BEST PRACTICE FOR DATABASE ENCRYPTION SOLUTIONS

INTRODUCTION

Encryption can provide strong security for data at rest, but developing a database encryption strategy must take many factors into consideration. Organizations must balance between the requirement for security and the desire for excellent performance. Encryption at the database level, versus application level and file level has proved to be the ideal method to protect sensitive data and deliver performance. There are a multitude of architectures and techniques to improve performance: the alternatives fall into two broad categories – alternative topologies to decrease encryption overhead and techniques to limit the number of encryption operations. In addition, performance and security, in real-world scenarios, are complex issues and experts should be used who understand all available options and the impact for each particular customer environment. Every organization must protect sensitive data or suffer potential legislative, regulatory, legal and brand consequences. Relying on perimeter security and database access control does not provide adequate security. Packaged database encryption solutions have proven to be the best alternative to protect sensitive data. This is a specialized and complex solution area and if internal resources don't have the cryptography expertise in relation to IT environment, outside expertise should be used to ensure superior performance. This paper reviews the performance aspects of three dominant topologies for database encryption. This paper offers detailed guidance on scalable implementations of data at rest encryption in an enterprise environment, including encryption, key management, backup, auditing and logging should be deployed to optimize security, performance, scalability, and administration.

BEST PRACTICE

Building and maintaining a secure and efficient cryptographic engine is not the easiest task. The best practice, in nearly all cases, is to use an engine that's already available and tested. In a straight comparison of costs, Local Encryption Services are generally cheaper but not secure. Dedicated Encryption Services provides high availability with key caching and real cpu offloading. Benchmarks in customer environments demonstrated the criticality of making the right selection between the different topologies for database encryption implementations. A central topology benchmarked decryption of only a few hundred database rows per second and a more distributed hybrid topology benchmarked in the range of million rows per second, typically needed in environments with a high volume OLTP or parallel systems for decision support. Be aware that exposing encryption services as a network resource will introduce an additional point of attack, and very limited scalability in a database environment. Private keys should be stored encrypted with several AES encryption keys that are nested within a hierarchy in which each key is protected by a parent key. This multi-layer hierarchy of keys ensures the highest level of protection against attack. Engines come in three flavors: central, local and dedicated. Not protected properly, stored unprotected in a software environment, and unprotected in server memory, keys are vulnerable to discovery. What's needed? The best protection against private key compromise is a combination of physical security and key management technology, including stringent security standards throughout the private key lifecycle.

DATA AT REST ENCRYPTION

DIFFERENT APPROACHES HAS ITS ADVANTAGES AND DISADVANTAGES

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

ISSUES WITH APPLICATION LEVEL ENCRYPTION AND STORAGE LEVEL ENCRYPTION

Application-layer encryption requires rewrite of existing applications which is impractical due to limited IT resources, lack of access to source code, or a lack familiarity with old code. Rewriting applications is also very costly, risky and introduces an implementation time delay factor. Lastly, all applications that access the encrypted data must also be changed to support the encryption/decryption model. Storage-layer encryption alone can only protect against a narrow range of threats, namely media theft and storage system attacks.

DATABASE-LAYER ENCRYPTION

Database-layer encryption protects the data within the DBMS and also protects against a wide range of threats, including storage media theft, well known storage attacks, database-layer attacks, and malicious DBAs. Deployment at the column level within a database table, coupled with access controls will prevent theft of critical data.

DIFFERENT DIMENSIONS OF DATABASE ENCRYPTION SUPPORT

There are three main dimensions to encryption support in databases:

- One is the granularity of data to be encrypted or decrypted. The field, the row and the page, typically 4KB, are the alternatives. The field is the best choice, because it would minimize the number of bytes encrypted. However, as we have discovered, this will require methods of embedding encryption within relational databases or database servers.
- The second dimension is software versus hardware level implementation of encryption algorithms. Our results show that the choice makes significant impact on the performance. We have discovered encryption within relational databases based on hardware level implementation of encryption algorithms entail a significant start up cost for an encryption operation. Each model also offers different operational performance, tuning possibilities, and encryption offloading capabilities. The loss of granular protection will impact the security level. This is discussed in more detail in [18].
- The third dimension is the location of the encryption service - local service, remote procedure service, or network attached service. Choosing the point of implementation not only dictates the work that needs to be done from an integration perspective but also significantly affects the overall security model.

NOT ALWAYS PRACTICAL TO ENCRYPT DATA AS SOON AS IT ENTERS THE NETWORK

The sooner the encryption of data occurs, the more secure the environment—however, due to distributed business logic in application and database environments, it is not always practical to encrypt data as soon as it enters the network. Encryption performed by the DBMS can protect data at rest, but you must decide if you also require protection for data while it's moving between the applications and the database. We considered several possible combinations of different encryption approaches, namely; software and hardware level encryption, and different data granularity. We started with software encryption at field level. We then developed search acceleration support to index encrypted fields, and experienced a low performance overhead when searching on encrypted fields, including primary index fields. We finally directed our research and experiments to hardware level encryption use only for master key encryption.

