Global Information Assurance Certification Paper

Database Encryption
Things to know before you encrypt

GIAC Security Essentials
Certification (GSEC)
Practical Assignment
Version 1.4b
Submitted: August 12, 2004
Submitted: August 21, 2004 v2

Option 1 - Research on Topics
in Information Security

Submitted by: James Summers
Location: SANS North Pacific 2004 - Portland, OR

Paper Abstract: Today we live in a society where information is power. Day by day through normal business activities our companies store pentabytes of information. Information that is valuable to our company, our company's clients, our competitors, terrorists, government agencies both foreign and domestic, organized crime and the enterprising young hacker. 20 years ago this was not a source for serious alarm, databases were stored on large mainframe systems with limited access, viruses were being developed in Petri dishes and the Internet was in its infancy. Today almost every company is connected to the internet, viruses infect computers and 13 year old kids know more about your security architecture than you do; don't you think it's time to take the next step? Database encryption, things to know before you encrypt.

Abstract

Today we live in a society where information is power. Day by day through normal business activities our companies store pentabytes of information. Information that is valuable to our company, our company's clients, our competitors, terrorists, government agencies both foreign and domestic, organized crime and the enterprising young hacker. 20 years ago this was not a source for serious alarm, databases were stored on large mainframe systems with limited access, viruses were being developed in Petri dishes and the Internet was in its infancy. Today almost every company is connected to the internet, viruses infect computers and 13 year old kids know more about your security architecture than you do; don't you think it's time to take the next step? Database encryption, things to know before you encrypt.

Why should I encrypt my database?

If you are seriously asking this question then you need to pay more attention to what's happening in the real world; i.e. put down that joystick and go browse Google, Yahoo or whatever your favorite news site is. Make sure to take a side trip down the technology links as well because every day a new hack or exploit comes out (http://isc.incidents.org/)(Ref. 17) or another company has lost sensitive information ("Citibank data on 123,690 clients vanished Feb. 21", Ref. 4) or had it stolen from them ("BJ's Wholesale suspect credit card leak", Ref. 20). Protecting the sensitive information of your company and your customers should be the most compelling reason for database encryption; it's the right thing to do. Unfortunately, this is not why companies are encrypting their sensitive information. There are two main reasons companies are forced into encrypting sensitive information. It's the "cost of doing business;" companies such as Visa and MasterCard require cardholder information to be encrypted, and financial impacts; financial loss due to information theft or regulatory fines imposed for not adhering to local, state and federal information security laws.

Visa's Cardholder Information Security Program (CISP) requires merchants and service providers to be compliant with their CISP requirements to do business with them. If these requirements are not met, your company could face fines of $50,000 for your first violation (not really a big deal), $100,000 for your second (starting to get your CEO's attention), up to getting kicked off the Visa network (not able to take Visa credit card payments anymore; dust off your resume, it's Monster time!). (Ref. 24) The CISP program has 39 pages of information security requirements, not all of which deal with encryption. However, for information specific to encryption of cardholder data, look to Requirement 3: Protect Stored Data in the "Visa U.S.A. Cardholder Information Security Program (CISP) Security Audit Procedures and Reporting as of 3/1/2004" document. (Ref. 25) MasterCard has a similar program called the MasterCard Site Data Protection Program. (Ref. 5)

States are passing laws such as California's SB 13866 that require any information loss or suspected information loss of unencrypted sensitive data by a public company to be reported to any effected California resident whether the company is California based or not. (Ref. 12 & 18) The law doesn't specifically require you to encrypt your sensitive information; however, it does state that if you do encrypt sensitive data then you do not have to give notice of a breach in security. The bill also gives provisions allowing a customer to take civil action to recover damages. Remember this only deals with unencrypted data. So, by not encrypting your data you open yourself up to lawsuits. Not a public company? In the article "IT Security Regulations Inevitable, Experts Say" (Ref. 3), the author Dennis Fisher writes about a new bill introduced by U.S. Rep. Adam Putnam that would require both public and private companies to give notice about breaches in security. Continuing on the federal theme, the federal government has passed laws like Sarbanes-Oxley, Gramm-Leach-Biley and HIPAA, none of which state anything about requiring data encryption but all of which have provisions requiring sensitive data to be protected. However, here is something to consider, especially with a law like HIPAA. In the medical community they have an understanding of standard of care. This is when a non-regulatory precedent has been set by the medical community about the manner in which you should care for a patient, a medical version of the information technology sector's best practices. In the medical world you can easily be sued for not following the standard of care for a particular treatment; how quickly do you think case law will be made for not doing due diligence by failing to follow industry best practices? If you have problems convincing your CEO that you should be encrypting sensitive data, just remind them that some of the above laws carry fines and jail time for the CEO. A good site for reviewing legal regulations is http://www.bakernet.com/ecommerce/. (Ref. 2)

Is database encryption the "silver bullet"?

