A Real-time Intrusion Prevention System for Commercial Enterprise Databases

Ulf Mattsson,
Chief Technology Officer
Protegrity, Incorporated.
1010 Washinton Blvd,
Stamford, CT 06901

Abstract

Modern intrusion detection systems are comprised of three basically different approaches, host based, network based, and a third relatively recent addition called procedural based detection. The first two have been extremely popular in the commercial market for a number of years now because they are relatively simple to use, understand and maintain. However, they fall prey to a number of shortcomings such as scaling with increased traffic requirements, use of complex and false positive prone signature databases, and their inability to detect novel intrusive attempts. This intrusion detection systems represent a great leap forward over current security technologies by addressing these and other concerns. This paper presents an overview of our work in creating a true database intrusion detection system.

Based on many years of Database Security Research, the proposed solution detects a wide range of specific and general forms of misuse, provides detailed reports, and has a low false-alarm rate. Traditional database security mechanisms are very limited in defending successful data attacks. Authorized but malicious transactions can make a database useless by impairing its integrity and availability. The proposed solution offers the ability to detect misuse and subversion through the direct monitoring of database operations inside the database host, providing an important complement to host-based and network-based surveillance. Suites of the proposed solution may be deployed throughout a network, and their alarms man-aged, correlated, and acted on by remote or local subscribing security services, thus helping to address issues of decentralized management. Inside the host, the proposed solution is intended to operate as a true security daemon for database systems, consuming few CPU cycles and very little memory and secondary storage. The proposed Intrusion Prevention Solution is managed by an access control system, with intrusion detection profiles, with item access rates and associating each user with profiles. Further, the method determines whether a result of a query exceeds any one of the item access rates defined in the profile associated with the user, and, in that case, notifies the access control system to alter the user authorization, thereby making the received request an unauthorized request, before the result is transmitted to the user. The method allows for a real time prevention of intrusion by letting the intrusion detection process interact directly with the access control system, and change the user authority dynamically as a result of the detected intrusion.

The method is also preventing an administrator impersonating a user of a relational database, which database at least comprises a table with at least a user password, wherein the password is stored as a hash value. 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; and replacing the stored password hash value with the new password hash value.

In this paper, the design of the first MATTSSONHYBRID prototype, which is for Oracle Server 8.1.6, is discussed. MATTSSONHYBRID uses triggers and transaction profiles to keep track of the items read and written by transactions, isolates attacks by rewriting user SQL statements, and is transparent to end users. The MATTSSONHYBRID design is very general. In addition to Oracle, it can be easily adapted to support many other database application platforms such as IBM DB2, Microsoft SQL Server, Sybase, and Informix.

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

1. Introduction

Part of the problem lies in the fact that most companies solely implement perimeter-based security solutions, even though the greatest threats are from internal sources. Additionally, companies implement network-based security solutions that are designed to protect network resources, despite the fact that the information is more often the target of the attack. Recent development in information-based security solutions addresses a defense-in-depth strategy and is independent of the platform or the database that it protects. As organizations continue to move towards digital commerce and electronic supply chain management, the value of their electronic information has increased correspondingly and the potential threats, which could compromise it, have multiplied. With the advent of networking, enterprise-critical applications, multi-tiered architectures and web access, approaches to security have become far more sophisticated. A span of research from authorization [9, 28, 14], to inference control [1], to multilevel secure databases [33, 31], and to multi-level secure transaction processing [3], addresses primarily how to protect the security of a database, especially its confidentiality. However, very limited research has been done on how to survive successful database attacks, which can seriously impair the integrity and availability of a database. Experience with data-intensive applications such as credit card billing, has shown that a variety of attacks do succeed to fool traditional database protection mechanisms. One critical step towards attack resistant database systems is intrusion detection, which has attracted many researchers [7, 21, 13, 10, 23, 26, 22, 17, 18]. Intrusion detection systems monitor system or network activity to discover attempts to disrupt or gain illicit access to systems. The methodology of intrusion detection can be roughly classed as being either based on statistical profiles [15, 16, 30] or on known patterns of attacks, called signatures [11, 8, 27, 12, 32]. Intrusion detection can supplement protection of network and information systems by rejecting the future access of detected attackers and by providing useful hints on how to strengthen the defense. However, intrusion detection has several inherent limitations: Intrusion detection makes the system attack-aware but not attack-resistant, that is, intrusion detection itself cannot maintain the integrity and availability of the database in face of attacks. Achieving accurate detection is usually difficult or expensive. The false alarm rate is high in many cases. The average detection latency in many cases is too long to effectively confine the damage. To overcome the limitations of intrusion detection, a broader perspective is introduced, saying that in addition to detecting attacks, countermeasures to these successful attacks should be planned and deployed in advance. In the literature, this is

referred to as survivability or intrusion tolerance. In this paper, we will address a useful technique for database intrusion prevention, and present the design of a practical system, which can do attack prevention.

2. Problem Description

In order to protect information stored in a database, it is known to store sensitive data encrypted in the database. To access such encrypted data you have to decrypt it, which could only be done by knowing the encryption algorithm and the specific decryption key being used. The access to the decryption keys could be limited to certain users of the database system, and further, different users could be given different access rights.
Specifically, it is preferred to use a so-called granular security solution for the encryption of databases, instead of building walls around servers or hard drives. In such a solution, which is described in this paper, 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.
Most preferably the encryption is made on such a basic level as in the column level of the databases. Encryption of whole files, tables or databases is not so granular, and does thus encrypt even non-sensitive data. It is further possible to assign different encryption keys of the same algorithm to different data columns. With multiple keys in place, intruders are prevented from gaining full access to any database since a different key could protect each column of encrypted data.

2.2. New Requirements

The complexity of this task was dramatically increased by the introduction of multi-platform integrated software solutions, the proliferation of remote access methods and the development of applications to support an increasing number of business processes. In the "good old days", files and databases contained fewer types of information (e.g., payroll or accounting data) stored in centralized locations, which could only be accessed, by a limited number of individuals using a handful of controlled access methods. As more types of information were migrated to electronic formats (and ever more databases proliferated, often with little planning), there was a simultaneous increase in the number of users, access methods, data flows among components and the complexity of the underlying technology infrastructure. Add to this the demand from users for ever more sophisticated uses of information (data mining, CRM, etc.) which are still evolving and the management's enhanced awareness of the value of its information, and It is safe to say that the price of poker has gone up. Database intrusion tolerance can mainly be enforced at two possible levels: operating system (OS) level and transaction level. Although transaction level methods cannot handle OS level attacks, it is shown that in many applications where attacks are enforced mainly through malicious transactions transaction level methods can tolerate intrusions in a much more effective and efficient way. Moreover, it is shown that OS level intrusion tolerance techniques such as those proposed in [23, 22, 24, 25, 4], can be directly integrated into a transaction level intrusion tolerance frame-work to complement it with the ability to tolerate OS level attacks.

The importance of privacy and security of sensitive data stored in relational databases is fueled by strong new legislation and the continuing push toward Web-accessible data. Protegrity's and Protegrity's products and services allow organizations to comply with data-privacy regulations, requirements and guidelines such as the recently enacted U.S. Gramm-Leach-Bliley Act (GLBA)…significantly affecting financial institutions and insurance companies; the U.S. Health Information Portability and Accountability Act (HIPAA)…covering the healthcare industry; the European Directive 95/46/EC on data protection…and E.U./U.S. Safe Harbor considerations; Canada's Personal Information Protection and Electronic Document Act (PIPEDA); Germany's Federal Data Protection Act; the UK Data Protection Act; Australia's Privacy Act); the Japan JIS Q 15001:1999 Requirements for Compliance Program on Personal Information Protection; the U.S. Software and Information Industry Association (SIIA) -An Electronic Citadel **-** A Method for Securing Credit Card and Private Consumer Data in E-Business Sites; the BITS (the technology group for the Financial Services Roundtable) Voluntary Guidelines for Aggregation Services; and potentially much more.

3. Solution Overview

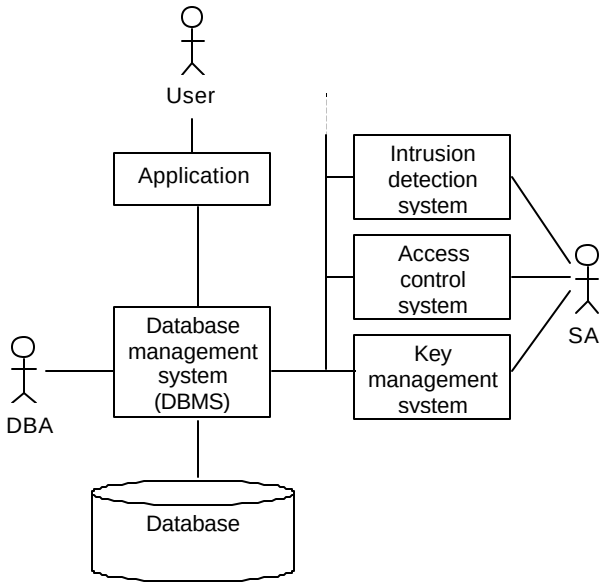

Fig. A schematic view of a system preventing attacks on a relational database.

In the above mentioned solutions the security administrator is responsible for setting the user permissions. Thus, for a commercial database, the security administrator operates through a middle-ware, the access control system (ACS), which serve for authentication, 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 elements 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. The SA could then administer privileges and permissions, for instance add or delete users.
For most commercial databases, the database administrator has privileges to access the database and perform most functions, such as changing password of the database users, independent of the settings by the system administrator. An administrator with root privileges could also have full access to the database. This is an opening for an attack where the DBA can steal all the protected data without any knowledge of the protection system above. The attack is in this case based on that the DBA impersonates another user by manipulating that users password, even though the user's password is enciphered by a hash algorithm. An attack could proceed as follows. First the DBA logs in as himself, then the DBA reads the hash value of the users password and stores this separately. Preferably the DBA also copies all other relevant user data. By these actions the DBA has created a snapshot of the user before any altering. Then the DBA executes the command "ALTER USER username IDENTIFIED BY newpassword". The next step is to log in under the user name "username" with the password "newpassword" in a new session. The DBA then resets the user's password and other relevant user data with the previously stored hash value.
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. Then the role of the DBA 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.