ALGORITM PERFORMANCE AND SECURITY

Initially we considered several encryption algorithms AES, RSA [10] and b) Blowfish [11] for the implementation. We conducted experiments using these algorithms and found that the performance and security of the AES algorithm is better than the RSA implementation and the Blowfish algorithm implementation. AES is fast, compared to other well-known encryption algorithms such as DES [12]. Detailed description of the algorithm is given in [12]. DES is a 64-bit block cipher, which means that data is encrypted and decrypted in 64-bit chunks. This has implication on short data. Even 8-bit data, when encrypted by the algorithm will result in 64 bits.

OPTIMIZING DATABASE-LEVEL ENCRYPTION IMPLEMENTATIONS

Database-level encryption allows enterprises to secure data as it is written to and read from a database. This type of deployment is typically done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. Database-level encryption protects the data within the DBMS and also protects against a wide range of threats, including storage media theft, well known storage attacks, database-level attacks, and malicious DBAs. Database-level encryption eliminates all application changes required in the application-level model, and also addresses a growing trend towards embedding business logic within a DBMS through the use of stored procedures and triggers. Since the encryption/decryption only occurs within the database, this solution does not require an enterprise to understand or discover the access characteristics of applications to the data that is encrypted. While this solution can certainly secure data, it does require some integration work at the database level, including modifications of existing database schemas and the use of triggers and stored procedures to undertake encrypt and decrypt functions.

PERFORMANCE ASPECTS OF DATABASE-LEVEL ENCRYPTION

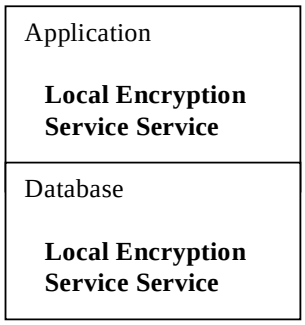
Additionally, careful consideration has to be given to the performance impact of implementing a database encryption solution, particularly if support for accelerated index-search on encrypted data is not used. First, enterprises must adopt an approach to encrypting only sensitive fields. Second, this level of encryption must consider leveraging hardware to increase the level of security and potentially to offload the cryptographic process in order to minimize any performance impact. The primary vulnerability of this type of encryption is that it does not protect against application-level attacks as the encryption function is strictly implemented within the DBMS. If we compare the response time for a query on unencrypted data with the response time for the same query over the same data, but with some or all of it encrypted, the response time over encrypted data will increase due to both the cost of decryption as well as routine and/or hardware invocations. This increase is referred to as the encryption penalty. An observation according to recent studies is that, different fields have different sensitivity [16]. It is possible for Hybrid to support encryption only on selected fields of selected tables. Encryption, by its nature, will slow down most SQL statements. If some care and discretion are used, the amount of extra overhead should be minimal. Also, encrypted data will have a significant impact on your database design. In general, you want to encrypt a few very sensitive data elements in a schema, like Social security numbers, credit card numbers, patient names, etc. Some data values are not very good candidates for encryption -- for example booleans (true and false), or other small sets like the integers 1 through 10. These values along with a column name may be easy to guess, so you want to decide whether encryption is really useful. Creating indexes on encrypted data is a good idea in some cases. Exact matches and joins of encrypted data will use the indexes you create. Since encrypted data is essentially binary data, range checking of encrypted data would require table scans. Range checking will require decrypting all the row values for a column, so it should be avoided if not tuned appropriately with an accelerated search index.

CENTRAL, LOCAL, OR DEDICATED ENCRYPTION SERVICES

The cryptographic engine lies at the core of the infrastructure. It implements algorithms, such as AES, DES, RSA, SHA, upon which the rest of the system depends. Any request to encrypt, decrypt, hash, or sign data ultimately passes through the engine. The management of encryption keys are the foundation of all encryption-based security solutions. Each of these approaches has its advantages and disadvantages. Database security is a wide research area [6, 3] and includes topics such as statistical database security [21], intrusion detection [19, 4], and most recently privacy preserving data mining [13], and related papers in designing information systems that protect the privacy and ownership of individual information while not impeding the flow of information, include [13, 14, 5, 8]. Users wishing to access data will now securely access it using the privacy management infra structure instead of developing multiple customized solutions for each application and data storage system. Applications and databases would not be impacted by an application specific implementation. This would alleviate the problem of maintaining the software and administrating privacy for each specific application and database.

LOCAL ENCRYPTION SERVICES

A Local Encryption Service is one in which the cryptography occurs on the same cpu as the rest of the application's processing. Typically for a Local Encryption Service, the algorithms are included as part of the application or as a library to which the application is linked. Examples of Local Encryption Services include RSA's Crypto-J, Cryptix, Mcrypt, and encryption libraries and toolkits included in products from some database vendors.

