An Electronic Citadel: A Method for Securing Credit Card and Private Consumer Data in E-business Sites and Database Systems

Tom Arnold, Chief Software Architect and VP of Product Strategy for the Merchant Services Division of InfoSpace, Inc.

Executive Summary

This paper presents an Electronic Citadel, a method and system by which e-Businesses can secure business critical data with reasonable assurance that the data will remain secure for an extended period of time. For e-Businesses, this private data most certainly includes customer information and credit card account numbers, but can include any other sensitive attributes like blanket PO numbers, names, or purchase detail.

Any discussion of security technology must be preceded by a review of threats and challenges. To accomplish this, the paper will focus on the recent attacks on e-Businesses where credit card numbers were the targets. These were highly visible and well publicized security breeches. The number and types of data attributes stolen are not really material to the discussion of electronic security principals, except to say that the impact to on-going business was extremely adverse. Although merchants carried the brunt of the damage as a result of the loss, many others were impacted. Confidence in using known brands was damaged; consumers felt threatened if their card accounts were violated; hundreds of banks had to close and reissue accounts; and the e-commerce industry was again branded a security risk by concerned users.

When discussing the complexities and intricacies of security systems, especially cryptographic systems, it helps to focus on a single, well-understood attribute like a credit card number. Especially an attribute that is universally accepted as a mechanism for concluding commerce transactions; where most people understand and respect the vulnerability that results from a data compromise.

The analogy of citadels constructed in the 18th and 19th centuries is an interesting parallel to e-Business system security in the 21st century. In the 17 and 1800s, the threat was from large professional armies in mass, open field formations firing smoothbore muskets. Defenses were based on a design of a series of geometrical shapes and angles allowing a multiple fields of fire to cover approaches. Further,

military fortification designers in the early-1800s knew that effective defense from intruders was to layer barriers to weaken and stop attackers, while at the heart of the citadel lay innermost stone fortress, the final major obstacle an intruder must defeat to win. The threat to e-Business systems is from hackers exploiting weaknesses in defenses, internal employees exploiting their trusted status, and brute-force attacks on passwords.

Unfortunately, many of today's e-Businesses implement the direct opposite of a citadel. I call it an "egg shell" security model: hard outer shell, soft in the center. Businesses following the egg model fortify the outer shell using filtering routers and firewalls. Defense against internal attack is defended by simple username and password logins. And, with the expansion of 802.11x wireless network technologies, individual empowered employees are installing wireless access points inside company LANs. Some companies implement more secure password protection mechanisms and compartmentalize sensitive data. Others segment networks deploying firewalls between segments. Segmentation, compartmentalization and strong passwords all are good, but once someone penetrates the outer shell, that person is functioning on the soft center of the organization and may only have to guess at a directory name to gain access to the most sensitive data.

The continued exploitation of the egg-shell model has caused the National Infrastructure Protection Center to issue [NIPC 01-003] an alert to e-Businesses about the growing threat to their operations.

In response, this paper introduces a method and system for an e-citadel security model. This model is about creating a final barrier that will protect sensitive data when network and system security barriers have been passed (by either authorized or unauthorized access).

Throughout this paper, I describe the citadel security model using end-user credit card account numbers as the critical item that MUST be protected, while addressing functions that will permit e-Business systems to continue to search and verify the credit card account number value from prior use. I will refer to the credit card account number as an *index* item, in that the value of the card number has significance for future reference. This paper proposes mandatory security methods; process and software services that need to be present to provide a minimal level of security while supporting continued e-Business operations.

New in this version

An alternative mechanism for creating the ccIndex[n] value has been included in this version of the Electronic Citadel.

The updated method allows the inclusion of additional attributes related to an original sales transaction with a credit card account number to create the index item. The example described in section 3.3.2 illustrates appending the date and time of original transaction with the credit card number to harden the computed [SHA] message digest. The sales order date and time was selected among numerous candidate elements based on the reliability of the values to the merchant.