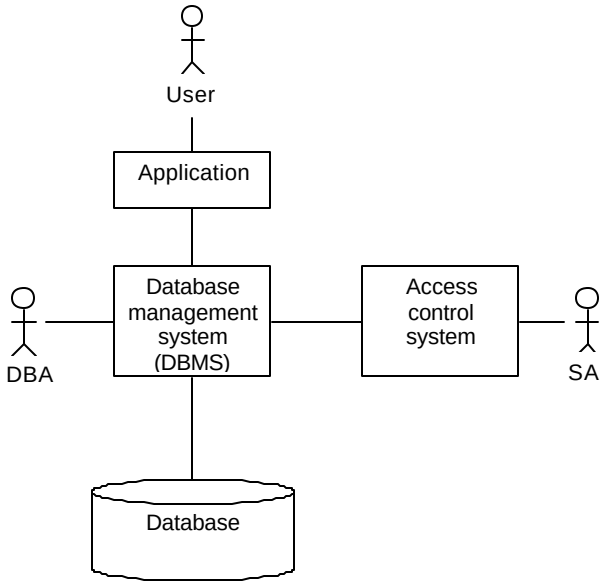

Fig. A schematic view of a system preventing an administrator impersonating a user of a relational database.

2.2. A New Approach to Support New Requirements

The solution protects the data in storage in a database. The architecture is built on top of a traditional COTS (Commercial-Of-The-Shelf) DBMS. Within the framework, the Intrusion Detector identifies malicious transactions based on the history kept (mainly) in the log. The Intrusion Assessor locates the damage caused by the detected transactions. The Intrusion Protector prevents the damage using some specific cleaning of field level transactions. The Intrusion Manager restricts the access to the objects that have been identified by the Intrusion Assessor as 'under attack', and unlocks an object after it is cleared by the security officer. The Policy Enforcement Agent (PEA) (a) functions as a filter for normal user transactions that access critical fields in the database, and (b) is responsible for enforcing system-wide intrusion prevention policies. For example, a policy may require the PEA to reject every new transaction submitted by a user as soon as the Intrusion Detector finds that the user submits a malicious transaction. It should be noticed that the system is designed to do all the intrusion prevention work on the fly without the need to periodically halt normal transaction processing.

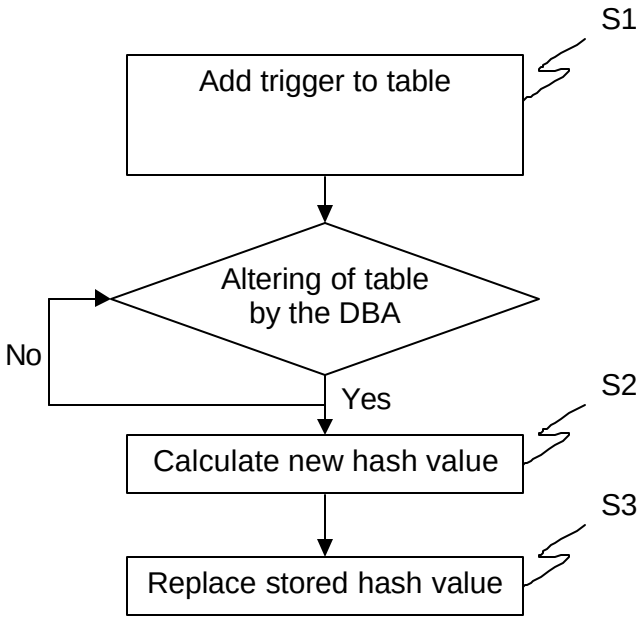

Fig. Flow-chart illustrating a process preventing an administrator impersonating a user of a relational database.

2.3. Summary of the method for preventing an administrator impersonating a user

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. Hereby, a method is provided, which overcomes the above mentioned problems. With such a method the database administrator (DBA) can not impersonate a user. Impersonation means that the DBA steals the identity of an user, and is able to act in the name of the user, preferably while the user is unaware of the impersonation. Even though the DBA still can

read the encrypted password and replace it, the attempt to impersonate a user will be detected and measures can be taken.
Preferably, the method comprises the further steps of:

- calculating a control value of the trigger, such as a hash value; and
- comparing the trigger at the startup and at regular intervals with a recalculated control value. With these additional steps the DBA can not even try to modify the trigger and thereby manipulate the impersonation prevention method.

With the method above the intrusion is detected when a user tries to log in, since the hash value of the users password will not match. In order to detect intrusion earlier the method can preferably comprise the further step of comparing for each active user having access to sensitive data, the hash value of the current login password with the currently stored password hash value, whereby the step is performed after every change of the database content by the user. In one implementation, the trigger comprises means for reading a log of actions on the database, means for identifying commands for altering of user passwords in the log and means for identifying which user passwords that have been changed. Preferably the trigger is a daemon process. Also according to the method a impersonation prevention system for a relational database preventing an administrator impersonating another user, which database at least comprises a table with at least a user password, wherein the password is stored as a hash value, the system comprises: calculation means for calculating a hash value of a user password; trigger means, which trigger at least the calculation means for calculation of a new hash value of the password when an administrator alters the table through the database management system (DBMS) of the database; and replacing means for replacing the stored hash value with the new hash value for each triggered calculation. Such a system will overcome the risk for a DBA impersonating a user with all the advantages as the method previously described.

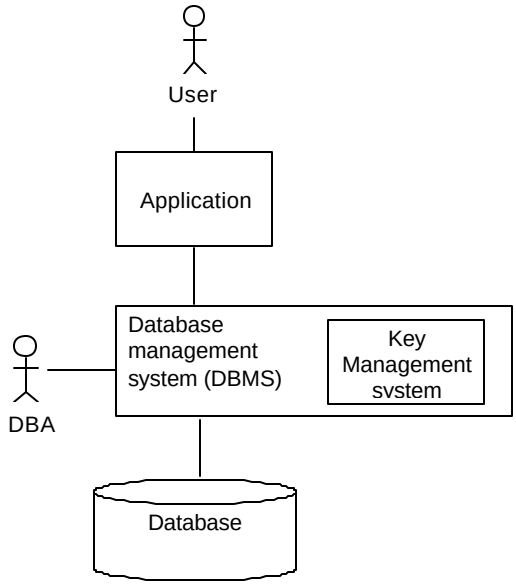

Fig. Encryption Keys exposed in the database environment.

The proposed solution will also prevent database encryption keys to be exposed in the application environment.

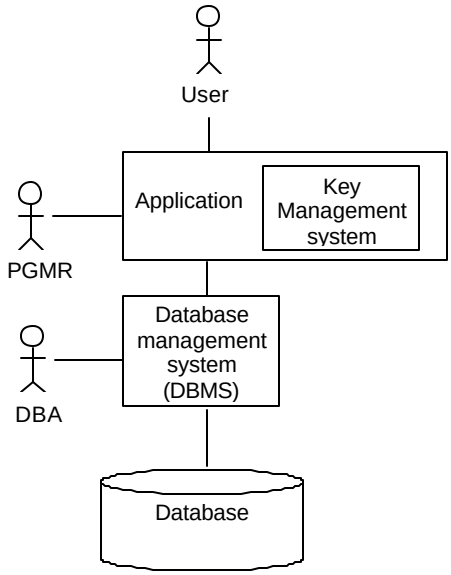

Fig. Encryption Keys exposed in the application environment.

3. A Hybrid Solution

A hybrid solution combines security technologies from several areas and provide a cost effective solution for some of the new privacy requirements. The hybrid solution will minimize the impact at the application level and combine the strengths and separation of duties from external security systems with the benefits from tight database integration. The hybrid solution will also provide Database Encryption functionality minimizing the performance impact by monitoring only the information that's critical from a security point of view, instead of entire databases. Privacy and Security Mandates, and other business requirements, will define what information that require this higher level of protection and audit.

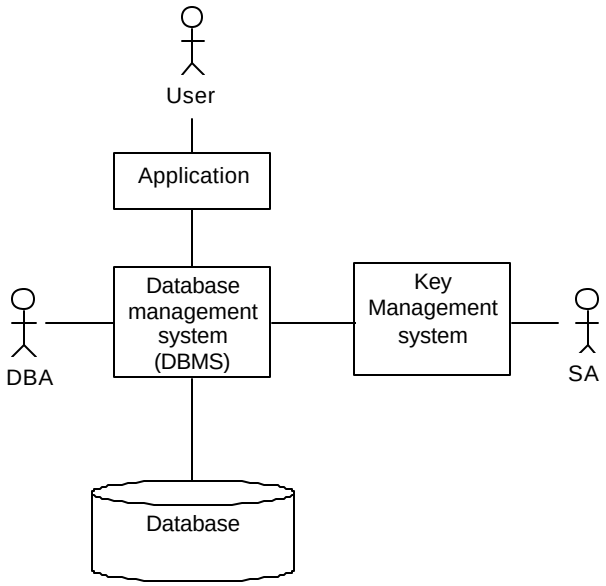

Fig. Encryption Keys managed securely in separate from the database environment

3.1. Secure Monitoring of Management Functions and Information Access

The foundation of the security audit function is a secure reporting and audit facility combined with organizational separation of security from administrative responsibilities. The creation of logs that track activities performed by security officers and unauthorized access attempts to protected data is a critical element. These logs must track information regarding the use of sensitive data, including records of user reads and updates. Managers can use this information to track trends, analyze potential threats, support future security planning, and assess the effectiveness of countermeasures. Ideally, the logs should focus on the most useful information for security managers; that is, activity around protected information. Focusing only on sensitive information minimizes the performance degradation and also maximizes the usefulness of the protected security audit log. Auditing isn't an all-or-nothing exercise; it should be selective. Selective and granular auditing saves time and reduces performance concerns by focusing on sensitive data only. By limiting accumulation of audit logs to only sensitive information, more critical security events are highlighted and reviewed. This solution allows the auditing strategy to be based on knowledge about application or database activity around sensitive data, in an effort to protect their own employees from being wrongfully suspected in the case of an internal breach. The log should contain all relevant operations on critical data elements, and contain security related information needed in the case of a breach. For example a security log should be tamper proof and provide evidence on who read what information and when.