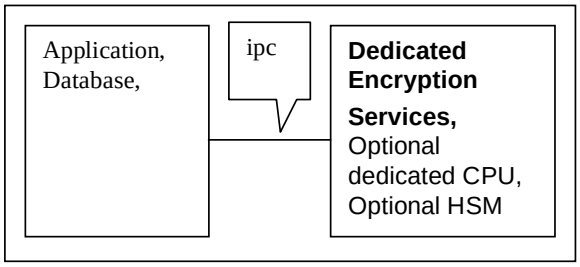


LOCAL ENCRYPTION SERVICES ARE GENERALLY CHEAPER BUT NOT SECURE

A Local Encryption Service is often easier to implement as there is no need to configure a separate hardware interface, and since there is no special hardware or software to purchase, the Local Encryption Service can be significantly less expensive, even free. Of course, the keys used by the engine are also significantly less secure. The application's performance will also suffer to a great extent by the inclusion of cryptographic processing on the application's cpu, and potential requirements in searching of encrypted database columns. Compared to a Dedicated Encryption Service, a Local Encryption Service offers a reduction in complexity in some areas. A Dedicated Encryption Service requires the installation and configuration of a separate layer of hardware. Local Encryption Services avoid this. However, Dedicated Encryption Services typically store the key away from the data and application and then wrap the key in layers of encryption and optionally use tamper-resistant hardware. Approximating this level of security using a Local Encryption Service requires a comprehensive and mature implementation. In a straight comparison of costs, Local Encryption Services are generally cheaper but not secure. The cost savings need to be balanced against security and performance issues.

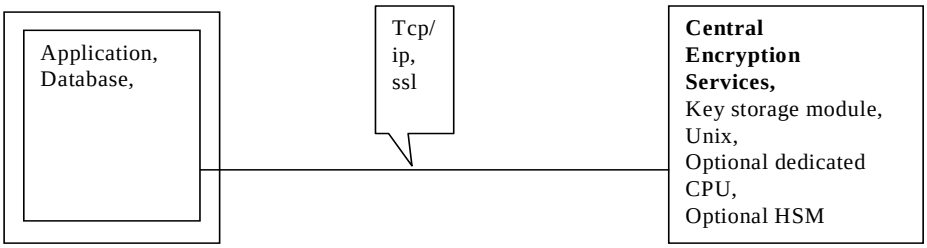
DEDICATED ENCRYPTION SERVICES

Dedicated Encryption Services contain separate processes dedicated just to cryptography. Dedicated Encryption Services could also contain a set of separeted cpu's dedicated just to cryptography. A typical example of a Dedicated Encryption Service is Protegrity Secure.Data Server in which the cryptographic processes are mounted. Another example of a Dedicated Encryption Service is a Hardware Security Module (hsm) in which the cryptographic cpu is mounted within a standalone, typically tamper-resistant, enclosure. An hsm might communicate with the application through a pci card, scsi, or over ipc (inter process communication). The Central Encryption Services may contain a number of separate cpu's dedicated just to cryptography. To maintain a high level of security the server platform should only contain securely encrypted lower level data encryption keys. Key encryption master keys should always be stored separately outside the server platform. Private keys should be stored encrypted with several AES encryption keys that are nested within a hierarchy in which each key is protected by a parent key. This multi-layer hierarchy of keys ensures the highest level of protection against attack.



CENTRAL ENCRYPTION SERVICES

Central Encryption Services can be implemented as a remote server with an optional HSM, or a network appliance (NAED). A typical example of a Central Encryption Service is a Server with a Hardware Security Module (hsm) in which the cryptographic cpu is mounted within a standalone, typically tamper-resistant, enclosure. An hsm might communicate with the application over the local area network using ssl. The goal of an hsm is to provide a secure environment for keys Thus most hsms combine the functionality of a key vault and engine. Relying only on remote central encryption support is not satisfactory, since it would always penalize network and total system performance, and more importantly, it is likely to open new security holes in the form of attacks on network exposed encryption services.



THE NETWORK ATTACHED ENCRYPTION DEVICE (NAED)

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

ENCRYPTION AS A NETWORK RESOURCE – A NEW POINT OF ATTACK

Be aware that exposing encryption services as a network resource will introduce an additional point of attack. An integrated central and distributed solution can protect from this vulnerability. Also, look for industry standard api support. Adopting a standard such as pkcs#11, will help ease the transition from one vendor's engine to another, and in some cases between different engines from the same vendor.

DENIAL OF SERVICE ATTACKS

A network attached engine, on the other hand, does not provide high availability, unless multiple engines are configured into a high availability cluster. Denial of service attacks are another related concern with network attached engines. Since the engine is available over tcp/ip, an attacker could flood the engine with traffic and block legitimate cryptographic requests. If required information can't be decrypted, then a customer may not be able to place an order or access account information. If the database stored encrypted records that are critical for the business operation, then a successful denial of service attack could be severe. None of the above are reasons not to use Dedicated Encryption Services, but rather factors to keep in mind when selecting a Dedicated Encryption Service.