This modification will assist in defeating a brute-force vulnerability to determine the [SHA] value for a given credit card number in two potential situation. The first vulnerability assumes the attacker knows the type of credit card (eg., first digit of card number is based on card type) and final four digits from the ccMask[n] value. The second vulnerability arises in the event that the merchant also stores the bank identification number (BIN) with the card mask. In this latter case, an attacker may know up to the first six digits of the account number along with the relative positions of the numbers. Here are two examples of what a potential attacker might know:

ccMask + card type = 4xxxxxxxxxxx4019
ccMask + type + BIN = 105418xxxxxx4019

Given these cases, the attacker need only to roll through permutations involving as few as 6 unknown characters and can test each permutation using modulo/10 prior to using SHA to compare digests.

Although the likelihood of this type of attack taking place seems remote, some merchants may want to derive the [SHA] digest value by combining variables not associated with a given credit card account that are easily known to the merchant prior to computing the digest. This is discussed in depth in section 3.3.2.

Status of this Document

This document describes a method and system for protecting critical consumer information. To the best of my knowledge at the time of writing, no patents, trademarks or service marks are violated. Others wrote the cryptographic methods described in this paper and appropriate credit has been given. Patents and copyrights protect some of the protocols, algorithms, or methods discussed in this paper. The reader is advised to obtain all necessary patent and copyright licenses before implementing.

Additionally, some companies may design, develop and launch products that are similar and address some, all or more requirements than are described in these pages. The author offers this method and system as a proposal and places the intellectual property contained herein into the **public domain** for the good of the Internet and the Internet community of businesses.

Comments and contributions are solicited for potential future versions of this paper. All comments should be addressed through to the author at the e-mail address listed in *About the Author*.

Important Author's note: My intent is to describe a subsystem that can be implemented within any overall e-Business systems environment that will resolve the growing problem related to the loss of consumer data. Not too long ago, my personal credit card data was obtained by criminal hackers (crackers) from an e-commerce site's database. As both a consumer and insider in the payments industry, I know both the personal pain and the business challenge that this security breach caused. Given that my account number was not the only one acquired, I know that the cost to the bank that issued that account was substantial when they had to notify me, close the account, and reissue a new credit card. And please consider that these were direct costs, not accounting for the loss of goodwill incurred by the merchant. This situation is bad for consumers; bad for banks; bad for merchants; bad for everyone.

Scope of this Paper

Although the concepts presented can be applied to many business secure data storage problems, the focus for this paper is on businesses that collect and store critical information on consumers and other business operation, whether this information is stored in an electronic form in either on-line or off-line databases. The specific scope is on businesses that use the public Internet to either capture or retrieve data.

This paper is NOT a dissertation of cryptography and cryptographic methods. Instead it is a concept involving the usage of cryptographic methods to implement the security model.

For the purpose of describing the Electronic Citadel, a credit-card account number will be used to illustrate the method.

Version and Copyrights

The first version of this paper was published by the author and copy-written as a white paper from the Technology Working Group, E-Business Division, Software and

Information Industry Association April 2001. This revision is made and published with permission from the Software and Information Industry Association. I want to thank the gang at the SIIA for their continued support and confidence.

Reader

Although this paper covers a technical topic that is frequently relegated to information security experts, the information is intended for business operations management as well.

Primary users include:

- Information security specialists
- Risk managers
- Systems analysts
- Information system architects
- Systems designers

Secondary readers include:

- Product managers
- IT program managers
- Chief information officers
- Chief technology officers
- Operations analysts
- Systems administrators

Introduction

This paper is not for the comfortable CIO who knows for sure that all of their critical business data are secure. It's not for the CEO who sleeps soundly at night knowing that his company's most critical data resources are under round-the-clock surveillance. Furthermore, this paper is not for the consumer who believes sending their personal information (like credit card account number) sent in e-mail is a smart thing, or that the unknown voice on the phone is a person who can be trusted.

No, this paper is not for them. It's for the rest of us in the industry who cringe each time we read a news story about hackers stealing credit card numbers from an e-Business's database, sensitive business secrets being stolen and used against a company, or military technology secrets being stolen.

In response to threats, security experts have long asserted the need to compartmentalize access to sensitive locations and data contained therein. The concept

of granting access on a *need-to*-know basis has long been a standard in the security industry. To some extent, this same technique has been used with large databases in that a user login has been coupled with a role (commonly referred to as *permissions*), and then data access restricted or granted appropriately. Unfortunately, each access barrier and each security system is subject to some level of compromise.

