncated to ten bytes without compromising security in order to save space. This approach can prove to be a reasonable compromise especially when combined with non-sensitive search criteria such as zip code, city, etc. and can significantly improve search performance 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.

When using CBC (Cipher Block Chaining) mode of a block encryption algorithm, a randomly generated initialisation vector is used and must be stored for future use when the data is decrypted. Since the IV does not need to be kept secret it can be stored 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. 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.

6. CONCLUSION

We addressed scalability as a particularly vital problem and propose alternative solutions for data encryption as an enterprise IT infrastructure component. In this paper, we introduced 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 a 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. A forthcoming paper will discuss performance aspects in more detail, transparent storage and search of encrypted database fields.

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 Publicatio*n, 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 Netherland*s, 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-ti*on. 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 AC*M, 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 Proceeding*s, 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