INTERFACE TO ENCRYPTION HARDWARE ADD OVERHEAD

Central/Dedicated Encryption Services may be used in environments where performance and scalability are not critical requirements. A Dedicated Encryption Service is typically a specially constructed device which is connected via a cable to the computer needing cryptographic services, including pci or scsi, for directly connected engines, or ethernet for network connected encryption services. When selecting a Dedicated Encryption Service, consider performance, scalability, and availability. Most Dedicated Encryption Services will perform cryptographic operations faster than Local Encryption Services for larger blocks of data, however the interface to the hardware can add considerabel overhead for shorter blocks of data. This overhead may be noticeable on directly connected engines at higher transaction volumes. Network based engines, though, might carry a performance penalty from the need to negotiate a secure tcp connection. If the connection remains open between requests, then the overhead may be lower but it certainly should be tested.

THE MYTH THAT NAEDS OFF-LOAD WORK FROM THE DATABASE

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

THERE ARE THREE POINTS OF OVERHEAD WITH THIS TOPOLOGY

Let's explore a simple example to demonstrate the overhead; a user requests 500,000 rows of encrypted data:

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

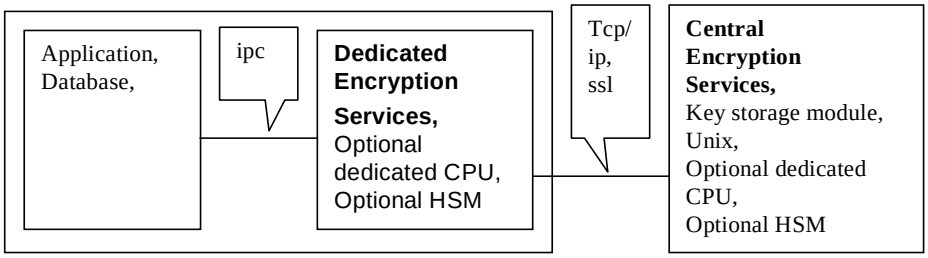
1. A NAED topology has three points of encryption versus one for other methods. In the example above, the 500,000 rows of data are sent over the wire to be decrypted at the NAED. The clear text is then encrypted using SSL to send back over the network and decrypted at the database to be served in clear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

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

A POWERFUL HYBRID SOLUTION

A powerful Hybrid solution combines the benefits of a Central Encryption Service with a Dedicated Encryption Service, on a general purpose computers running a standard operating system, but stripped of all but the most essential services. Amongst those services would be a cryptographic server and a key storage module.

To maintain a high level of security the server platform should only contain securely encrypted lower level data encryption keys. Key encryption master keys should always be stored separately outside the server platform on the central encryption services platform. Private keys should be stored encrypted with several AES encryption keys that are nested.



A MATURE DATABASE ENCRYPTION SOLUTION

The Hybrid solution Central Encryption Services provides secure and flexible key management and key backup. Dedicated Encryption Services provides high availability with key caching and real cpu offloading. Packaged and Integrated Local Encryption Services operations provides the highest operational performance and the highest availability for encryption services.

ALL DATA ITEMS ARE NOT EQUAL

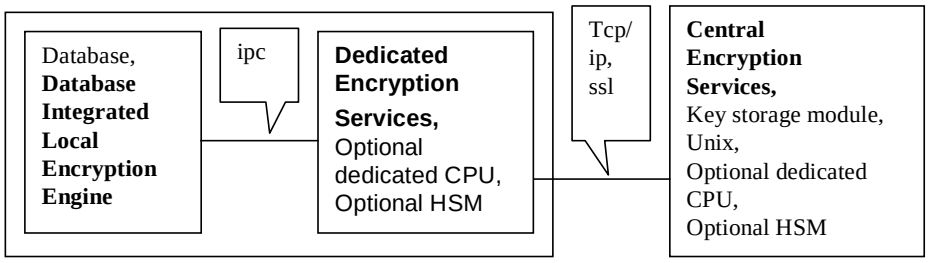
Some data requires a higher level of protection. Data classification can be used to determine if a specific data item should be processed locally, in a dedicated service, central service, or on a hsm. Risk management can help in defining the balance between these requirement for security, cost, and acceptable performance and scalability.

ALL KEYS ARE NOT EQUAL

Some encryption keys requires a higher level of protection. Master keys and some data encryption keys requires a higher level of protection. Data classification can be used to determine if a specific data item encryption key should be processed locally, in a dedicated service, central service, or on a hsm. Risk management can help in defining the balance between these requirements for security, cost, and acceptable performance and scalability.

A HIGH LEVEL OF SECURITY

To maintain a high level of security the server platform should only contain securely encrypted lower level data encryption keys. Key encryption master keys should always be stored separately outside the server platform on the central encryption services platform.



While most Dedicated Encryption Services are devices specifically constructed for cryptography, some Dedicated Encryption Services might be general purpose computers running standard oses, but stripped of all but the most essential services. Amongst those services would be a cryptographic server and a key storage module. At the heart of the server is a library such as the ones used for a Local Encryption Service. For that reason, these types of Dedicated Encryption Services. Private keys should be stored encrypted with several AES encryption keys that are nested within a hierarchy in which each key is protected by a parent key. This multi-layer hierarchy of keys ensures the highest level of protection against attack.