4. Key Management

Guaranteeing that unauthorized users cannot access data ensures data privacy. Encryption is the primary solution for ensuring data privacy, trust, and verification in Internet banking services. Encryption is basically defined as the protection of information by converting it into a form that is unintelligible until it is converted back to its original form. Encryption must be employed in all cases where customers can perform, or authorized users are provided with access, to transactions that involve confidential information within the target system's database. All Secure.Data key management activities are automatically logged and adequate information maintained such that all key management processing can be reviewed.

4.1. Automated Encryption Key Management and Encryption Key Escrow

Cryptographic keys are in multiple component form, and controlled by more than one key custodian, where one custodian can never learn or know any other key component but their own. Split knowledge is enforced under which two or more parties must separately and confidentially have custody of components of a single key. Different security administration roles are responsible for different functions, so that there is a clear separation of duties. Allows encryption key escrow by a secure backup of master keys. The Secure.Data solution provides secret symmetric or private asymmetric keys and can be managed by two key custodians who do not have knowledge of the each other's keys or key components. When was the last time that internal information access controls in your financial institution were independently validated? These are not idle questions. Weak internal controls played a role in all three of the national banks that failed in 1999. In one case, improper record keeping and accounting contributed to the bank's failure. In the second case, the bank lacked adequate external audit. All encryption keys are generated using traditional seeding principles or optional hardware based generators. The optional hardware based and automated encryption key management provides a tamper evident environment, FIPS 140 Level 3, to protect the most sensitive encryption keys from exposure. Additional mechanisms for protection for the encryption keys in memory and in the database are used. A number of different types of keys are also used to achieve a high level of security:

Master Keys -There is one Master Key per server and one per Manager. All other keys (i.e. communication keys, internal keys and application data keys) generated on that entity (Server/Manager) are encrypted using the Master Key. The Master Key is stored in the database protected with software based or optional hardware based encryption.

Communication Keys – Communication key is negotiated with Diffie-Hellman between the Server and Manager at the first connection ("push" or "synchronize" of the policy). The communication key is unique to each Manager-Server pair, and all traffic between the Server and Manager is encrypted under that key. The communication key in the current implementation is triple-DES keys. Both parties have a copy of the communication key after the negotiation. The key is stored locally encrypted under the Master Key for each entity.

Internal (Policy) Keys - Internal Keys are used to encrypt / decrypt policy information in the policy database that the entity keeps.

Data Keys - Data Keys are used to encrypt / decrypt application data associated with Item/Objects in the policy. All Data Keys are stored encrypted in the Secure.Data Server policy database. Data key uniqueness depends on the properties of the Item/Object. If the property “unique key” is chosen, the key is unique. If “base key” is selected for an Item/Object, the key protecting that data will be shared with the other Item/Objects with the “base key” option selected.

3.3. Secure Key Management Implementation

The Secure Key Management system for encryption of individual data elements comprising of encryption devices of at least two different types, the types being tamper-proof hardware and software implemented. The encryption processes of the system are of at least two different security levels, differing in the type of encryption device holding the process keys for at least one of the process key categories and also differing in which type of device executing the algorithm of the process. Each data element to be protected is assigned an attribute indicating the usage of encryption process of a certain security level.

3.4 A Combined Hardware and Software Based Encryption System

In order to protect information stored in a database, it is known to store sensitive data encrypted in the database. To access such encrypted data you have to decrypt it, which could only be done by knowing the encryption algorithm and the specific decryption key being used. The access to the decryption keys could be limited to certain users of the database system, and further, different users could be given different access rights.

Specifically, it is advantageous to use a so-called granular security solution for the encryption of databases, instead of building walls around servers or hard drives. In such a solution, which is described in this paper, 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. Most preferably the encryption is made on such a basic level as in the column level of the databases. Encryption of whole files, tables or databases is not so granular, and does thus encrypt even non-sensitive data. It is further possible to assign different encryption keys of the same encryption algorithm to different data columns. With multiple keys in place, intruders are prevented from gaining full access to any database since a different key could protect each column of encrypted data. In present systems for such granular protection of data, the encryption process is performed within hardware. Using a tamper-proof hardware for protection of the algorithms and the keys results in a strong protection. One purpose of such a system is to provide data elements with different degrees of protection. However, when encrypting small blocks of data, such as individual data records in a database, a hardware encryption device could experience performance problems. Thus, even though granular encryption techniques on data elements in databases provides flexibility on the encryption level, this flexibility is not sufficient for commercial purposes. For example, in a application with increasing amounts of data and/or data processing, it could be of interest to significantly reduce the security level when encrypting for example older data, while maintaining a higher security level when encrypting new data. This would result in

increased overall performance. The current solutions do not provide a sufficient flexibility, which forces the operator to invest in additional hardware resources in order to maintain the systems overall performance. Current hardware encryption systems utilizes a tamper-proof hardware device for encrypting the data elements. The hardware device's processing capability is dependent on the device's processor, memory, architecture, etc. The only way, without changing the device's hardware configuration, to increase a system's performance utilizing such a device, is to use simpler encryption algorithms, for instance reduce the key length etc. However, the reduction of encryption security level reaches a level where the used processing power does not decrease proportionally, since the initial overhead for each access to the tamper-proof hardware will still be constant. Therefore, such systems experience a performance problem when faced to increased load and when encryption of data elements requiring lower protection increases.

6.4. Summary of the key management method

According to the method a relational database system for encryption of individual data elements comprises a plurality of encryption devices being of at least two different types, the types being 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.

Hereby, a system is provided, which overcomes the above mentioned problems. With such a system it becomes possible to combine the benefits from hardware and software based encryption. The tamper-proof hardware device could for example be a device with a security level 4 as described in the Federal Information Processing Standard (FIPS) Publication 140-1 developed by the National Institute of Standards and Technology (NIST) or any equivalent, the publication hereby included by reference. 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 algorithms, but exposes the keys resulting in a lower level of security. The present method takes advantage of the fact that all data elements in a database do not need the same level of encryption.

With such a system it becomes possible to rapidly change the system's security levels and performance, respectively. For example, when an attack is detected, it will be possible to easily raise the security levels for a selection of data elements. In another situation, for instance in an electronic commerce system, the performance for a part of the online store could swiftly be increased by changing the security level of a selection of data elements. In a preferred implementation a process of a first higher security level essentially utilizes the tamper-proof hardware device and a process of a second lower security level essentially utilizes the software implemented device. Preferably, the encryption processes differ in the type of encryption device holding the process keys for at least one of the process key categories and also in which type of device executing the algorithm of the process. However, this is only one way of configuring such a system.

The system includes encryption process of a first security level having the tamper-proof hardware device for holding the process keys for the process key categories master keys, key encryption keys, and data encryption keys, and the tamper-proof hardware device for executing the encryption algorithm of the first security level process; and
an encryption process of a second security level having the tamper-proof hardware device for holding the process keys for the process key categories master keys and key encryption keys, and the software implemented device for holding the at least one process key of the process key category data encryption keys, and the software implemented device for executing the encryption algorithm of the second security level process. The first encryption process should then be used for the most sensitive data. The second encryption process utilizes both the tamper-proof hardware device and the software implemented device in order to encrypt data. The tamper-proof hardware device holds all but the data encryption keys, which are checked-out from the tamper-proof hardware device. Thus, the tamper-proof hardware device holds the master key and the key encryption keys are not exposed outside the hardware device. The data processing and storage device now use the checked-out data encryption key for encryption of a data element. Encryption by the software implemented device is most advantageous for small blocks of data. Preferably the attributes for short data blocks, 8-16 bytes of data, are automatically set to use the second encryption algorithm.
In another implementation, the system comprises a third security level having the software implemented device for holding the process keys for the process key categories master keys, key encryption keys, and data encryption keys, and the software implemented device for executing the encryption algorithm of the third security level process. Using a third encryption process for some data elements could even further improve the performance of the system, since it probably will reduce the load on the tamper-proof hardware device. Preferably the attributes also comprises information about initialization vectors and length of the encryption key. In one implementation the system further comprises a key caching feature. This is useful when a large number of different keys are used on short blocks in order to increase the performance of the system. For example, the key is cached the first time it is decrypted and used inside the tamper-proof hardware device. Also according to the method a method for encryption of individual data elements in relational database system, wherein the system comprises a plurality of encryption devices being of at least two different types, the types being tamper-proof hardware and software implemented, comprises the steps of:
providing encryption processes of at least two different security levels, where a process of a first higher security level essentially utilizes the tamper-proof hardware device and a process of a second lower security level essentially utilizes the software implemented device; assigning an data element which is to be protected an attribute indicating the level of encryption needed, the encryption level corresponding to an encryption process of a certain security level; choosing an encryption process correlating to the security level assigned to the data element which is to be protected; encrypting, using chosen encryption process, the data element which is to be protected. Hereby, a method is provided, which overcomes the above mentioned problems. With such a method it becomes possible to combine the benefits from hardware and software based encryption. Referring to fig. 1, a schematic view of a system according to an implementation of the method is illustrated. The system comprises a tamper-proof hardware device 1, a software implemented device 2, which are used for encrypting data elements in a relational database 3. The software implemented device is as previously described any data processing and storage device. The term software implemented is to be understood an opposite to the tamper-proof hardware device. For example, the software implemented device could be a traditional personal computer, having a