With the expansion of electronic commerce on the Internet, billions of dollars worth of merchandise and services are purchased each year in the United States alone. It's widely accepted that the business-to-consumer (B2C) industry is the fastest growing retail merchandising sector. Each day, millions of people type their credit card numbers and other personal data into Web-based order forms to initiate a quick purchase transaction. With any growth market, the opportunity for criminal abuse grows as well. The Electronic Citadel described in this paper borrows the analogy of permanent fortification design to suggest a security model that is scalable, flexible, achieves business objectives, provides protection of critical assets, and can be affordably implemented by e-businesses.

This section introduces this document, requirements and scope of the Electronic Citadel. The chapters following provide detailed information about security threats, software systems and architecture of the Electronic Citadel.

Terminology

As with any technical topic, there are sometimes unique terms related to the discipline or related to the subject matter. This section describes a set of terms and how each is used in this document. Some of which have common meaning, others unique to data security or this paper's context. Unfortunately, this list is probably not exhaustive. Feedback and questions about terminology used in this document is always welcome.

Business Terms

Consumer	As used, this is the person who interacts with an e-Business site to consummate a commerce transaction. This person may act as an individual or as a purchasing agent for another business entity.
e-Business	This is the business entity that implements a Web site for the purpose of presenting products and conducting commercial transactions.
(e-Business) site	This includes all databases, application servers, web servers and IT systems that support Web sites used to accept commercial transactions from consumers.
End-user	Same as consumer.
Internet	This refers to the publicly switched network including routers,

	switches, hubs and other network devices where digital traffic are sustained, hosted and accessed.
Vulnerability	An weakness that can be exploited to cause damage to an e-Business or data stored at an e-Business site.
Threat	The probability that a vulnerability will be exploited.

Technology Terms

Brute-force	A method of attacking where all possible keys or secret codes are attempted to break in. Commonly used in cryptanalysis.
Cipher text	This is the encrypted clear text.
Clear text	The machine or human readable document. Sometimes referred to as the original document.
Compartmentalization	A method of partitioning either data or physical environments so that only authorized users can access them.
Cryptanalysis	The discipline of code breaking; the methods used to try decrypting a cipher text when no key exists.
Cryptography	The discipline of creating algorithms, systems and methods to encrypt and decrypt data so that only authorized users or systems can see the clear text. See [Applied] for more technical details on cryptography, including protocols, algorithms and source code. For specific cryptographic algorithms, see [DES], [3DES], [RC4].
Decryption	The method of using a key or secret fact to unscramble a cipher text and retrieve the clear text.
Encrypted text or encrypted document	Same as cipher text.
Encryption	The methods of using a key or secret fact to scramble the characters in a clear text document and create a cipher text.
Fingerprint or electronic fingerprint	As used in this document, same as a message digest.
Key	A string of characters or symbols used to drive an encryption/decryption algorithm. The key, plus the encryption algorithm, plus the clear text document, results in the cipher text. The key, plus the decryption algorithm, plus the cipher text, results in the clear text document.
Key cipher	This is the encrypted version of a key that is stored on a system.
Key generation	An algorithm that accepts seed values and generates a key.
Key management	The process where keys are securely stored and retrieved as needed to encrypt and decrypt documents

Message digest	A algorithm used to create a unique fingerprint for any data or document. The original document is left untouched by the digesting algorithm. Should any change occur after the digest is created, a new digest will result in a different value. See [MD4], [MD5] and [SHA].
Obscurity	The unaccepted method of securing a document or data by hiding.
One-way encryption	Use of a key and an encryption algorithm to scramble the characters of a clear text document and make the document permanently unreadable. This would be the same as encrypting a document and throwing away the key so that the key cannot be recovered, and thus, the clear text cannot be recovered.
One-way hash	Same as message digest.
Plain text	Same as clear text.
Wipe	A cryptographic algorithm involving the writing bits in all clusters/storage blocks that contained a file on magnetic media. This involves the destruction of stored data such that it cannot be recovered. See [Applied – p. 228-229] for summary and [WIPE] for details.