EFFECTIVE KEY PROTECTION IN MEMORY

Memory attacks may be theoretical, but cryptographic keys, unlike most other data in a computer memory, are random. Looking through memory structures for random data is very likely to reveal key material. Well made libraries for use as Local Encryption Services go to great efforts to protect keys even in memory. Key-encryption keys are used to encrypt the key while it is in memory and then the encrypted key is split into several parts and spread throughout the memory space. Decoy structures might be created that look like valid key material. Memory holding the key is quickly zeroed as soon as the cryptographic operation is finished. These techniques reduce the risk of memory attacks. Separate encryption can also be used for different data. These encryption keys can be automatically rotated based on the sensitivity of the protected data. Dedicated Encryption Services are also vulnerable to memory attacks. However, a well made Dedicated Encryption Service runs only the minimal number of services. Since web servers, application servers, and databases have no place on a dedicated cryptographic engine, these common attack points aren't a threat. This severely constrained attack surface makes it much more difficult to gain the access needed to launch a memory attack. To maintain a high level of security backups contain the encrypted data and only securely encrypted lower level keys. Master keys should be backed up separately.

SECURE KEY BACK UP

A weak link in the security of many networks is the backup process. Often, private keys and certificates are archived unprotected along with configuration data from the backend servers. The backup key file may be stored in clear text or protected only by an administrative password. This password is often chosen poorly and/or shared between operators. To take advantage of this weak protection mechanism, hackers can simply launch a dictionary attack (a series of educated guesses based on dictionary words) to obtain private keys and associated certificates.

HIGH PERFORMANCE AND SCALABILITY

This topology combines the enhanced performance of the Software structure with the added security of a hardware device. A HSM, in some situations, is an ideal way to add additional protection for the most important element of any encryption solution – the encryption keys. HSM devices are fast and tamper proof, so they make an excellent vault to store the crown jewels – the encryption keys. The performance in this topology is essentially identical to the earlier pure software structure, with an occasional transit to the HSM to refresh and retrieve the master encryption keys. During the majority of processing time, performance is identical to the software solution. In our 500,000-row example, in contrast to the NAED structure - where all 500,000 rows flowed over the wire to the NAED - the encryption service in the database server accesses the key from the HSM one time and thereafter all crypto operations are completed in the database by the software encryption service. The Hybrid system is implemented as distributed processes that scales with the number of processors and database server available. In the Software topology the database server becomes the platform for encryption services, removing the network and a remote device from the equation.

BENCHMARKS FROM REAL WORLD CUSTOMER ENVIRONMENTS

We studied the industry standard SQL benchmark [15] as a model for workloads. Some simple sample tests on Oracle and DB2. The first benchmark was focus on a particular customer scenario. Subsequent benchmarks used a workload combined from multiple customer case studies. The technological aspects of developing database privacy as an enterprise IT infrastructure component lead to new research challenges. First and foremost is the issue of encryption key management. Most corporations view their data as a very valuable asset. The key management system would need to provide sufficient security measures to guard the distributed use of encryption keys. We propose a combined hardware and software based data encryption system as the solution to this problem. A distributed policy and audit capability is proposed for the control the use of different encryption keys. Detailed investigation of this solution is presented below. Since the interaction between the database and the enterprise IT infrastructure component there are potential over-heads introduced by encryption. Therefore the sources of performance degradation and its significance should be determined.

DEDICATED ENCRYPTION SERVICES CAN OFF-LOAD THE ENCRYPTION

As mentioned before, cryptography consumes a fair amount of cpu cycles; with a Local Encryption Service application performance could be spread over a number of local processors. Dedicated Encryption Services off-load the encryption to a pool of separate processors. Some Dedicated Encryption Services also use special proprietary processors. Similar performance out of general purpose hardware could be easy and less expensive when using state of the art general servers. Obtaining a relevant degree of key security with a Local Encryption Service will require key storage separated from the local server. A central server with optional hardware to store the key can provide a cost effective solution in some environments. A final security consideration with Local Encryption Services is due to fact that the same physical memory is shared between the Local Encryption Service and the application.

SCALE BY ADDING ADDITIONAL SERVERS

Many modern application architectures scale by adding additional servers. In the case of directly connected engines, each new server requires a new engine. Directly connected engines in a highly available cluster of servers should provide cryptographic availability. Should one engine fail, then processing could shift to the other servers where the engines were still operational.

CUSTOMER REQUIREMENTS SCENARIOS

Target environment specified the following Requirements and Solution. Each Data Item was assigned a Data Security Class. The critical processing of Each Data Item determines the Performance requirement level. The requirements is mapped to the type of Encryption Service needed., and weather a HSM is required. In some cases Master Key processing requires a HSM.

HOW TO DELIVER ON SECURITY, PERFORMANCE, AND SCALABILITY

Local and Dedicated Encryption Services provides high availability with key caching and real cpu offloading. Packaged and Integrated Local Encryption Services operations provides the highest operational performance and the highest availability for encryption services.