microprocessor for executing the algorithms and where the different keys and algorithms are stored on a storage media connected thereto, such as a hard disk. The storage media could be organized as a relational database with a database management system, and the keys stored in the database. Upon request from the system, according to the method, the keys and algorithms would then be read from the storage media into the working area of a random access memory. There, the microprocessor of the software implemented device would process a data element of the relational database 3 in order to obtain an encrypted data element. The tamper-proof hardware device 1 holds a master key 4, key encryption keys 5 and data encryption keys 6. The tamper-proof hardware device 1 has mechanisms for executing encryption algorithms. As an example, and not limited to, the tamper-proof hardware could be a multi-chip embedded module, packaged in a PCI-card. In addition to cryptographic hardware, and circuitry for tamper detection and response, it could include a general-purpose computing environment: a 486-class CPU (99 Mhz in Model 2), executing software stored in ROM and in FLASH. The multiple-layer software architecture preferably comprises foundational security control (Layer A and Layer B), supervisor-level system software (Layer C), and user-level application software (Layer D). The Layer C component is designed to support application development. Within Layer C, a kernel provides standard OS abstractions of multiple tasks and multiple address spaces. Then the software implemented device is a multiple-layer software architecture comprising foundational security control (Layer A and Layer B), basic crypto functions software (Layer C), and user-level application software (Layer D). The software implemented device 2 also holds another set of keys; one software master key 7, software key encryption keys 8 and software data encryption keys 9. By software keys 7, 8 and 9 are meant keys stored in the software implemented device 2. The relational database system 2 comprises data elements organized in tables with rows and columns. Each data element have an attribute, which describes the security level of the data element, for example in a scale from A-C. The security level could then represent different encryption processes, and preferably further information about the encryption process. Such information could comprise where the keys are stored, which encryption algorithms to use, where to execute the algorithm, key values, key length or an initialization vector, etc. An example of an algorithm that could be used for an encryption process is DES with ECB, in CBC mode with rotating IV. The processes according to the implementation differ in their security level. An example of the implementation of respective encryption process security levels are given in table 1 below.

	Security level A	Security level B	Security level C
Storage of master key	H/W	H/W	S/W
Storage of key encryption keys	H/W	H/W	S/W
Storage of data encryption keys	H/W	S/W	S/W
Execution of encryption algorithm	H/W	S/W	S/W

Table 1: Example of security levels

According to table 1, a data element having an attribute stating security level A, will have the strongest protection. Then, none of the keys will be exposed outside the tamper-proof hardware 1 and the encryption process will take place within the tamper-proof hardware 1. A data element with security level B, will check-out (preferably by decrypting and exporting), a data encryption key 6 from the tamper-proof hardware 1 to the software implemented device 2 and use it a software encryption key 9. This data encryption key 9 will then be used by an encryption algorithm processed in the software implemented device 2 described above. After processing the data encryption key 9 will be stored in the software implemented device 2 for later decryption. Finally, data elements requiring a not so strong protection will have the attribute security level C. This means that they all the keys involved the crypto-process are stored in the software implemented device 2, where also the encryption process takes place. The method has been described above in terms of a preferred implementation. However, the scope of this method should not be limited by this implementation, and alternative implementations of the method are feasible, as should be appreciated by a person skilled in the art. For example, the software keys 7,8 and 9 could be stored in the same database as the data elements that are subject for encryption. Such implementations should be considered to be within the scope of the method, as it is defined by the appended claims.

4. Implementation based on Oracle

A high level of application transparency can be accomplishes by providing a view that corresponds with the original physical table being protected. Without any changes to the application or any knowledge from the end-user, all queries to the original table are now being handled by the integration view. All access to the underlying (encrypted) data is handled by this implemented view. In a database environment protected by the Secure.Data encryption services, direct or indirect access to a view with an attached security policy causes the data-server always to consult the policy function for verification. The policy function returns only authorized data, dynamically modifying the external user's data access. The example is based on a simple table 'tab'. The original base table 'tab' holds an identity 'id' column and a secret code column 'secret':

id	secret
1	a
2	b

Fig. Example based on a simple table 'tab'.

Create the new base table 'tab_enc' that will hold encrypted values in the 'secret' column:

```
create table tab_enc (
        id integer,
```

secret varbinary (32));

id	secret
1	#
2	%

Fig. new base table ‘tab_enc’
that will hold encrypted values.

Create a view with the same name as the original base table ‘tab’, and create a triggers on the view ‘tab’ to be able to insert, update, and delete data:

```
create or replace view tab(id, secret) as
        SELECT id, decrypt('item secret', secret)
        FROM tab_enc
create or replace trigger tab_insert
instead of insert on tab
        for each row
        begin
        insert into tab (
                id,
                secret)
                values (
                        :new.id,
                        pty.ins_encrypt(‘item_secret’, :new.secret));
        end;

create or replace trigger tab_update
instead of update on tab
        for each row
        begin
        update tab set
                        id = :new.id,
                        secret = pty.upd_encrypt(‘item_secret’, :new.secret))
                where id = :old.id;
        end;

create or replace trigger tab_delete
instead of delete on tab
        for each row
        begin
                pty.del_check(‘item_secret’);
                delete tab
                where id = :old.id;
        end;
```

Fig. Create view, and triggers

5. Deployment on a 24-by-7 operational database system

In most commercial applications accessibility is a critical issue, and customers expect a service to be accessible when they want to use it. Hereby method is provided which significantly improves the uptime of a database system. With this method the database owner easily can alter encryption settings in the database while it is up and running. Since a rerouting of the access is provided, data will always be accessible. Thus, the security administrator (SA) can independently of any constraints regarding when the database has to be up add or remove encryption when it is needed. For example, if a security leak is found in a web-application such as an Internet store during rush hours, the management of that company would with previous solutions have had to decide whether to risk sales or risk that someone would intrude in their system gaining access to unencrypted data in the database. This is eliminated with the method according to the implementation. Another advantage is that regular maintenance work can be performed during daytime, reducing the need for costly overtime since the maintenance personnel don't have to work when the database can be taken offline, which mostly is during night hours. This is a method which allows altering of encryption status in a relational database in a continuous process, which significantly reduces or eliminates the need for making the database unavailable or only partly available, overcoming the above mentioned problems. The method comprises the steps of: copying all records from a base area to a maintenance area; directing action of commands intended for the base area to the maintenance area; altering encryption status of the base area; and copying all data records from the maintenance area to the base area; and redirecting action of commands to the base area.

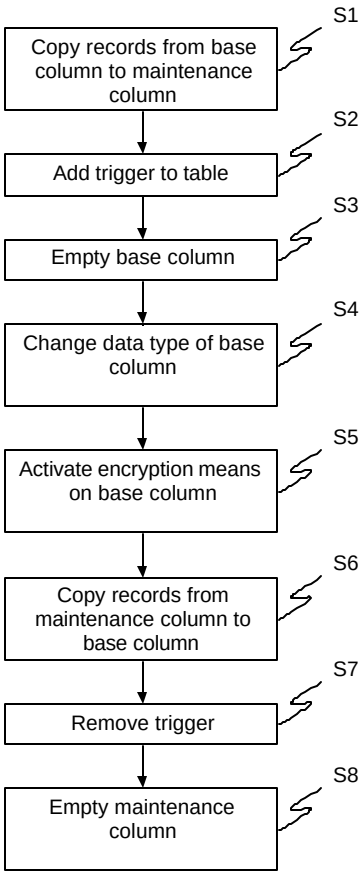

Fig. Process flow for altering encryption status

The term encryption status is to be understood as how to protect data elements in the base area, for instance whether or not the data elements are subject for encryption. In another implementation it could also be understood as changing the encryption level, from strong to weak. If the purpose is to remove encryption for data elements in the base area, the data elements are decrypted while they are copied to the maintenance area. Then, if the purpose if to add encryption to data elements, they are encrypted as they are copied to, or from, the maintenance area. Then, when the data elements are temporarily stored in the maintenance area, the settings could be changed for the base area. The database which is described comprises one or more tables. Action of commands could for example be reading commands resulting in a read operation, or a write command resulting in a write operation. Preferably, the step of directing is implemented in a trigger which is added to the table. In an implementation of the present method the commands are data manipulation language (DML) statements. In an implementation of the present method each base area in the database table have a corresponding maintenance area. In an implementation of the present method the method comprises the further step of emptying the base area before the step of altering. Preferably this done by updating all the records of the column with NULL. In an implementation of the present method the method comprises the further step of changing the data type of the base area. Preferably, this is changed to RAW. In an implementation of the present method the base area is a first column of the table and the

maintenance area is a second column of the table. However, the method is not limited to this interpretation of an area, for example an area could comprise a set of columns. According to another implementation of the method a method for altering encryption status in a relational database in a continuous process, wherein at least one table of the database comprises at least one base area, and for each base area a corresponding area, comprising the steps of: activating encryption means for the corresponding column; directing action of commands intended for the base area to the maintenance area; copying all records from the base area to the corresponding area; and emptying the base area. Hereby a method is provided which, in addition to the above mentioned advantages, allows continuous encryption on tables that have explicit locks i.e. row exclusive (RX) or share row exclusive (SRX) locks. The tables I and II below illustrates an example of a database table, "tab", for which encryption is to be added to a column. Table I describes the structure of the database table "tab" and Table II is an example of the contents in such a table.