Electronic Citadel Terms

Index item	This is a single data element that will need to be located from a database.
Mask	This is a partially displayed clear text value of the target data.
Pass-phrase	This is a string that is used by an authorized process to access a key.
Recovery period	This is the time period that target data will need to be recovered by some authorized process.
Search item	Same as index item.
Target data	This is the data element that needs to be protected.
Encrypt/Decrypt algorithm	Electronic Citadel suggests using [3DES] stream cipher for encryption of target data objects. [3DES] will also be used during the recovery period to decrypt. More will be said on this in following chapters.
Index item algorithm	Electronic Citadel suggests using [SHA] message digest algorithms to create the index item attribute. More will be said in following chapters on this subject.

Threats and challenges

Important Note: The remainder of this paper will use a consumer credit card account number as the target data object to be secured by the Electronic Citadel. Threats and business challenges will focus on securing this data element. This is a partial list of the data security threats and management challenges that confront e-Businesses.

Threats do not encompass all risks and have been generalized for the purpose of this discussion. A detailed analysis of data security threats can be found in many books and Web sites. E-Business operators or IT managers should know and subscribe to the services of [CERT] and [NIPC] for the latest alerts and update on security threats.

Data security threat

The following is a partial list of general threats and methods used against sensitive e-Business data:

Application attack	Force application to fail-over and acquire 'root' access to system. At a minimum, deny service after causing failure.
Backup tapes	Copies of sensitive data backed up on magnetic tape. Are copies encrypted? How are they stored? Who has access?
Brute-force attack	This is a cracking method used to break into a system.
Buffer overflow	Flood data into a form field (works either internal or external) applications causing an overflow condition and the failure of the application and possibly system. This may seem malicious, but many applications don't fail well and can cause servers to reboot or may grant super-user access to the perpetrator when this is done. This is a common method used to force the installation of a Trojan horse on a infected system.
Cracking and entering	Breaking into system to steal things. Fine line between this and the trespasser.
Dictionary attack	Use of a word-list to facilitate a brute-force attack. These are commonly used during a password fishing attack or when searching of any secure phrase assigned by end-users.
Dumpster diving	Method of filtering through discarded documents to locate important information. For instance searching through waste documents to find discarded sales receipts with credit card information.
Employee theft	Employee with access copies, misuse, steals and sells or exceeds access privileges to data. Not to be underestimated, this is the largest

	threat.
Infection or infected system	This refers to a system that has been compromised by any malicious code (virus or Trojan horse) that was not specifically installed by an authorized system administrator.
Monkey-in-the-middle attack	Sometime referred to as a man-in-the-middle attack. This involves unauthorized sniffing or snooping on a network to monitor and analyze traffic. Common mechanism to acquire customer data, passwords or employee data.
Password fishing	Trying to log into a system by attempting common passwords. This would include attempting to use known default passwords as set in vendor products. Usually the same as a brute-force attack.
Reverse social engineering	Similar to the former, except this may takes the form where the unsuspecting victim calls the hacker into assist with a system failure. For instance, clerk's computer is disabled. When discovered, the clerk notices a new sign saying to call a number for IT help. Clerk calls hacker for help with down machine.
Snooping	Use of a software program to intercept data and send it elsewhere. Monitoring data communication traffic from a host:port is a form of snooping.
Social engineering	This is the largest problem and one that crackers pride themselves at being adept at. It takes many forms, but may involve smooth talking your way into someone letting you have access. Call up and act like a network engineer and ask for a password.
Some new attack	Something else thought up by a hacker or cracker that no security administrator, IT person, security expert, or me thought of yet.
Tailgating	Logon behind someone else, or use a desktop that was left logged-on.
Temporary files	Work files used while processing transactions. Many applications that communicate with banking systems leave copies of work files lying around on a system
Trespass	Classic hacking. Break-in and have a look around. Frequently, Trojan horses or "back-door" programs are left lying around so the hacker can return to the server using some other method after the initial penetration.
Trojan horse	Can be used for snooping. Typically a program that diverts information or has some additional functions. Frequently used in a virus attack.
Viruses	These are fairly well known, but they can also be used to implant Trojan horses.