No. It is very important to understand that encryption doesn't make all your security problems disappear. Just because you encrypt sensitive information doesn't mean that the information is unobtainable. To better understand this let's look at two scenarios.

On Friday, March 12, 2004 BJ's Wholesale Club Inc revealed a possible computer system break-in where a small fraction of its 8 million member's credit card information may have been stolen. (Ref. 21) Working with BJ's, MasterCard and Visa both warned consumers to check their credit card bills carefully. Further, M&T Bank informed their potentially effected customers by sending out letters to thousands of cardholders. So here are the questions that should be going through your mind: How much did it cost BJ's to investigate the break-in, resolve the issues around the break-in and notify its customers? How much did it

cost M&T Bank to notify their customers? What was the financial cost of the damage to BJ's reputation? Would this have happened if their database had been encrypted? It depends, but it should be noted that we don't know if BJ's had the sensitive information encrypted. The article doesn't say.

However, let's use this information to create a fictional scenario of a possible attack against BJ's. For this scenario let's assume that BJ's does encrypt their sensitive information. Further, we are going to assume that BJ's website is vulnerable against SQL injection attacks. Now let's say that we have an enterprising young hacker with too much time on his hands. Our young hacker has just finished reading, "SQL Injection: Modes of attack, defence (sic), and why it matters" by Stuart McDonald (Ref. 6) and wants to see if what he read is really possible. He goes to BJ's website looking for a form whose input would be used to query a database to retrieve information and sees that you can renew your membership online and that all you need is your membership number and a password. Now, our young hacker's mother is a member of BJ's, so he takes her card that has the membership account number on it to inspect the characteristics of the account number. He discovers that the Membership number is a 16-digit string. He uses this information to create a database of possible membership numbers (Don't want to get caught hacking with mom's account.). In the article on SQL insertion our attacker just read, he learned that many sites build SQL queries on the fly using information retrieved from user input forms. Noting from the article that one of the most basic mistakes is when a programmer writes SQL code on the fly and does not validate the input parameters. So, with form input parameters of <acctnum> and <password> lets assume the web page generates the following SQL query to retrieve the user profile information:

```
select * from userprofile where acctnum=<acctnum> and password='<password>';
```

Reading on our hacker learned that a query like this could be tricked into returning the user profile information without entering a valid password. This can be attempted by placing two dashes (--) after the membership account number in the acctnum input field and then any or no data in the password input field. This causes SQL to think of everything after the two dashes as a comment. The SQL query string would be the following if the entered <acctnum> was "1234567890123456--" and the <password> was "tom".

```
Select * from userprofile where acctnum=1234567890123456-- and password='tom';
```

Which to SQL would be the same query string as:

```
Select * from userprofile where acctnum=1234567890123456
```

This query string basically says retrieve all columns of information from the table userprofile where the acctnum is equal to 1234567890123456. So our hacker tries the input values outlined above and because he was lucky enough to pick a membership number that was active in the database, he is returned all of the

user profile information for account 1234567890123456. The application was even kind enough to return unencrypted values for the credit card number and expiration date to allow the profile user to verify and/or modify them. Now that our attacker has discovered this he can take his database of possible membership numbers and create a script to check them with the modified SQL insertion string against the web site and have his script automatically store the returned information in his database. Due to the ratio of possible account numbers to true account numbers, this type of attack should only retrieve a small fraction of the user profiles before a system administrator detects the increased load on the web and database servers (At least we hope, are you monitoring and alarming on such activity?). The important note here is that this attack clearly shows that even if your database stores all of its data encrypted, there are times when you need to decrypt the information. Attackers will attempt to exploit those times to retrieve the information they are looking for. As a side bar, this specific attack fails if you single quote the <acctnum> in the SQL query single as follows:

Select * from userprofile where acctnum=' 1234567890123456-- ' and password='tom';

This SQL query is functionally equivalent to the non-quoted query, but has the side benefit of breaking this SQL insertion attack. To learn more about this and other SQL insertion attacks, follow our enterprising young hacker's lead and go read Stuart McDonald's paper "SQL Injection: Modes of attack, defence (sic), and why it matters." (Ref. 6)

An article from the Japan Times (Ref. 4) reports in February of 2004, Citibank's Japan unit had a magnetic tape, containing 123,690 customer accounts with the customers' names, addresses, account numbers and account balances, vanish. The tape in question was being transported to a data processing center but never arrived. The article goes on to state that the tape was protected in such a way that it could only be read using a specific computer system. Here the article hints that the backup process used to create the tape did so in a manner that it encrypted all the data and that only a specific system could be used to retrieve the data. The article however doesn't state whether the tape information was a backup of a database or not. However, let's consider the case in which your database containing customers' names, addresses and encrypted credit card and social security numbers is backed up to tape where no other backup data encryption is used. What's the risk to your company if that tape is stolen? The thief gets a list of your customers which may or may not be useful to him. But he doesn't get what he really was after: your customers' credit card and social security numbers.