Data element	Data type	Value	Comment
cust_id	NUMBER	NOT NULL	Primary key
name	VARCHAR2(64)	NOT NULL	
date_of_birth	DATE	NOT NULL	
user_name	VARCHAR2(32)	NOT NULL	
password	VARCHAR2(32)	NOT NULL	To be encrypted
maint	VARCHAR2(32)	NULL	

Table I

cust_id	name	date_of_birth	user_name	password	maint
1001	MAX	19910101	MNN	abc	NULL
1002	MARTIN	19920202	MKR	cdf	NULL
1003	JOHAN	19930303	JON	ghi	NULL
1004	MARIE-LOUISE	19940404	MLA	jkl	NULL

Table II

The method comprises a first step S1, wherein data is copied from the base column "password" to the maintenance column "maint". The contents of "tab" after the step S1 are shown in Table III.

cust_id	name	date_of_birth	user_name	password	maint
1001	MAX	19910101	MNN	abc	abc
1002	MARTIN	19920202	MKR	cdf	cdf
1003	JOHAN	19930303	JON	ghi	ghi
1004	MARIE-LOUISE	19940404	MLA	jkl	jkl

Table III

Preferably, if needed, the method contains a step, which checks whether the column "password" is nullable, i.e. the column does not have a NOT NULL constraint. Then the column is altered to be nullable. In another step S2 a trigger is added. The object of the trigger is to direct all commands aimed at the base column to the maintenance column, i.e. a synchronization function. Thus, when a user for example sends a update command for the base column, this command is directed to the maintenance column. In order to overcome problems during copying and activation of the trigger, the trigger could be built up from several steps. For instance, it could first synchronize the base and the maintenance column, then when the contents are identical, stop updating the base column at the same time let the maintenance column take over the actions taken on the base column. Preferably the copying of the records from the base column is performed simultaneously with the addition of the trigger. In another step S3, the base column "password" is emptied. For instance, this could be performed by updating the base column with NULL. Preferably, if it is required by the later applied encryption, the method comprises the further step S4, wherein the table is altered in order to change the base column data type to the data type RAW. The present structure and contents of "tab" is described in tables IV and V, respectively.

Data element	Data type	Value	Comment
cust_id	NUMBER	NOT NULL	Primary key
name	VARCHAR2(64)	NOT NULL	
date_of_birth	DATE	NOT NULL	
user_name	VARCHAR2(32)	NOT NULL	
password	RAW	NULL	To be encrypted
maintenance	VARCHAR2(32)	NOT NULL	

Table IV

cust_id	name	date_of_birth	user_name	password	maint
1001	MAX	19910101	MNN	NULL	abc
1002	MARTIN	19920202	MKR	NULL	cdf
1003	JOHAN	19930303	JON	NULL	ghi
1004	MARIE-LOUISE	19940404	MLA	NULL	jkl

Table V

Then, the step S5 of activating encryption means is performed. Thus, all data written to the base column "password" will now be written in encrypted form. The means for encryption could be a standard software or hardware, for example a apparatus with a DES algorithm. The data is read from the maintenance column and processed by encryption means. The encryption could be either symmetrical or asymmetrical, for example DES or RSA respectively. After step S5, the records from the maintenance column are copied to the base column through the encryption means in step S6. Thus, the contents of the base column "password" is now stored in an encrypted form. Then the trigger is removed in step S7. This is done in such a manner that synchronization problems are overcome. Preferably the copying of the records from the maintenance column is performed simultaneously with the removal of the trigger. Since the maintenance column now contains unencrypted data, it is important that this column is emptied, which is performed in step S8. This can be performed by either updating the

column with NULL or writing a random value into the column. Then this example table, “tab”, will have the contents as shown in table VI.

cust_id	name	date_of_birth	user_name	password	maint
1001	MAX	19910101	MNN	7je	NULL
1002	MARTIN	19920202	MKR	skj	NULL
1003	JOHAN	19930303	JON	9fj	NULL
1004	MARIE-LOUISE	19940404	MLA	xjr	NULL

Table VI

In order to let the altering of the table have effect on views, the views have to be recreated after each ALTER of a table. An alternative implementation will now be described. The above mentioned implementation is used under the presumption that there are not any table locks (RX/RSX = Row Exclusive/Row Share Exclusive) on the table. In the case of such database locks, additional maintenance columns have to be added in advance. This is preferably performed during installation or planned maintenance, and has not to be done when the actual adding or removing of encryption takes place. Thus, there will be created a maintenance column for each column, which is not currently encrypted. The method according to the alternative implementation is similar to the preferred implementation described above and comprises of the steps: activating encryption means for the maintenance columns corresponding to the base column, which is to be encrypted; adding a trigger to the table, which transfers action of data manipulation language (DML) statements intended for the base column to the maintenance column; copying all records from the base column to the corresponding maintenance column through the encryption means; and emptying the base column.

7. Storage-to-storage encryption for Mobile client applications.

The security policy defines the specific packaging format and encryption method and algorithms for fields when stored in the database and when transported over networks. The DTP (data type preservation) format is an option that is type and length transparent for applications and database schemas. The client side decrypts received data fields, based on the security policy, and provides a secure local storage.

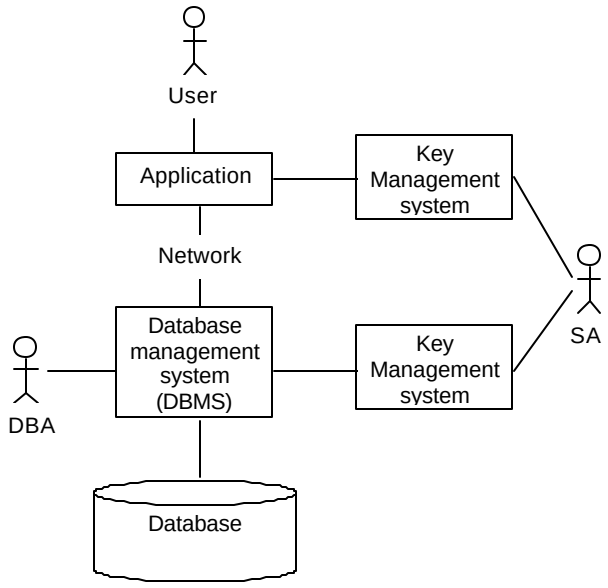

Fig. Storage-to-storage encryption for Mobile client applications.

6. The Intrusion Prevention Functionality

The method allows for a real time prevention of intrusion by letting the intrusion detection process interact directly with the access control system, and change the user authority dynamically as a result of the detected intrusion. The hybrid solution combines benefits from database encryption toolkits and secure key management systems. The hybrid solution also provides a single point of control for database intrusion prevention, audit, privacy policy management, and secure and automated encryption key management (FIPS 140 Level 3). The Database Intrusion Prevention is based on 'context checking' against a protection policy for each critical database column, and prevents internal attacks also from root, dba, or 'buffer overflow attacks'. The Database Intrusion Prevention and alarm system enforces policy rules that will keep any malicious application code in a sand box regarding database access. The policy enforcement includes checking on:

- Session Authentication and Session Encryption.
- Software Integrity, Data Integrity, and Meta Data Integrity.
- Time of Access, and other policy rules.

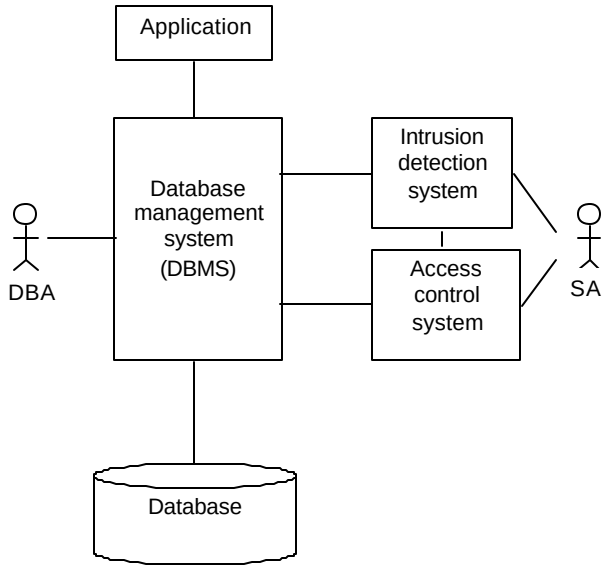

Fig. A schematic view of a intrusion prevention system for a relational database.

In database security, it is a well-known problem to avoid attacks from persons who have access to a valid user-ID and password. Such persons cannot be denied access by the normal access control system, as they are in fact entitled to access to a certain extent. Such persons can be tempted to access improper amounts of data, by-passing the security. Solutions to this problem have been suggested:

Network-Based Detection - Network intrusion monitors are attached to a packet-filtering router or packet sniffer to detect suspicious behavior on a network as they occur. They look for signs that a network is being investigated for attack with a port scanner, that users are falling victim to known traps like .url or .lnk, or that the network is actually under an attack such as through SYN flooding or unauthorized attempts to gain root access (among other types of attacks). Based on user specifications, these monitors can then record the session and alert the administrator or, in some cases, reset the connection. Some examples of such tools include Cisco's NetRanger and ISS' RealSecure as well as some public domain products like Klaxon that focus on a narrower set of attacks.

Server-Based Detection - These tools analyze log, configuration and data files from individual servers as attacks occur, typically by placing some type of agent on the server and having the agent report to a central console. Some examples of these tools include Axent's OmniGuard Intrusion Detection (ITA), Security Dynamic's Kane Security Monitor and Centrax's eNTrax as well as some public domain tools that perform a much narrower set of functions like Tripwire which checks data integrity. Tripwire will detect any modifications made to operating systems or user files and send alerts to ISS' RealSecure product. Real-Secure will then conduct another set of security checks to monitor and combat any intrusions.

Security Query and Reporting Tools - These tools query NOS logs and other related logs for security events or they glean logs for security trend data. Accordingly, they do not operate in real-time and rely on users asking the right questions of the right systems. A typical query might be how many failed authentication attempts have we had on these NT servers in the past two weeks." A few of them (e.g., SecurIT) perform firewall log analysis. Some examples of such tools include Bindview's EMS/NOSadmin and Enterprise Console, SecureIT's SecureVIEW and Security Dynamic's Kane Security Analyst.