This is just a quick dump of the types of threats confronting IT professionals and e-Businesses. If you don't believe these are real threats or would like to know more, check [CYBNOTE] for more information.

Many of these threats directly target databases of credit card numbers stored on e-Business sites. Using security systems to create barriers and address each of these threats is surely a defense. The problem clearly arises when the defense has been breached and the cracker is inside looking around. The response has been the development and fielding of numerous security monitoring applications that watch the defensive barriers and try to identify when a cracker is in the process of attacking. These systems are used to predict the chance of penetration and allow the IT system administrators time to react. You can think of this as an alarm that calls your troops to man the outer fortifications. All of these systems are fine and well and need to be in place. The problem truly arises when the security monitoring infrastructure is weak or fails to detect the penetration. What next?

Business challenges

E-Business investment in security infrastructure has been based on reaction to actual problems and based on vendor trust. In essence, "I bought ABC's software to run credit card processing. I didn't know that it left temporary files with credit card numbers lying around on a disk directory." Unfortunately, ABC Company won't be the one out of business after news hits that your credit card database is in Russia.

The one clear constraint is that the all-knowing, all-seeing security system does not exist and cannot be built. Even the Electronic Citadel proposed in this paper has weaknesses and could be compromised. Given this, e-business's approach to information security has to be finely balanced. The technology and monitoring systems cannot be so over-bearing as to eliminate the business model. On the other hand, not having a security system (or, leaving security up to a vendor's product) is just plane stupid business.

Lastly, remember that trying to hide things, using the obscurity method, is the next closest thing to doing nothing. After all, unless you're willing just to loose a data element forever (e.g., destroy the thing using [WIPE]), somebody knows where it is and how to retrieve it. Keep in mind that when data are deleted from a computer, the file or elements may not actually be deleted. Delete programs do not actually destroy data. Usually the storage addresses of the data being deleted are moved to a free space list and made available to be overwritten. Needless to say, the deleted data can be recovered. For this reason, just deleting something is akin to using the obscurity method of security.

General Requirements

This section describes a set of general requirements for an e-Business to receive credit card account data from consumers and process commerce transactions. The requirements are stated in the context of the Electronic Citadel and refer to best security practices for the e-Business.

Important note: The terms MUST, MUST NOT, SHOULD, SHOULD NOT, MAY and MAY NOT are used in accordance with [RFC 2119].

Overall Objectives

This section summarizes general business requirements

Overall credit card security system MUST not get in the way of a consumer initiating a transaction and a merchant completing transaction processing.

Primary objective MUST be to secure credit card data and increase consumer confidence.

General Requirements for Building Mask and Index Values

This section describes the general requirements for the formation of the ccIndex[n] and ccMask[n] values.

A ccMask[n] value MUST be created and stored to assist in post-sale identification of a specific credit card used to provide payment on an order. The ccMask[n] field has been padded with 'x' (ASCII "\120") characters denoting the length of the credit card account number. This SHOULD be added to allow users to read the string and identify the relative location of the mask characters. In a database, the ccMask[n] MAY store the padded string or only the readable characters. In the following example, the ccMask0 value stored in the database is the value 4019. The 'x' characters are added to make the value human recognizable. At the time the ccMask[n] value is displayed the merchant MAY prefix the presented value by including the first character as a numeric representation of the type of credit card.

As an example:

Masked card value:	4xxxxxxxxxxx4019
Or, without card type	xxxxxxxxxxxx4019

The clear text credit card value MUST be converted to a ccIndex[n] value using [SHA] in accordance with the Electronic Citadel model. A ccMask[n] value MUST also be created. These are the only two values that MUST be stored as a reference to the original credit card number used in the transaction. The ccIndex[n] digest value which is a byte array SHOULD be represented as hex characters or as base 64 encoded characters prior to being stored in a database. The following is an example of the value:

Masked card value:	4xxxxxxxxxxx4019
Base 64 encoded digest:	da6vV0EJRNhUdo+zXxM/A7fReXw=
Hex characters:	75aeaf57410944d854768fb35f133f03b7d1797c

The masked card value represents the display of a ccMask[n] value for the original card. This value is described in the prior section and is included for illustration. The base 64 and hex values are character representations of the [SHA] digest byte arrays. They are equivalent. Either MAY be used depending on merchant storage capabilities and preference. The base 64 representation is smaller and can be easier to store and retrieve.