So what does database encryption buy me?

Database encryption buys you:

- An additional barrier in case other access controls are thwarted
- The ability to defend against ad-hoc queries for sensitive information specifically directed to your database (Note: the attacker might get information back, but the information should be encrypted)
- Security against hardware left where database information resides, like hard drives, backup tapes or an entire system
- The ability to have separation of duties between database administrators, that do not need access to credit card or social security information, and personnel who need access to the sensitive information to do their jobs but in limited amounts through an application's interface
- The ability to securely store backup copies or full database restores on a disaster recovery database in a "lights out" or hosted facility

Database encryption doesn't protect against:

- Not having an overall security policy
- Lack of strong information security policies and procedures
- Lack of security awareness throughout the corporation, which makes social engineering attacks much easier
- Poorly written applications
- Inadequate system-level security
- Inadequate network security which allows a remote attacker to take control of an inadequately secured system
- An inside attacker that has the rights to the unencrypted information through their normal job function
- Data being modified or deleted

Database encryption, like all other levels of security, is just one layer in your over-all security posture. There are many other areas of security to which you need to assert the same level of due diligence to create a solid security stance. Remember, the database is the prize for getting to the center of your security "Tootsie Pop." You need to make sure that your outer security layers (company security awareness, network security, systems security, etc...) are just as strong as a hard candy shell. The goal is to force an attacker to have to take a long time to get to the soft chew center -- your data. The more time it takes the better the chances are you will detect and stop him. The following are papers that touch on those other layers of security. You can find more online at the SANS Reading Room, http://www.sans.org/rr/ (Ref. 18).

Moving from Consciousness to Culture: Creating an Environment of Security Awareness by Mary Mulney (Ref. 7)
Making Your Network Safe for Databases by Duane Winner (Ref. 26)
Database - The Final Firewall by Brian Suddeth (Ref. 20)

Ok, I understand that database encryption is not the solution to life, the universe and everything. (Ref. 1) So what do I do next?

Well, the easy part's done. You have decided to encrypt your database. Now comes the more difficult part of deciding and then documenting what to encrypt and what the encryption environment should be. Let's go over the following topics to help you decide what encryption architecture you want to implement. This will prepare you for writing your database encryption plans.

- What information should I encrypt?
- Should I purchase software, hardware or build it in-house?
- What encryption algorithm(s) should I use?
- What about key management?

What information should I encrypt?

The answer to this question depends on the information that your company stores. I would look to laws like Sarbanes-Oxley, Gramm-Leach-Biley, HIPAA and California's SB 13866 or partner regulations like the Visa CISP or Master Card's Site Data Protection Program when deciding what to encrypt.

HIPAA defines sensitive information as any information that identifies an individual, or could be reasonably believed to identify an individual. (Ref. 21) (Well that's clear.) Requirement 3: Protect Stored Data in the "Visa U.S.A. Cardholder Information Security Program (CISP) Security Audit Procedures and Reporting as of 3/1/2004" document, Visa states, "The MINIMUM account information that needs to be encrypted is the Visa account number." (Ref. 25) (Personally, I'd go a little father than this.) In California's SB 13866 section 2e (Ref. 19), California defines personal information to be a person's full name combined with any of the following: social security number, driver license number, state identification card number and/or credit or debit card number in combination with any security code/password. All right, California! I understand that you might not agree with the overall law, but at least they have set forth a pretty clear definition of personal information.

Here is a list of items worthy of encryption:

- Any payment device information
  - Bank Account Number
  - Check Numbers and the MICR
  - Credit Card Number and expiration date
  - Debit Card Number
- Any identification cards

    - Drivers License Number
    - Social Security Number
    - State Identification Card Number
- Company HR Related Items
    - Employee Salaries
    - Stock option amounts
    - Bonuses

Remember, you also might want to encrypt other data points as well. It just depends on what your company considers to be sensitive information. Once you have determined your definition of what sensitive information is, create a security policy clearly stating what the company considers to be sensitive information, how to handle sensitive information and the penalties for not adhering to the policy. Make sure to have your CEO sign off on the policy and get your HR staff to start training the employees.

Should I purchase software, hardware or build it in-house?

Instead of telling you what you should do, I am going to use this section to compare some of the options available to you. I will describe the options and give the pros and cons associated with each. This way you can make the decision of what best fits your environment.

Encrypted Files Systems (EFS)
EFS are when you encrypt a file(s), directory or the entire drive(s) that house your database. This is generally carried out by the operating system or by a third party software package. An example of this would be the Windows encrypted file system native to Windows 2000 Server.
Pros

- Relatively easy to implement
- No code changes to your applications
- Encrypts the data and the database schema

Cons

- Noticeable system and database performance hit
- Does not protect against system administrators who have the ability to modify file access rights (no separation of duties)
- Does not protect against an attacker that gains root access to your system
- Very slow for medium to large databases
- Might not support encrypted backup
- No real concept of key management
- Weak encryption algorithm depending on the base OS
- Operating system service packs or security updates could break EFS

James Summers		

Personal View

Don't consider this option unless you are working with a very small in-house or personal database with no remote user requirements. However, in a situation where you do not have software engineers to modify in-house or third party application code or the financial resources to purchase a better solution, this option provides an improvement over no encryption.