6.1. Inference detection

A variation of conventional intrusion detection is detection of specific patterns of information access, deemed to signify that an intrusion is taking place, even though the user is authorized to access the information. A method for such inference detection, i.e. a pattern oriented intrusion detection, is disclosed in US patent 5278901 to Shieh et al. None of these solutions are however entirely satisfactory. The primary drawback is that they all concentrate on already effected queries, providing at best an information that an attack has occurred.

6.2 The intrusion detection profile

By defining at least one intrusion detection profile, each comprising at least one item access rate, associating each user with one of the profiles, receiving a query from a user, comparing a result of the query with the item access rates defined in the profile associated with the user, determining whether the query result exceeds the item access rates, and in that case notifying the access control system to alter the user authorization, thereby making the received request an unauthorized request, before the result is transmitted to the user. According to this method, the result of a query is evaluated before it is transmitted to the user. This allows for a real time prevention of intrusion, where the attack is stopped even before it is completed. This is possible by letting the intrusion detection process interact directly with the access control system, and change the user authority dynamically as a result of the detected intrusion. The item access rates can be defined based the number of rows a user may access from an item, e.g. a column in a database table, at one time, or over a certain period of time. In a preferred implementation, the method further comprises accumulating results from performed queries in a record, and determining whether the accumulated results exceed any one of the item access rates. The effect is that on one hand, a single query exceeding the allowed limit can be prevented, but so can a number of smaller queries, each one on its on being allowed, but when accumulated not being allowed. It should be noted that the accepted item access rates not necessarily are restricted to only one user. On the contrary, it is possible to associate an item access rate to a group of users, such as users belonging to the same access role (which defines the user's level of security), or connected to the same server. The result will be restricting the queries accepted from a group of users at one time or over a period of time. The user, role and server entities are not exclusive of other entities which might benefit from a security policy. According to an implementation of the method, items subject to item access rates are marked in the database, so that any query concerning the items automatically can trigger the intrusion detection process. This is especially advantageous if only a few items are intrusion sensitive, in which case most queries are not directed to such items. The selective activation of the intrusion detection will then save time and processor power. According

to another implementation of the method, the intrusion detection policy further includes at least one inference pattern, and results from performed queries are accumulated in a record, which is compared to the inference pattern, in order to determine whether a combination of accesses in the record match the inference policy, and in that case the access control system is notified to alter the user authorization, thereby making the received request an unauthorized request, before the result is transmitted to the user. This implementation provides a second type of intrusion detection, based on inference patterns, again resulting in a real time prevention of intrusion.

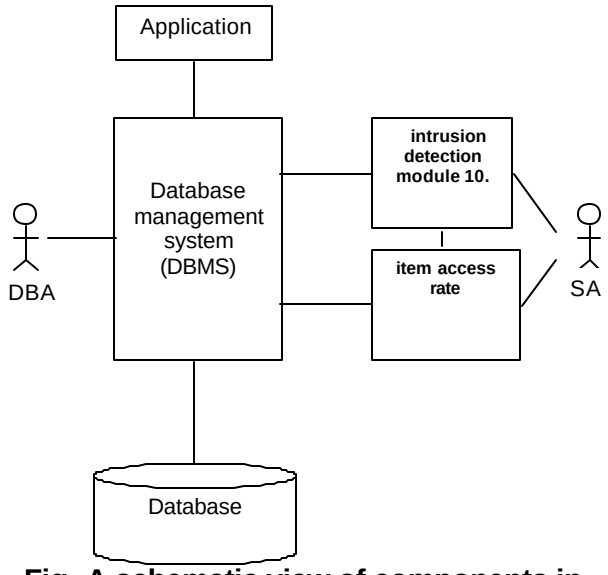

Fig. A schematic view of components in an intrusion prevention system.

6.3. Detailed description of the intrusion prevention system

The present method may be implemented in an environment of the type illustrated in fig 1. The environment comprises a number of clients 1, connected to a server 2, e.g. a Secure.Data™ server from Protegrity, providing access to a database 3 with encrypted data 4. Several clients 1 can be connected to an intermediate server 5 (a proxy server), in which case we have a so called three tier application. Users 6 use the clients 1 to access information 4 in the database 3. In order to verify and authorize attempted access, an access control system (ACS) 7 is implemented, for example Secure.Server™ from Protegrity. The server is associated with an intrusion detection module 10, comprising software components 12, 13 and 18 for performing the method according to the method. Although the intrusion detection module 10 here is described as a separate software module, its components can be incorporated in the server software 2, for example in a security administration system (SAS) 8, like Secure.Manager™ from Protegrity. It can reside in the server hardware 16, or in a separate hardware unit. A first component 12 of the intrusion detection module 10 enables marking of some or all data items (e.g. columns in tables) in the database, thereby indicating that these items should

be monitored during the intrusion detection process, as described below. A second component 13 of the intrusion detection module 10 is adapted to store all results from queries including marked items, thereby creating a record 14 of accumulated access of marked items. If advantageous, the record can be kept in a separate log file 15, for long term storage, accumulating data access over a longer period of time. The server 2 further has access to a plurality of security policies 20, preferably one for each user, one for each defined security role, or the like. These security policies can be stored in the security administration system 8, but also be stored outside the server. Each policy 20 includes one or several item access rates 21 and optionally an inference pattern 22. An item access rate 21 defines the maximum number of rows of the selected item (e.g. column of a table) that a given user, role or server may access during a given period of time. The period of time can be defined as one single query, but can also be an accumulation of queries during a period of time. Preferably, a separate item access rate is defined for at least each item that has been marked in the database 3 by the component 12 of the intrusion detection module 10. An inference pattern 22 defines a plurality of items (columns of certain tables) that when accesses in combination may expose unauthorized information. This means that an attempt by a user, role or server to access certain quantities of information from items in an inference pattern during a given period of time (e.g. in one request) implies that an intrusion is taking place, even if the associated item access rates have not been exceeded. For further information about the inference concept of intrusion, see US 5278901. Returning to the intrusion detection module 10, a third component 18 is adapted to compare the result of a query with an item access rate 21 and an inference pattern 22. The component 18 can also compare the access rates 21 and inference patterns 22 with accumulated results, stored in the record 14 or log file 15. When a user tries to access a database, the access control system 7 completes an authority check of the user. Different routines can be used, including automatic authorization by detecting IP-address, or a standard log-in routine. In one implementation, the authorized user will only have access to items defined in his role, i.e. the table columns that the user is cleared for and uses in his/her work. The access control system 7 then continually monitors the user activity, and prevents the user from accessing columns he/she is not cleared for. This process is described in detail in WO 97/49211, hereby incorporated by reference. The intrusion detection according to the described implementation of the method is directed toward the situation where a user, authorized to access certain items, abuses this authority and tries to obtain information broaching the security policy of the database owner. The intrusion detection is divided into two different stages, a real time stage and an à posteriori analysis stage.

6.4. Real time analysis

With reference to fig 2, a request is received by the server in step S1, resulting in the generation of a result in step S2, i.e. a number of selected rows from one or several table columns. The software component 12 determines (step S3) if any items in the result are marked for monitoring in the database. If no marked items are included in the result, the result is communicated to the user in a standard way (step S4). If, however, mared items are included in the result, the intrusion detection component 13 stores the query result, or at least those parts referring to the marked items, in the record 14, and the program control initiates the intrusion detection (step S6-S10). First, in step S6, the

intrusion detection component 18 compares the current query result and the updated record 14 with the item access rate 21 included in the security policy associated with the current user, the role that the user belongs to, or the server the user is connected to. Note that only item access rates 21 associated with the marked items comprised in the current result need to be compared. If the current query result or accumulated record 14 includes a number of rows exceeding a particular item access rate 21, such a request will be classified as an intrusion (step S7), and the access control system 7 will be alerted (step S10). Secondly, in step S8, if no item access rate is exceeded, the intrusion detection process compares the query result and accumulated record 14 with any inference pattern included in the relevant security policy. If the result includes a combination of items that match the defined inference pattern, such a request will also be classified as an intrusion (step S9), and the access control system will be alerted (step S10). If no intrusion is found in step S7 nor step S9, the program control advances to step S4 and communicates the result to the user. Upon an ACS alert (step S10), the access control system 7 is arranged to immediately alter the user authorization, thereby making the submitted request unauthorized. This can be effected easily, for example if the ACS 7 is part of the Secure.Data™ server from Protegrity. For the user, the request, or at least parts of the request directed to items for which the item access rate was exceeded, will thus appear to be unauthorized, even though authority was initially granted by the access control system 7. In addition to the immediate and dynamic alteration of the access control system 7, other measures can be taken depending on the seriousness of the intrusion, such as sending an alarm to e.g. the administrator, or shutting down the entire database. The server software 11 can send an alarm to a waiting process that a potential breach of security is occurring.

6.5. Long term analysis

The query result can also be stored in the log file 15 by the intrusion detection module, as described above. The log file 15, which thus contains accumulated query results from a defined time period, can also be compared to the inference patterns 22 in the security profiles 20 of users, roles or servers, this time in a “after the event” type analysis. Even though such an analysis cannot prevent the intrusion from taking place, it may serve as intelligence gathering, improving the possibilities of handling intrusion problems. While the real time protection is most efficient when it comes to preventing security breaches, the long term analysis can be more in depth, and more complex, as time is no longer a critical factor. Many three-tier applications (e.g. connections with a proxy 5) authenticate users to the middle tier 5, and then the TP monitor or application server in the middle tier connects to the database 3 as a super-privileged user, and does all activity on behalf of all users 6 using the clients 1. Preferably, the method is implemented in a system, for example Secure.Data™ from Protegrity, in which the identity of the real client is preserved over the middle tier thereby enabling enforcement of ”least privilege” through a middle tier. The intrusion detection module 10 therefore can audit access requested both by the logged-in user who initiated the connection (e.g., the TP monitor), and the user on whose behalf an action is taken. Audit records capture both the user taking the action and the user on whose behalf the action was taken. Auditing user activity, whether users are connected through a middle tier or directly to the data server, enhances user accountability, and thus the overall security of multi-tier systems. Audit records can be sent to the database audit trail or the operating system's audit trail, when the operating system is capable of receiving them. This option, coupled with the broad selection of audit options and the ability to customize auditing with triggers or stored

procedures, provides the flexibility of implementing an auditing scheme that suits any specific business needs.