For additional security (especially in the case where credit card numbers are being protected), the ccIndex[n] value MAY be created by [SHA] digesting the order time and date along with the credit card number. As an example:

1.	Card value with date and time:	4xxxxxxxxxxx40190503200215:45:13
	Base 64 encoded digest of card and time:	e4RNpZlqy2JKenEbc306i6mryY4=
2.	Card value with date and time:	4xxxxxxxxxxx40190503200215:46:02
	Base 64 encoded digest same card, different time:	dnjolvHOU3bjmrUuFohQPFurBkM=

These two examples show what happens to the resulting base 64 encoded digest values when the same card number is used and the order time is different. Assuming the time (to seconds) and date of the order are proprietary to the merchant and that the merchant only wants to verify that a credit card presented after the sale is the same used for the original order, this method adds two additional variables under

the merchants control to help obscure the credit card value. If an attacker does not know the exact date and time of the order, it would be magnitudes more difficult to brute force derive the credit card number from this digest value.

Selling

E-businesses implement two types of selling modes. One accepts credit card and consumer data through a Web-based interface (like a shopping cart or order form). The other is a convenience method that permits a consumer to make an instant purchase without having to fill out forms again. I've used the term *one-click* buy. This section describes requirements related to the selling process.

Clear text credit card data SHOULD be available to submit to a bank so as to obtain a pre-sale authorization. Shy of any ability to hand encrypted credit card data to a bank credit card processor, the merchant will have to submit name, amount, card number and other data according to the card processor's specification.
There is no method to check the authenticity of a credit card account or to obtain billing address verification service without requesting an authorization. Businesses will continue to request $1.00 authorizations on cards to obtain this information prior to initiating an actual transaction.

Standard Shopping Cart / Order Form

The vast majority of e-Business commerce transactions originate through the use of a shopping card or standard order form. In this scenario, the consumer finishes shopping and decides to *checkout* and pay for their purchases. Usually, a secure server, using [SSL] to protect the transmission of data between customer's browser and merchant, delivers an order form for display on the customer's browser. Using this form, the customer will type in billing information, name, credit card account number, and expiration data. When this form is submitted back to the Web site, the merchant's server opens a connection to a payment processor, submits the appropriate card information with the intent of obtaining an *authorization.* Should the credit card processor not object to the proposed transaction, the merchant's server will receive an authorization string as a response. From this point forward, the order acceptance process will continue.
Credit card account data, name, billing address, shipping address, or other personal data MUST be collected through a secure form. Secure form uses [SSL] with minimum key length of 128 bits.

These forms MUST be secured through an authenticated interface in that the customer's browser verifies the server that it is connected with. Further discussion on this topic is beyond the scope of this paper.

A cipher text of the clear text credit card number MUST be produced using [3DES]. As an example, clear text credit card number encrypted generates ccCipher[n]. The ccCipher[n] value MUST be the only value stored from which the clear text card number can be retrieved at a future time. (Note: the management of keys will be discussed in later sections).

The key used to create the [3DES] cipher text MUST be securely stored such that the key cannot be derived or stolen. Further, this key MUST be recoverable for the recovery period such that cipher text can be decrypted as would be required to format and submit a batch capture request to a payment processor.

Once the ccCipher[n], ccIndex[n] and ccMask[n] of the credit card number are created and stored (e.g., indexed and written to database), the clear text credit card number MUST be destroyed. If the clear text value is only stored in shared memory of the e-Business server, the memory allocation MUST be explicitly destroyed (automatic garbage collection is not sufficient). If the clear text of the credit card number is stored in a temporary file, the clear text value MUST be destroyed using [wipe].