Whole Database Encryption Software

This is when you use a third party software solution to encrypt the entire database file(s). Examples would be the Secure.Data.Suite from Protegrity (Ref. 13) or the NetLib Encryptionizer for SQL Server by NetLib (Ref. 11).

Pros

- After EFS, full database encryption is the easiest to implement
- No code changes to your applications
- Multiple encryption algorithms to choose from: DES, 3DES, AES
- Allows for separation of duties
- Most support encrypted backup
- Can protect against hardware theft if used in combination with external security device
- Encrypts the data and the database schema

Cons

- Noticeable system and database performance hit for medium to large online transactional databases
- Not easily scalable
- Compatibility issues with OS security updates, service packs and patches
- Need multiple server licenses to have redundant, reporting or backup servers with encrypted copies of the production database
- Encryption software, encryption keys and the encrypted data are typically stored on the same system

Personal View

Whole database encryption can be a good option for databases that do not have many users needing quick response times to complete their tasks. Small internal use only databases fit into this category. However, an online order transaction processing database serving thousands of customers a day does not. This is the preferred option over EFS in a situation where you do not have software engineers to modify in-house or third party application code. Just remember that you will need a software license per database server supporting encrypted databases.

Column Encryption Software on the Database Server

This is when you use a third party software solution to encrypt select columns of the database. Examples again would be the Secure.Data.Suite from Protegrity (Ref. 13) or the NetLib Encryptionizer for SQL Server by NetLib (Ref. 11).

Pros

- Quicker than full database encryption
- Application can control who , what where and when data is viewed
- Multiple encryption algorithms to choose from: DES, 3DES, AES
- Allows for separation of duties
- Supports encrypted backup
- Can protect against hardware theft if used in combination with external security device

Cons

- Slight system performance hit
- Requires code changes to database stored procedures and/or in-line SQL code for in-house or third party applications
- Not easily scalable
- Compatibility issues with OS security updates, service packs and patches
- Need multiple server licenses to have redundant, reporting or backup servers with encrypted copies of the production database
- Encryption software, encryption keys and the encrypted data are typically stored on the same system

Personal View

This is a solution I just can not recommend. If you are going to take the time to modify your in-house or third party code, then uses a solution that provides scalability for growth and the added advantage of not having the encryption software, the encryption keys and the encrypted data all on the same box. But as the caveat, if you really can't afford the cost of going to a hardware based solution and have a database that needs quicker access such that full database encryption is not an option then this might be the solution for you.

Hardware Solutions

For all the solutions outlined above, the database server is responsible for encrypting and decrypting the data thus the reason for the system performance hit. In a hardware solution scenario, you attach a specialized hardware device to the system through a PCI, SCSI, USB or network interface. The system then sends all encryption/decryption requests along with the data to encrypt/decrypt and some form of encryption key or key identifier to the encryption hardware device. The device does the required cryptographic function and returns the encrypted or decrypted data back to the server. Examples of such devices are nCipher's netHSM and nShield (Ref. 10) devices and SafeNet's Luna SA (Ref. 15) device.

Pros

- Fast
- Scalable
- Tamper-resistant
- Application can control who, what, where and when data is viewed
- Multiple encryption algorithms to choose from: DES, 3DES, AES
- The devices singled out above are FIPS 140-2 Level 3 (Ref. 9) validated

- Allows for separation of duties
- Supports encrypted backup
- Protects against hardware theft
- Encryption keys are not stored on the same system as the encrypted data

Cons

- Requires significant modification to in-house or third party application code
- Devices tend to be pricy
- It is very easy to have implementation flaws in code

Personal View

This is the solution I recommend. Again, if you are going to take the time to modify your in-house or third party code, then use a solution that provides scalability for growth, the added advantage of not having the encryption software, the encryption keys and the encrypted data all on the same box and has been validated by independent labs using the National Institute of Standards and Technology (NIST) testing requirements (Ref. 9) to be Federal Information Processing Standard (FIPS) 140-2 level 3 compliant. Because the hardware devices tend to be pricy, you could develop an intermediate service, a "Crypto-Service," to call the encryption hardware. Have your application servers send all encryption / decryption requests to the "Crypto-Service." If you only deploy the "Crypto-Service" on a small number of servers it would give you the benefit of having many application servers call a fewer number of "Crypto-Service" servers; thus, fewer connection licenses and/or hardware encryption devices needed. Now you have the ability to use as many application servers as you want limited only by the ability of each "Crypto-Service" server and hardware encryption device to keep up with the encryption/decryption requests. Two other benefits of this model are a unified programmer's application interface that all applications use to call the "Crypto-Service" making application development easier and the ability to load balance crypto requests over multiple "Crypto-Service" servers providing greater redundancy and scalability.