8. Liability Aspects

This solution provides protection and controls to prevent unauthorized access of the data as well as necessary auditing capabilities that can be used to demonstrate compliance with these new regulations. Other benefits include:

- Compliance with legal standards and requirements to protect the privacy of non-public personal information from internal and external unauthorized access, through selective encryption, separation of duties, and centralized, independent and trusted audit functions for protected information.

- Reduced liability for the Board of Directors and Executive Management, by enabling a dual control security solution that addresses any environment: ASP (Application Services Provider & Aggregation Services Provider), MSP (Managed Services Provider) and B2B (Business-to-Business).

Utilizing the Hybrid Technology for data-privacy will qualify for up to a 40% discount on breach of computer security insurance coverage from a number of insurance companies. Placed with Lloyd's of London, this policy provides the insured broad first party e business protection for highly secure risks. Coverage includes protection against losses resulting from computer hacking, illegitimate use of computer systems and other Information Technology security risks. Below are a few issues executives need to consider:
- Class and individual action suits
- Loss of network/database integrity and availability
- Loss of intellectual capital
- Loss of employee productivity
- Defamation of brand name and reputation

4. Support for Industry specific Data-privacy Regulations

Companies are mandated to comply with industry specific data-privacy regulations best practice requirements and industry guidelines regarding the usage and access to customer data. Privacy requirements for protecting non-public personal information include, selective encryption of stored data, separation of duties and centralized independent audit functions. Some examples of security requirements that mandate specific actions for protecting databases from external and internal intruders:

- U.S. Gramm-Leach-Bliley Act requires financial institutions and their partners to protect non-public personal information by implementing a variety of access and security controls. There are specific requirements relating to administrative, technical, and physical safeguards for customer records and information. Security measures should include management controls that provide effective segregation of duties and restrictions on accessing data. Database auditing is an essential requirement.

7. Cost Aspects of the Solution

The issue for management is how to implement cost-effective and efficient ways to secure these large and valuable assets and their complex infrastructure. Such a solution must also be relatively easy to administer and should provide management with information on historical security performance and potential future steps. There is neither a single data security solution nor perfect security, but the basic tenets of a security solution can readily be identified. Data security requires a coordinated program, which includes threat assessment, strategy development, administration and management of corporate security policies and procedures, initiation of a separate security audit function, implementation of automated security capabilities (firewalls, encryption, incident reporting, etc.) and an ongoing commitment of time and budget from senior management. The security audit function should be supported by strong authentication, protected by encryption, and independent of the database's audit mechanism and administrative procedures. While considerable time and money have been expended in building security to protect networks and servers from external threats, organizations also need to be aware of the need to protect databases from potential threats, including those from within the firewall.

8. Related work

In database security, it is a well-known problem to avoid attacks from persons who have access to a valid user-ID and password. Such persons cannot be denied access by the normal access control system, as they are in fact entitled to access to a certain extent. Such persons can be tempted to access improper amounts of data, by-passing the security. Solutions to this problem have been suggested:

There is a variety of related research efforts that explore what one can do with audit data to automatically detect threats to the host. An important work is MIDAS [50], as it was one of the original applications of expert systems—in fact using P-BEST—to the problem of monitoring user activity logs for misuse and anomalous user activity. CMDS, by SAIC, demonstrated another application of a forward-chaining expert-system, CLIPS, to a variety of operating system logs [48]. USTAT [39] offered another formulation of intrusion heuristics using state transition diagrams [46], but by design remained a classic forward-chaining expert sys-tem inference engine. ASAX [37] introduced the Rule-based Sequence Evaluation Language (RUSSEL) [42], which is tuned specifically for the analysis of host audit trails.

9. Conclusion

While the existing paradigms of computer security are still very useful and serve perfectly well in their capacities, there has existed a gap in the computer security space. Our technology and approach fills that gap by providing procedural based intrusion detection and response. We suggest that this gives Watcher the unique ability to detect and halt completely novel attacks that have yet to be seen on the Internet, and better yet, we have the ability to protect the first person to see a new attack or exploit. No one needs to be sacrificed to the new virus or worm anymore.

In essence, we have learned to solve the right problem. Removing all software vulnerabilities is clearly an unsolvable problem. Providing restrictive and onerous barriers to software use makes the software uncomfortable and difficult to use. Monitoring and controlling program execution at run time through behavioral control is the missing piece in the security puzzle. The complete puzzle has three pieces; data control (encryption), access control, and behavioral control.

In conclusion, while the overall complexity of the security program has dramatically increased, enterprises can still implement effective security solutions by integrating sound external protection and internal security controls with appropriate security audit procedures. There are no guarantees that any one approach will be able to deal with new and innovative intrusions in increasingly complex technical and business environments. However, implementation of an integrated security program which is continuously audited and monitored provides the multiple layers of protection needed to maximize protection as well as historical information to support management decision-making and future policy decisions.

This solution protects the data during transport, providing security from the server to the client. The client device requires a means of accessing the secure data, and a means of access control and secure storage of locally held information. The implementation for Laptops and PDAs provides mandatory access control, secure local storage of sensitive data and key management capabilities. This solution includes a method for detecting intrusion in a database, managed by an access control system, comprising defining at least one intrusion detection profile, each comprising at least one item access rate and associating each user with one of the profiles. Further, the method determines whether a result of a query exceeds any one of the item access rates defined in the profile associated with the user, and, in that case, notifies the access control system to alter the user authorization, thereby making the received request an unauthorized request, before the result is transmitted to the user. The method allows for a real time prevention of intrusion by letting the intrusion detection process interact directly with the access control system, and change the user authority dynamically as a result of the detected intrusion.

The GLBA/OCC and the VISA U.S.A. CISP requirements as well as other requirements in the Health Care Industry, and Safe Harbor will require a unique demonstration of cooperative and open but protected communication, storing information among individuals and organizations across competitive lines and regulatory boundaries safeguarding non-public personal information. Information sharing among reliable and reputable experts can help institutions reduce the risk of information system intrusions. The OCC encourages management to participate in information-sharing mechanisms as part of an effort to detect and respond to intrusion and vulnerabilities. Financial institutions have to work together in an unprecedented fashion with other financial institutions, service providers, software vendors, trade associations, regulators, and other industries to share information and strategies to respond to legal requirements and media reports or perceptions that could decrease public confidence in the financial services industry. With the introduction of regulatory privacy acts like the U.S. Gramm-Leach-Bliley Act, the U.S. HIPAA, the U.S. FDA 21 CFR 11 and the E.U. member states privacy laws, companies are being mandated to provide more detailed information regarding the usage and access of customer and consumer data.

Acknowledgments

The author is grateful to all the people who have contributed to the design, implementation, evaluation and evolution of this solution. I thank Tamojit Das, Christian Olsson, Thomas Valfridsson, and Ulf Dahl at Protegrity Inc., and their teams, for their work on the requirements and development of this solution. I also thank Kurt Lennartsson, and his team at Pointsec, for the work on the client side of this solution. I also thank two major credit card companies in US, and a number of the major banks in US, for sharing their case studies on this topic with us.

References

[1] M. R. Adam. Security-Control Methods for Statistical Database: A Comparative Study. ACM Computing Surveys, 21(4), 1989.
[2] P. Ammann, S. Jajodia, and P. Liu. Recovery from malicious trans-actions.
IEEE Transactions on Knowledge and Data Engineering, 2001. To appear.
[3] V. Atluri, S. Jajodia, and B. George. Multilevel Secure Transaction Processing. Kluwer Academic Publishers, 1999.
[4] D. Barbara, R. Goel, and S. Jajodia. Using checksums to detect data corruption. In Proceedings of the 2000 International Conference on Extending Data Base Technology, Mar 2000.
[5] P. A. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, MA, 1987.
[6] S. B. Davidson. Optimism and consistency in partitioned distributed database systems. ACM Transactions on Database Systems, 9(3):456–581, September 1984.
[7] D.E.Denning. An intrusion-detection model. IEEE Trans. on Software Engineering, SE-13:222–232, February 1987.
[8] T.D. Garvey and T.F. Lunt. Model-based intrusion detection. In Proceedings of the 14th National Computer Security Conference, Balti-more, MD, October 1991.
[9] P. P. Griffiths and B. W. Wade. An Authorization Mechanism for a Relational Database System. ACM Transactions on Database Systems, 1(3):242–255, September 1976.
[10] P. Helman and G. Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. IEEE Transactions on Software Engineering, 19(9):886–901, 1993.
[11] K. Ilgun. Ustat: A real-time intrusion detection system for unix. In Proceedings of the IEEE Symposium on Security and Privacy,Oak-land, CA, May 1993.
[12] K. Ilgun, R.A. Kemmerer, and P.A. Porras. State transition analysis: A rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 21(3):181–199, 1995.
[13] R. Jagannathan and T. Lunt. System design document: Next generation intrusion detection expert system (nides). Technical report, SRI International, Menlo Park, California, 1993.
[14] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 474–485, May 1997.
[15] H. S. Javitz and A. Valdes. The sri ides statistical anomaly detector. In Proceedings IEEE Computer Society Symposium on Security and Privacy, Oakland, CA, May 1991.