ALL DATA ITEMS AND KEYS ARE NOT EQUAL

Some encryption keys requires a higher level of protection. Master keys and some data encryption keys requires a higher level of protection. Data classification can be used to determine if a specific data item encryption key should be processed locally, in a dedicated service, central service, or on a hsm. Risk management can help in defining the balance between these requirements for security, cost, and acceptable performance and scalability.

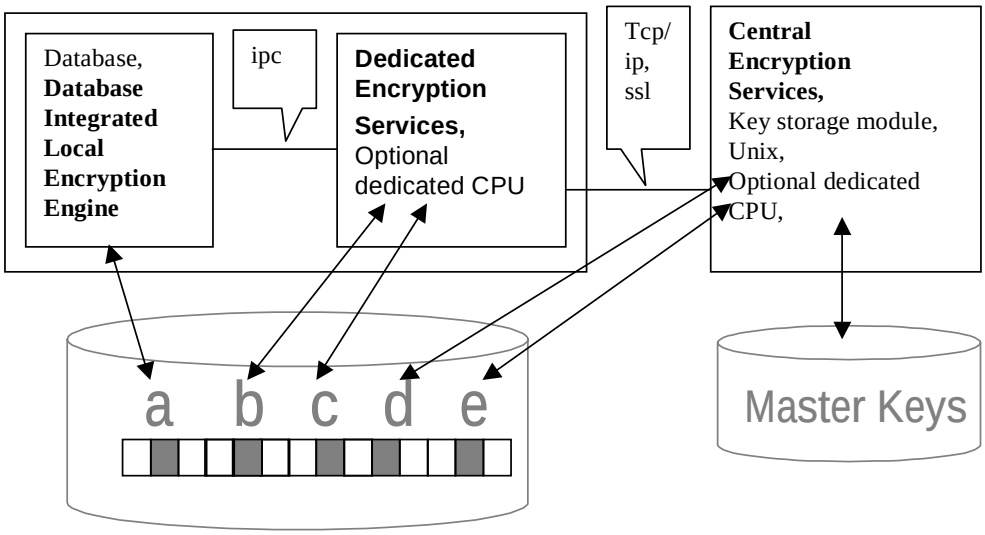
A TYPICAL RETAIL, BANK, AND INSURANCE CUSTOMER REQUIREMENTS SCENARIO

Target environment specified the following Requirements and Solution. Each Data Item was assigned a Data Security Class. The critical processing of Each Data Item determines the Performance requirement level. The requirements is mapped to the type of Encryption Service needed., and weather a HSM is required. In some cases Master Key processing requires a HSM.

Requirements and Solutions for this customer scenario:

Target	**Requirements**		**Solution**	
Data Item	**Data Security Class**	**Performance**	**Encryption Service**	**HSM**
a	Confidential	250,000 rows/second	Local Integated	N
b	Confidential	25,000 rows/second	Dedicated	N
c	Restricted Confidential	25,000 rows/second	Dedicated	N
d	Restricted Confidential	10,000 rows/second	Central	N
e	Restricted Confidential	25,000 rows/second	Central	N
Master Key	Master Key Class 2	1000 rows/second	Central	N

Solution and topology for this customer scenario:



TYPICAL GOVERNMENT CUSTOMER REQUIREMENTS SCENARIO

Target environment specified the following Requirements and Solution. Each Data Item was assigned a Data Security Class. The critical processing of Each Data Item determines the Performance requirement level. The requirements is mapped to the type of Encryption Service needed., and weather a HSM is required. In some cases Master Key processing requires a HSM.

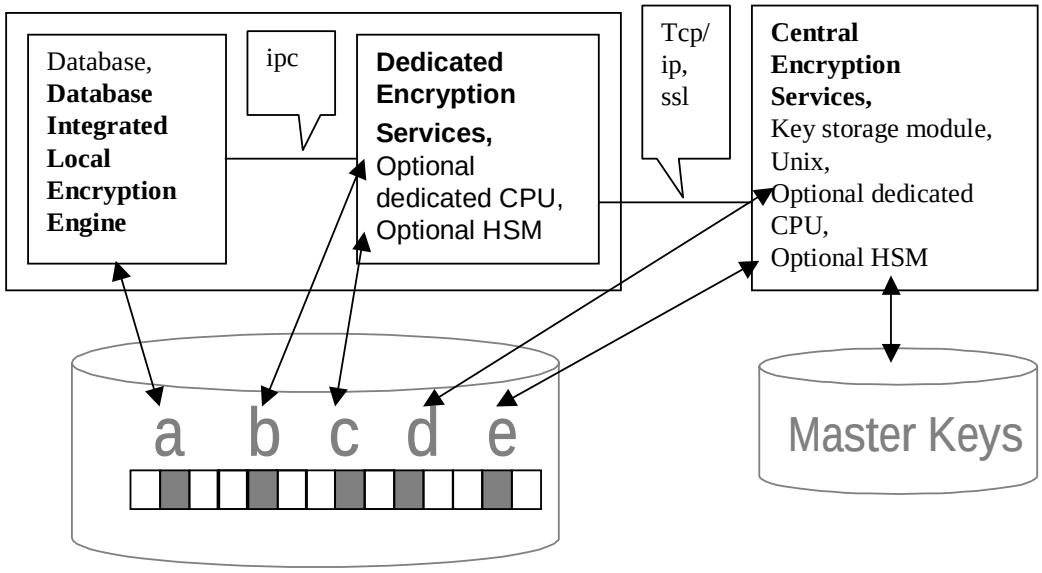
Requirements and Solutions for this customer scenario:

Target	**Requirements**		**Solution**	
Data Item	**Data Security Class**	**Performance**	**Encryption Service**	**HSM**
a	Confidential	250,000 rows/second	Local Integated	
b	Confidential	25,000 rows/second	Dedicated	
c	Restricted Confidential	1000 rows/second	Dedicated	Y
d	Restricted Confidential	10,000 rows/second	Central	
e	Restricted Confidential	1000 rows/second	Central	Y
Master Key	Master Key Class 1	1000 rows/second	Central	Y

THE HYBRID PROVIDES SECURE AND FLEXIBLE KEY MANAGEMENT

The Hybrid solution Central Encryption Services provides secure and flexible key management and key backup. Dedicated Encryption Services provides high availability with key caching and real cpu offloading.

Solution and topology for this customer scenario:



SELECTING A SOLUTION

Local and Dedicated Encryption Services (and hybrids) have their place in a cryptographic infrastructure, and often both are used. A Dedicated Encryption Service generally provides better security for the keys and off-loads the cryptographic processing so that the main application cpu isn't burdened with the math-heavy demands of cryptography. On the downside, Dedicated Encryption Services requires a tight integration with the operational database and os. If an hsm is used, a security officer frequently needs to be physically present at the device for some key management tasks. We will compare three interesting topologies for database encryption services:

1) Central Encryption Services – implemented as a remote server with optional HSM, or a network appliance (NAED).
2) Dedicated Encryption Services – implemented as a local process in software with optional HSM support.
3) A Hybrid Combination Services – implemented as a remote server with optional HSM, in combination with a local processes in software with optional local HSM support.

SECURE DATA

With Secure.Data, private keys remain private. In contrast to typical encryption configurations where private keys are stored insecurely, Secure.Data safeguards keys against compromise throughout the entire lifecycle of sensitive data, in both hardware and software, including removable media or optional hsm. With Secure.Data private keys are stored encrypted with several AES encryption keys that are nested

Secure.Data has designed a secure backup system where encryption keys are never exported from the product in clear text. The backup file is strongly encrypted using 3DES. Secure.Data makes it impossible for attackers to launch dictionary attacks and other password-guessing techniques aimed at exposing an administrative password and unlocking the backup file. Your private keys can never be exported in clear text and cannot be released without cracking several layers of triple-DES encryption, ensuring secure preservation of key data in all backup and storage activities.

HIGH VOLUME OLTP AND PARALLEL SYSTEMS SUPPORT

Secure.Data stands alone in this chaos as a product consciously designed from the base up to be an encryption system for high volume OLTP and parallel systems for decision support. This paper reveals some of the techniques architected into the Secure.Data product that have allowed parallelism in its fullest form to blossom. Being born for parallelism is the single most important fact in establishing Secure.Data dominance and success in the database encryption market today, and we'd like you to understand why. With a "Just say no!" attitude toward single-threaded operations, designers of Secure.Data parallelized everything, from the entry of SQL statements to the smallest detail of their execution. The Secure.Data entire foundation was constructed around the idea of giving each component in the system many sister-like counterparts. Not knowing where the future bottlenecks might spring up, developers weeded out all possible single points of control and effectively eliminated the conditions that breed gridlock in a system. Limitless interconnect pathways, optimizers, host channel connections, gateways, and units of parallelism can be defined in Secure.Data , increasing flexibility and control over performance that is crucial to large-scale decision support today. Secure.Data basic unit of parallelism is the Local and Dedicated Engine approach. From system configuration time forward all queries, data loads, backups, index builds, in fact everything that happens in a Secure.Data system, is shared across those pre-defined number of processes. The parallelism is total, predictable, and stable.

TAMPER RESISTANCE

Secure.Data is available with an optional, tamper-resistant hardware security module (HSM). Secure.Data HSM's are certified to FIPS 140-2 Level 3, the widely accepted standard of government-specified best practices for network security. Private keys are generated and stored in encrypted form within a tamper-resistant hardware security module (HSM). Keys stored in the HSM are protected from physical attacks and cannot be compromised even by stealing the hsm itself. Any attempt to tamper with or probe the card will result in the immediate destruction of all private key data, making it virtually impossible for either external or internal hackers to access this vital information. These products encrypt private keys using a group key that is divided between a small, predefined number of smart cards. Only Protegrity products that are members of the same security group (i.e. have access to the same group key) can share the sensitive key information contained within the backup files. Group key information is transferred between Secure.Data platforms using the appropriate number of smart cards. Secure.Data supports k of n secret sharing for the group key for increased security. The group key information is distributed across a collection of smart cards (n) where group membership

requires verification of a minimum number of smart cards (k). Using a k of n schema ensures a high level of data safety. Even if k-1 smart cards are stolen, the thief will not be able to access the configuration data stored on the tamper-resistant card because they do not have enough cards to meet the defined k of n criteria.