Please note that I purposely left out discussing any options where you write the entire encryption solution in-house. I did this because I want to stress the point that this should only be done by security companies looking to sell there solutions out on the open market. These companies are paying for the best and the brightest in the cryptographic field to come up with these solutions. Further, they pay thousands of dollars to have independent labs certify their solutions. If you are smart enough to make the decision to encrypt your sensitive data and reap the benefits of having the data encrypted; be smart enough to also take the added benefit of making someone else assume the risk of having to justify how they implemented the standard cryptographic algorithms in their solution.

What encryption algorithm(s) should I use?

First let's talk about the algorithm to use for encrypting/decrypting your data and then we will consider the use of other cryptographic function to assist in data retrieval.

There are many encryption algorithms out there, although NIST has defined validation tests for only the following: standard DES, 3DES, AES and Skipjack. (Ref. 8) So which one do we pick? Let's start by eliminating DES and Skipjack. We do this because it is considered relatively trivial to break standard DES encryption and therefore, NIST no longer recommends its use. (Ref. 14) The Skipjack encryption algorithm is used in escrowed encryption applications like the U.S Government's Clipper Chip, not for database encryption. So now we must choose either 3DES with a 168-bit key or AES with key lengths of 128, 192 and 256-bit. Either of these algorithms is good choices, but if you pick AES, implement with a 256-bit key. Why you ask? Because you can!

Now let's consider the use of another cryptographic function, a one-way hash. A one-way hash is an algorithm that takes variable length input and returns a fixed length output. The hash algorithm is considered one-way because it should be computationally infeasible to decrypt a hashed string. Hash functions also have the following two characteristics:

- The same input value will always return the same hashed value
- No two input values will have the same hashed value

The SHA-1 hash is such a hash. It will take a variable length input string and return a 160-bit output string. At this time it should be noted that the length of the encrypted output for AES and 3DES is dependent on the length of the input. One should also note that each time you encrypt the same input value of Y you will receive a different encrypted output value.

Well thanks for sharing. So, why are you telling me this?

I bring this up because I want to point out something you should consider when designing your database encryption plan. If you do lookups by any of your encrypted columns like, "Do I have credit card 4444333322221111 in my blacklisted credit card table?" How are you going to search for that information? You will not be able to encrypt the credit card with 3DES or AES and search by the encrypted value because each time you encrypt the input value of 4444333322221111 you will receive a different encrypted output value. Thus you will never receive a match on your query even if you have 4444333322221111 in your blacklisted credit card table. If try to search by decrypting every row in your credit card blacklist table and then doing your comparison, you are going to bring the database to a halt, especially if you have to do this for every transaction you attempt to process. So what do you do?

James Summers		

Try this as a possible solution. Add a new 160-bit column to your database to represent the field you are trying to query. In this field place the SHA-1 hashed value of the following combined string:

<secret seed value><credit card number>

Later I will explain the reason for the secret seed. For now just know that the secret seed value can be a string of any length containing any combination of letters, numbers and/or symbols you wish. The only requirement for the secret seed value is that a very limited number of people (like the CFO and CSO) know what the value is. With that in mind, remember that one of the characteristics of a hashed output value is that it will always be unique; no two input values will return the same hashed output value. And to recap the second characteristic, you will always get the same hashed output value when you input the same input value. So, if you have credit card number 4444333322221111 and throw in the seed value of "Hi Mom," you would have

"Hi Mom4444433333222211111" as your input value

<seed><credit card number>

Each and every time you SHA-1 hash this value you will get the following as your hashed output value:

26DBE7B89328FE4DD7B94E988A9F191E5707C930

So, now you have a unique representation of your original credit card number that can be indexed and searched on without having to decrypt any information.

Ok, now let's revisit the concept of adding a secret seed value to the hash input value. The reason you do this is to further protect the credit card number and the encryption keys. Let's assume that you did not add the secret seed to the input value and only hashed the credit card number. Then in your database you would have the hashed credit card value stored along with the 3DES encrypted credit card value. For the sake of the scenario let's say all this information is in a table called customer-profile. Let's also assume someone steals your database and is able to read all of the data. Right away the hacker goes to your customer-profile table and by viewing the data sees that you are encrypting information. He notes this because not too many companies are using strings like