[16] H. S. Javitz and A. Valdes. The nides statistical component description and justification. Technical Report A010, SRI International, March 1994.
[17] T. Lane and C.E. Brodley. Temporal sequence learning and data reduction for anomaly detection. In Proc. 5th ACM Conference on Computer and Communications Security, San Francisco, CA, Nov 1998.
[18] Wenke Lee, Sal Stolfo, and Kui Mok. A data mining framework for building intrusion detection models. In Proc. 1999 IEEE Symposium on Security and Privacy, Oakland, CA, May 1999.
[19] P. Liu, S. Jajodia, and C.D. McCollum. Intrusion confinement by isolation in information systems. Journal of Computer Security, 8(4):243–279, 2000.
[20] P. Luenam and P. Liu. Odam: An on-the-fly damage assessment and repair system for commercial database applications. In Proc. 15th IFIP WFG11.3 Working Conference on Database and Application Security, Ontario, Canada, July 2001.
[21] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H. S. Javitz, A. Valdes, P. G. Neumann, and T. D. Garvey. A real time intrusion detection expert system (ides). Technical report, SRI International, Menlo Park, California, 1992.
[22] Teresa Lunt and Catherine McCollum. Intrusion detection and response research at DARPA. Technical report, The MITRE Corporation, McLean, VA, 1998.
[23] T.F. Lunt. A Survey of Intrusion Detection Techniques. Computers & Security, 12(4):405–418, June 1993.
[24] J. McDermott and D. Goldschlag. Storage jamming. In D.L. Spooner, S.A. Demurjian, and J.E. Dobson, editors, Database Se-curity IX: Status and Prospects, pages 365–381. Chapman & Hall, London, 1996.
[25] J. McDermott and D. Goldschlag. Towards a model of storage jamming. In Proceedings of the IEEE Computer Security Foundations
Workshop, pages 176–185, Kenmare, Ireland, June 1996.
[26] B. Mukherjee, L. T. Heberlein, and K.N. Levitt. Network intrusion detection. IEEE Network, pages 26–41, June 1994.
[27] P.A. Porras and R.A. Kemmerer. Penetration state transition analysis: A rule-based intrusion detection approach. In Proceedings of the 8th Annual Computer Security Applications Conference, San Antonio, Texas, December 1992.
[28] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next generation database systems. ACM Transactions on Database Systems, 16(1):88–131, 1994.
[29] P. Liu S. Ingsriswang. Aaid: An application aware transaction level database intrusion detection system. Technical report, Department of Information Systems, UMBC, Baltimore, MD, 2001.
[30] D. Samfat and R. Molva. Idamn: An intrusion detection architecture for mobile networks. IEEE Journal of Selected Areas in Communications, 15(7):1373–1380, 1997.
[31] R. Sandhu and F. Chen. The multilevel relational (mlr) data model. ACM Transactions on Information and Systems Security, 1(1), 1998.
[32] S.-P. Shieh and V.D. Gligor. On a pattern-oriented model for intrusion detection. IEEE Transactions on Knowledge and Data Engi-neering, 9(4):661–667, 1997.
[33] M. Winslett, K. Smith, and X. Qian. Formal query languages for secure relational databases. ACM Transactions on Database Systems, 19(4):626–662, 1994.
[34] The U.S. Health Information Portability and Accountability Act (HIPAA) - compliance by October 2002 www.hipaacomply.com
[35] The European Union 95/46/EC Directive on Data Privacy - compliance October 1998 - and individual EU member state privacy legislation - various compliance dates http://europa.eu.int/comm/internal_market/en/dataprot/
[36] EU/US Safe Harbor - compliance 11/1/2000 www.export.gov/safeharbor

http://europa.eu.int/comm/internal_market/en/dataprot/modelcontracts/index.htm
[37] EU member state privacy legislations see http://europa.eu.int/comm/internal_market/en/dataprot/law/impl.htm
[38] Germany's Federal Data Protection Act (Der Bundesbeauftragte für den Datenschutz) - compliance May 23, 2001 www.bfd.bund.de
[39] Sweden's Personal Data Act (Personuppgiftslagen - PuL) - compliance October 1, 2001 www.datainspektionen.se
[40] UK's Data Protection Act - Compliance March 1, 2000 www.dataprotection.gov.uk
[41] Canada's Personal Information Protection and Electronic Document Act (PIPEDA) Compliance 1/1/2001 to 1/1/2004 www.privcom.gc.ca
[42] Australia's Privacy Act – Compliance by December 21, 2001 www.privacy.gov.au
[43] The VISA U.S.A. Cardholder Information Security Program (CISP) – Compliance May 1, 2001 http://usa.visa.com/business/merchants/cisp_index.html
[44] The VISA International Account Information Security Standards (AIS) and Best Practices Guide https://www.visa.com/nt/gds/main.html
[45] The U.S. Software and Information Industry Association (SIIA) - An Electronic Citadel - A Method for Securing Credit Card and Private Consumer Data in E-Business Sites www.siia.net/sharedcontent/divisions/ebus/citadel.pdf
[46] The BITS (the technology group for the Financial Services Roundtable) Voluntary Guidelines for Aggregation Services www.bitsinfo.org/FinalAggregationBook051601.pdf
[47] The U.S. Gramm-Leach-Bliley Act (GLBA) (TITLE V-Consumer Privacy), regulated by the SEC, FTC, FDIC, OCC, OTS, FRB, NAIC, and NCUA, which covers a broad range of financial services and virtually affects any company who accepts credit cards - compliance July 1st, 2001 www.complianceheadquarters.com/Privacy/Privacy_Research/privacy_research.html
[37] J. Habra, B. Le Charlier, A. Mounji, and I. Mathieu. ASAX: Software architecture and rule-based language for universal audit trail analysis. In Y. Deswarte et al., editors, Computer Security – Proceedings of ESORICS 92, volume 648 of LNCS, pages 435–450, Toulouse, France, Nov. 23–25, 1992. Springer-Verlag.
[38] L. T. Heberlein et al. A network security monitor. In Proceedings of the 1990 IEEE Symposium on Security and Pri-vacy, pages 296–304, Oakland, California, May 7–9, 1990.
[39] K. Ilgun. USTAT: A real-time intrusion detection system for UNIX. In Proceedings of the 1993 IEEE Symposium on Security and Privacy, pages 16–28, Oakland, California, May 24–26, 1993.
[40] U. Lindqvist and P. A. Porras. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In Proceedings of the 1999 IEEE Symposium on Security and Privacy, pages 146–161, Oakland, California, May 9–12, 1999.
[41] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In H. Debar, L. M´ e, and S. F. Wu, editors, Recent Advances in Intrusion Detection (RAID 2000), volume 1907 of LNCS, pages 162–182, Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.
[42] A. Mounji. Languages and Tools for Rule-Based Distributed Intrusion Detection. PhD thesis, Institut d'Informatique, University of Namur, Belgium, Sept. 1997.
[43] P. G. Neumann and P. A. Porras. Experience with EMERALD to date. In Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, Apr. 9–12, 1999. The USENIX Association.
[44] A. One. Smashing the stack for fun and profit. Phrack Magazine, 7(49), Nov. 8, 1996. http://www.fc.net/phrack/files/ p49/p49-14.

[45] J. Picciotto. The design of an effective auditing subsystem. In Proceedings of the 1987 IEEE Symposium on Security and Privacy, pages 13–22, Oakland, California, Apr. 27–29, 1987.
[46] P. A. Porras and R. A. Kemmerer. Penetration state tran-sition analysis: A rule-based intrusion detection approach. In Proceedings of the Eighth Annual Computer Security Ap-plications Conference, pages 220–229, San Antonio, Texas, Nov. 30–Dec. 4, 1992.
[47] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In Proceedings of the 20th National Information Systems Security Conference, pages 353–365, Baltimore, Maryland, Oct. 7–10, 1997. National Institute of Standards and Tech-nology/National Computer Security Center.
[48] P. Proctor. Audit reduction and misuse detection in heterogeneous environments: Framework and application. In Proceedings of the Tenth Annual Computer Security Applications Conference, pages 117–125, Orlando, Florida, Dec. 5–9, 1994.
[49] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., Calgary, Alberta, Canada,
Jan. 1998. http://www.clark.net/~roesch/idspaper.html.
[50] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst. Expert systems in intrusion detection: A case study. In Proceedings of the 11th National Computer Security Conference, pages 74–81, Baltimore, Maryland, Oct. 17–20, 1988. National Institute of Standards and Technology/National Computer Security Center.
[51] Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA. SunSHIELD Basic Security Module Guide, Solaris 7, Oct. 1998. Part No. 805-2635-10.
[52] U.S. Department of Defense. Trusted Computer System Evaluation Criteria, Dec. 1985. DoD 5200.28-STD.
[53] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In H. Debar, L. M´ e,and S. F. Wu, editors, Recent Advances in Intrusion Detection (RAID 2000), volume 1907 of LNCS, pages 80–92,Toulouse, France, Oct. 2–4, 2000. Springer-Verlag.
[54] U. T. Mattsson, and T. Valfridsson. An automated method to minimize the risk for exposure of encryption keys and encrypted database information. EPC Patent number – 00/975134.8.
[55] U. T. Mattsson. A method for implementation of encryption in a 24 by 7 production database. US Patent number 09/712 926.
[56] U. T. Mattsson. A method for detecting and preventing intrusions in commercial databases. EPC Patent number EP 01127906.4.
[57] U. T. Mattsson. A method for protecting databases against internal attacks. Sweden Patent number 0004189-7.
[58] U. T. Mattsson. Basic Data Type transparent method for storing and transporting of encrypted data. US Patent number 09/721 942.
[59] U. T. Mattsson. A method for combining software based encryption and hardware based encryption and key management. US Patent number 09/712 941.

Ulf T. Mattsson, Chief Technology Officer, Protegrity Inc., holds a master's degree in physics and a number of patents in the IT security area. His extensive IT and security industry experience includes 20 years with IBM as a manager of software development and a consulting resource to IBM's Research and Development organization, in the areas of IT Architecture and IT Security. Mattsson is an IBM Certified IT Architect and a research member of the International Federation for Information Processing (IFIP) WG

11.3 Data and Application Security, and a member of the IBM Privacy Management Advisory Council.