MANAGING KEYS FOR MAXIMUM SECURITY

With any product or system, key management security is only as good as the policies and practices that support it. Protegrity supports the following recommendations to ensure that your administrative practices are as secure as the key management technology we employ.

ADMINISTRATIVE ACCESS

The first step toward secure key management involves securing access to the Secure.Data product. This can be done is several ways:

- Protegrity products can be managed via a Central work station, and is not depending on an isolated or protected network. Protegrity is not depending the use of VLANs or similar technologies.

- Enforcing two-factor authentication for administration is accomplished through the use of RSA SecurID.

LOGGING, AND AUDITING CAPABILITIES

Extensive audit and activity logs are essential to all security systems. It is imperative to know who did what and when they did it for all actions in a secure system. Secure.Data provides both secure audit and activity logs that are transferable to a logging server. In addition, logs transferred off the Protegrity product are signed to expose later modification. The facility works by having the Secure.Data product sign log files with a log-signing certificate when they are copied onto the logging server or downloaded to a browser. All configuration changes on Secure.Data platforms are logged in the audit log. This ensures a documented record of all modifications to the secure networking product, including date and time, administrator username, and a description of the configuration change. The activity log records all connections to the Secure.Data platform. Each entry records the connection origin and the HTTP communications protocol request made. It also contains some security information such as the TLS cipher used to secure the connection.

CONCLUSION

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[19] Mattsson, Ulf T., Social Science Research Network, 'A Real-time Intrusion Prevention System for Commercial Enterprise Databases', http://papers.ssrn.com/sol3/papers.cfm?abstract_id=482282

[20] Mattsson, Ulf T., Search Security and Techtarget http://searchsecurity.techtarget.com/whitepaperPage/0,293857,sid14_gci1014677,00.html

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

[22] R. Agrawal and J. Kiernan. Watermarking relational databases. In 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.
[23] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In Proc. of the 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.
[24] Agrawal,J.Kiernan,R.Srikant,andY.Xu.Implementing P3P using database technology. In Proc. of the 19th Int'l Con-ference on Data Engineering, Bangalore, India, March 2003.
[25] R.Agrawal,J.Kiernan,R.Srikant,andY.Xu.AnXPath-based preference language for P3P. In Proc. of the 12th Int'l World Wide Web Conference, Budapest, Hungary, May 2003.
[26] D. E. Denning. Cryptography and Data Security. Addison-Wesley Publishing Company, Inc., 1982.
[27] T. Dierks and C. Allen. The TLS Protocol - Version 1.0, Internet-Draft. November 1997.
[28] A. Freier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0, Internet-Draft. November 1996.
[29] S. Garnkel and G. Spa ord. Web Security & Commerce. O'Reilly & Associates, Inc., 1997.
[30] S. B. Guthery and T. M. Jurgensen. Smart Card Developer's Kit. Macmillan Technical Publishing, 1998.
[31] Informix. Informix-Online Dynamic Server Administrator's Guide, Version 7.1. INFORMIX Software, Inc., 1994.
[32] G. Koch and K. Loney. Oracle8: The Complete Reference. Osborne/McGraw-Hill, 1997.
[33] J. C. Lagarias. Pseudo-random number generators in cryptography and number theory. InCryptology and Computational Number Theory, pages 115{143. American Mathematical Society, 1990.
[34] T. F. Lunt. A survey of intrusion detection techniques. Computer & Security, 12(4), 1993.
[35] National Bureau of Standards FIPS Publication 180. Secure Hash Standard, 1993.
[36] National Bureau of Standards FIPS Publication 46. Data Encryption Standard (DES), 1977.
[37] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. IEEE Communications, 32(9):33{38, 1994.
[38] San Jose Mercury News. Web site hacked; cards being canceled, Jan. 20, 2000.
[39] Oracle Technical White Paper. Database Security in Oracle8i, November 1999.
[40] W. Rankl and W. E_ng. Smart Card Handbook. John Wiley & Sons Ltd, 1997.
[41] R. Rivest. The MD5 Message-Digest Algorithm, RFC1321 (I). April 1992.
[42] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public key cryptosystems. Communications of the ACM, 21:120{126, February 1978.
[43] W. Rosenberry, D.Kenney, and G. Fisher. Understanding DCE. O'Reilly & Associates, Inc.,1992.
[44] A. Shamir. How to share a secret. Communication of the ACM, 22(11):612{613, 1979.
[45] D. R. Stinson. Cryptography; Theory and Practice. CRC Press, Inc., 1995.
[46] M. Lindemann and SW Smith, Improving DES Hardware Throughput for Short Operations, IBM Research Report, 2001, http://www.research.ibm.com/secure_systems_department/projects/scop/pap