26DBE7B89328FE4DD7B94E988A9F191E5707C930

as a street address. Further evaluation reveals that there is a column where the data for each row is unique and 160-bits length. Our hacker guesses that this column contains SHA-1 hashed data, due to the 160-bit length, and because he knows that only valuable information is encrypted he assumes the column represents the credit card number. To validate this guess, he calls up and orders a product with a fake or stolen credit card to get known information into your database. He then steals the database again, looks up the data in the column he believes to be the SHA-1 hashed credit card for his recently created customer profile and compares the value to the SHA-1 hash he generated from his fake or stolen credit card. The one he gave to the customer support person when ordering. The values match. Now he knows that this column contains the SHA-1 hash of the customer's credit card. So he jumps out to his favorite hacker site and finds a credit card number generator program and generates a list of 10

million random credit card numbers. He takes these numbers and individually SHA-1 hashes each and places both the generated and hashed data in a database of his own. With his newly created database he queries his copy of your database to see if any SHA-1 values match. If one does, then he takes all the rest of the information associated with that matching record and places it in his database. When done our hacker has a database with all the information in your customer-profile table along with the unencrypted customer's credit card number for all records he was able to match the hashed values. Further, because you have the 3DES encrypted credit card number in the same table; he decides to try a brute force known plaintext attack against your 3DES encrypted credit card information to see if he can discover the encryption key. If he can get the encryption key, game over; he wins. Now he has the means to decrypt the rest of the encrypted data. This is why you might want to use a secret seed along with the credit card number to create your data for the hashed column. You might also want to consider not placing information like the user name, the SHA-1 hashed value and the 3DES encrypted value in the same table or database.

Key Management? What about Key Management!

The encryption key is the plain text string used by 3DES and AES to control the encryption and decryption of data. Since knowing the key gives you the ability to decrypt any data encrypted with that key, it has to be protected at all cost. Because of the need for such strict security when dealing with the key, you need to understand and create a policy to deal with:

- Key Control Measures
- Key Generation
- Key Storage
- Key Retirement and Deletion

The best way to understand these points is to see how they are defined in an actual policy. In Appendix A, I attached a cleaned version of such a policy. When I created this policy, I had no one single source to look to. So, I scoured though many books, whitepapers, websites and regulatory documents and came up with what you see in Appendix A. However, a greater portion of the document is from the SANS sample security policy template, "Acceptable Encryption Policy" (Ref. 16). Please feel free to use any or all of it as a template to assist you in the creation of your policy. I hope it helps. However, there are two items I specifically would like to draw you attention to: One, don't use the same encryption key for every task. If for any reason a specific key is compromised the person with the compromised key will only be able to decrypt a subset of your total encrypted data. And two, validate your ability to rotate your keys on a regular bases in your development process, not after you go live. The rotation of keys is a necessary step in encryption. Encryption algorithms such as 3DES and AES are based on

the fact that it is computationally infeasible to decrypt the encrypted information, not impossible. With enough time and raw computational power it can happen. It did for standard DES. So a way to limit your exposure is to rotate the keys on a regular basis. Further, if your key does ever become compromised you will need to change it quickly. This is not the time for discovering a coding error isn't allowing you to change your encryption key.

Conclusion

Whether you like it or not, sooner rather than later a business partner or government law is going to require you to encrypt your sensitive information. And they should. You know it's the right thing to do. You might even already have an idea of how to sell it to your CEO. Just remember to take the time now, while you have time, to carefully plan out your encryption architecture before laws are passed requiring sensitive information encryption. Don't forget to document it by creating, maintaining and following company policies and procedures. This will give you the ability to create a stronger security posture while still providing the company the information it needs to continue doing business post encryption.

References

1. Adams, Douglas, THE ULTIMATE HITCHHIKER'S GUIDE. New York, NY: Portland House, 1997.

2. Baker & McKenzie, "E-COMMERCE LAW RESOURCES", 2004 URL:http://www.bakernet.com/ecommerce/

3. Fisher, Dennis, "IT Security Regulations Inevitable, Experts Say", May 12,2004, URL:http://www.eweek.com/article2/0,1759,1591366,00.asp

4. Japan Times, The "Citibank data on 123,690 clients vanished Feb. 21", March 20,2004, URL:http://202.221.217.59/print/business/nb03-2004/nb20040320a1.htm

5. MasterCard, "Introducting the MasterCard Site Data Protection Program" URL:https://sdp.mastercardintl.com/

6. McDonald, Stuart, SQL Injection: Modes of attack, defence (sic), and why it matters", April 8, 2002 URL: http://www.sans.org/rr/papers/index.php?id=23

7. Munley, Mary, "Moving from Consciousness to Culture: Creating an Environment of Security Awareness", July 25, 2004, URL: http://www.sans.org/rr/papers/index.php?id=1439

8. National Institute of Standards and Technology, "Advanced Encryption Standard (AES), Data Encryption Standard (DES), Triple-DES, and Skipjack Algorithms", April 17,2002 URL: http://csrc.nist.gov/cryptval/des.htm

9. National Institute of Standards and Technology, "FIPS PUB 140-2, Security Requirements for Cryptographic Modules", May 13, 2004 URL: http://csrc.ncsl.nist.gov/cryptval/

10. nCipher Corporation , "Hardware Security Modules (HSMs)", 2004 URL: http://www.ncipher.com/hsms/

11. NetLib, "Data Security for SQL Server 2000 and SQL Server 7", 2004 URL:http://www.netlib.com/sql-server-encryption.shtml

12. PGP Corporation, "California Senate Bill 1386, Data Security and Encryption", 2004 URL:http://www.pgp.com/products/whitepapers/Regulation_CSB_040419_FL.pdf

13. Protegrity,Inc., Protegrity Home Page, 2004
URL: http://www.protegrity.se/

14. Roberts, Paul, “NIST says DES encryption 'inadequate'”, July 29, 2004
URL: http://www.infoworld.com/article/04/07/29/HNdesinadequate_1.html

15. SafeNet, Inc., “Luna SA – Network-Attached Hardware Security Appliance”, 2004
URL: http://www.safenet-inc.com/products/luna/luna_sa.asp

16. SANS, "Acceptable Encryption Policy"
URL: http://www.sans.org/resources/policies/Acceptable_Encryption_Policy.doc

17. SANS, "Internet Storm Center"
URL:http://isc.incidents.org/

18. SANS, "SANS InfoSec Reading Room"
URL:http://www.sans.org/rr/

19. Senator Peace & Assembly Member Simitian, "California SB 1386", March, 2002
URL:http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html

20. Suddeth, Brian, "Database - The Final Firewall", January 28, 2002,
URL:http://www.sans.org/rr/papers/index.php?id=11

21. Sullivan, Bob, "BJ's Whoesale suspect credit card leak", March 12,2004,
URL:http://www.msnbc.msn.com/id/4516301/

22. US Department of Health, "HIPAA Protecting Health Information"
URL: http://dchealth.dc.gov/hipaa/hipaaphi.shtm

23. US Department of Health, "HIPAA OVERVIEW"
URL:http://dchealth.dc.gov/hipaa/hipaaoverview.shtm

24. VISA, "Cardholder Information Security Program",
URL:http://usa.visa.com/business/merchants/cisp_index.html

25. VISA, "Visa U.S.A. Cardholder Information Security Program (CISP) Security Audit Procedures and Reporting as of 3/1/2003",
URL:http://usa.visa.com/media/business/cisp/Security_Audit_Procedures_and_Reporting.pdf

26. Winner, Duane, "Making Your Network Safe for Databases", July 21, 2002,
URL:http://www.sans.org/rr/papers/index.php?id=24APPENDIX A

Appendix A

1.0 Purpose

The purpose of this policy is to provide a set of encryption standards. By adhering to these standards <Company Name> will be able to better assure the security of sensitive customer and internal information. Additionally, this policy provides direction to ensure that <Whatever Company or legal> regulations are followed.

2.0 Background

Proven, standard algorithms such as 3DES, IDEA, Blowfish, RSA, and RC5 should be used as the basis for encryption technologies. These algorithms represent the actual cipher used for an approved application. For example, PGP Corporation's Pretty Good Privacy (PGP) uses a combination of IDEA and RSA or Diffie-Hillman, while Secure Socket Layer (SSL) uses RSA encryption. Be aware that the export of encryption technologies is restricted by the U.S. Government. Residents of countries other than the United States should make themselves aware of the encryption technology laws of the country in which they reside.

2.0 Scope

This policy applies to all <Company Name> employees and affiliates.

3.0 Policy

<Company Name> is committed to protecting sensitive information, both customer and internal intellectual property. <Company Name> will encrypt sensitive information while in transit to <Company Name> and while in storage at <Company Name>. Symmetric cryptosystem key lengths must be at least 128 bits. Further, no shared symmetric keys are to be used for more then one-to-one communication between two entities. Asymmetric crypto-system keys must be of a length that yields equivalent strength. <Company Name>'s key length requirements will be reviewed annually and upgraded as technology allows. The use of proprietary encryption algorithms is not allowed for any purpose, unless reviewed by qualified experts outside of the vendor in question and approved by I.T.

3.1 Sensitive Data

<Company Name> defines sensitive information as <create your list>. All sensitive information must be encrypted while in transit and storage.

3.1.1 Sensitive Data in transit via https

All sensitive information must be encrypted via 128 bit RSA SSL encryption while the data is in transit from the Internet to <Company Name>'s Web DMZ. The DMZ servers must decrypt the SSL data, do field and format validation on the data and then re-encrypted the data via 128 bit RSA SSL encryption to transmit the data from <Company Name>'s Web DMZ to <Company Name>'s internal partner application servers. No

sensitive credit card information may be stored on <Company Name>'s Web DMZ or internal application servers.

3.1.2 Sensitive Data in transit via e-mail

Any e-mail containing sensitive information must be sent encrypted.

3.2 Stored Sensitive Data

All sensitive information must be triple-DES or AES-256 encrypted and logically isolated by Partner when stored. A one way irreversible hash of sensitive information with a secret seed may be generated for indexing functionality.

4.0 Authorized Encryption Hardware & Software

4.1 Authorized Secure Socket Layer (SSL) Encryption

Symmetric cryptosystem key lengths must be at least 128 bits.

4.2 Authorized Encryption Software

<Company Name>'s standard is to meet or exceed industry best practices; therefore, <Company Name> requires that all software encryption used by <Company Name> must at minimum be certified to the National Institute of Standards and Technology (NIST)'s Federal Information Processing Standards (FIPS) 140-1 Level 2 regulations. <Company Name> employees and affiliates are authorized to use PGP Corporation's Pretty Good Privacy (PGP) v8.x and above, or the open source GPG software, for encrypting sensitive data. All sensitive data must be encrypted with the standard triple-DES or AES algorithms. Any e-mail containing sensitive information must be sent using the above PGP or GPG software.

4.3 Authorized Hardware Encryption Devices

<Company Name>'s standard is to meet or exceed industry best practices; therefore, <Company Name> requires that all hardware encryption devices used by <Company Name> must at minimum be certified to the National Institute of Standards and Technology (NIST)'s Federal Information Processing Standards (FIPS) 140-2 Level 3 regulations. For this reason <Company Name> has standardized on nChiper's netHSM and nShield product lines. All hardware encryption devices must be placed in secured restricted areas where access to the area is limited to authorized personnel only, controlled by electronic badge and is monitored by video cameras 24x7x365. These products must be configured to run in FIPS 140-2 Level 3 mode during installation.

5.0 Hardware Encryption Device – Key Management

Equally as important as setting a standard to meet or exceed industry best practices on purchasing and use of hardware encryption devices is management of the keys that control the devices. Therefore, the following regulations must be strictly adhered to to protect encrypted keys against disclosure and misuse.

A. All symmetric cryptosystem key lengths must be at least 128 bits for strong key generation.
B. Access to cryptographic keys will be on an as needed basis.

a. The key-encrypting key will be generated with 6 administrative secure-id cards each with a unique password, any two of which will need to be present to modify the key in any manner. The password must follow the same password characteristics as set forth in <Company Name>'s Electronic Account Policy. Four of the card holders will be <Company Name>'s CEO, CFO, VP of Operations and VP of Payment & Risk. The remaining 2 cards, cards A & B, will be stored in separate fire safes. Secure-id card A and the password for secure-id card B will be stored in <Company Name>'s finance fire safe. Secure-id card B and the password for secure-id card A will be stored in <Company Name>'s IT fire safe.
b. The data-encrypting keys will be generated with Y operator secure-id cards where X is 2 plus the number of production or development hardware encryption devices used for a particular function. One operator card will need to be present in the hardware encryption device for the key to be used; therefore X of the operator secure-id cards will remain in the hardware secure-id card readers while the remaining 2 cards will be stored in fire safes; one in <Company Name>'s financial fire stare and one in the <Company Name>'s IT fire safe. If for any reason <Company Name> believes any data key has been compromised an authorized staff member, a person with physical access to <Company Name>'s datacenter, can remove the operator secure-id cards and cause all encryption processing to stop.

C. To securely distribute a key, the key must first be stored in a triple-DES encrypted format that can not be unencrypted anywhere but on the hardware encryption device. Only then can the key be distributed to other systems which will then use the hardware encryption device for encrypting and decrypting data.
D. Any key stored off the hardware encryption device must be stored in a triple-DES encrypted format. A key can not be stored in the clear anywhere but on the hardware encryption device; further the triple-DES encrypted key must not be able to be unencrypted anywhere but on the hardware encryption device. <Company Name> also requires that the key-encrypting keys and the data-encrypting keys be stored in different locations.
E. Each <Company Name> Partner requires their own data-encrypting key. No data-encrypting keys can be shared between Partners.
F. Data-encrypting keys must be changed out at least once a year. The old key will be stored for 1 year after it has been retired and then destroyed.

6.0 Legal & Regulatory Adherence

Be aware that the export of encryption technologies is restricted by the U.S. Government. Residents of countries other than the United States should make themselves aware of the encryption technology laws of the country in which they reside.

7.0 Enforcement

Any employee found to have violated this policy may be subject to disciplinary action, up to and including termination of employment.

8.0 Definitions

Term	Definition
Asymmetric Cryptosystem	A method of encryption in which two different keys are used: one for encrypting and one for decrypting the data (e.g., public-key encryption).
Proprietary Encryption	An algorithm that has not been made public and/or has not withstood public scrutiny. The developer of the algorithm could be a vendor, an individual, or the government.
Sensitive Payment Information	<Company Name> defines sensitive payment information as the credit card number, credit card expiration date, the credit card bin range, the check MICR number and the check routing number.
Symmetric Cryptosystem	A method of encryption in which the same key is used for both encryption and decryption of the data.
<Company Name> Partner	Any company paying <Company Name> to provide services.