One-click Buy Program*

As a convenience to consumers, some merchants implement a type of *loyalty* program focused on streamlining the shopping experience for repeat customers. This program typically involves some type of registration process where data about a *first-time buyer* is collected. Data are frequently stored in database servers. After registration, the consumer is given a username and password, secure token, or some combination of these. When the user shops on the site in the future, they need only reference their membership to use the stored information and make an *instant purchase.*

An Electronic Citadel MUST support the implementation of these one-click purchase loyalty programs should they exist.

For the full period of time that a consumer is a member-in-good-standing with a merchant's one-click program, the target data element (credit card account number) MUST be securely stored and retrievable by the system to complete a transaction.

* One-click buy programs and systems are covered by patents owned by others. The discussion related to this program is for illustration only and to introduce the concept. No recommendation to implement this type of program should be construed or assumed as a part of this discussion. The reader is strongly advised to seek patent advice prior to implementing this type of system. See [5,715,399] for more information.

If a consumer stores multiple credit card account numbers based on some profile with the merchant (an example of this might be a business and personal travel profile with an airline site), a mask of the target data SHOULD be used to allow the consumer to select which card they are referring to. An example of a mask would be: “xxxxxxxxxxxx4019”. This shows the final four digits that refer to the card account number.

The index item or surrogate value of the target data MUST NOT be passed as any obscured value or hidden field to the end user’s browser. The assumption being that hidden fields can be compromised and tampered with.

When used, both mask and key values MUST match the target data being referenced as an additional measure of security against tampering.

Capture of Funds

The prior section described the process associated with obtaining a clear text credit card account number from a consumer as a part of the selling transaction. While the selling process is focused on the obtaining a pre-sale authorization, this section assumes the selling transaction is complete and a pre-sale authorization has been obtained from the e-Business merchant bank processor.

In general, an e-Business issues a request to capture funds from the consumers credit card account after the merchandise or service has been shipped to the consumer. This process is typically initiated at a point in time after the original sale transaction is completed and is usually submitted as a batch (combined with other sale transactions) to the e-Business bank processor. Most bank processor protocols require the clear text credit card number to be submitted in the batch.

In some cases, this process is performed on behalf of the merchant by a payment gateway provider. In this situation, the payment gateway provider would provide these services and appropriated processing of the credit card data.
The clear text credit card number MUST be recoverable from the cipher text credit card number to facilitate capture and settlement of funds.

The system process used to prepare batches for submission to the e-Business merchant bank processor MUST acquire the decryption key from a secure key storage (described in later sections) and use the key to decrypt the cipher text.

Many batch processes create temporary files to support batch processing. If this is the case, the system MUST destroy all temporary files using [wipe] after the batch is submitted.

The system MUST maintain the ability to recover clear text credit card account numbers until verification that funds have been settled and deposited into the e-Business merchant account.

Should the batch capture process fail, the system SHOULD be capable of reprocessing the batch or subset of the original batch. In this situation, the cipher text and key MUST be retrieved and a new clear text credit card account number acquired for the resubmission.

The clear text of the credit card account number MUST NOT be maintained or stored in any file or memory location beyond the time necessary to prepare and submit a capture batch.

Refunds and Returns

This is one of the most frequently overlooked processes in an e-Business site. Obviously, any company selling a product or service on-line must have some mechanism for reversing a payment transaction. This process refers to the case where a consumer contacts the e-Business directly about returning the product or getting a refund. This is NOT referring to a situation where the processor or merchant bank has issued a charge-back.

As in the prior section, this process may be performed on behalf of the merchant by a payment gateway provider. In this situation, the payment gateway provider would provide these services and appropriated processing of the credit card data.

Since there are several business policies that can affect whether refunds are granted or not, let's assume the policy discussion is out of scope and a decision to issue a refund has been made.

As a Best Practice, the consumer SHOULD be in direct communication with the e-Business and SHOULD provide the credit card number from the original transaction to receive the refund. This process is not usually supported by a form that the customer interacts directly with. Instead, the consumer is in voice contact with the e-Business customer support staff.

If the refund process requires the consumer to electronically re-submit the credit card number, the transmission of the credit card number to the business MUST be through a secure form using [SSL].

The ccMask[n] value SHOULD be used to help identify the correct credit card to be requested from the consumer.

Once the clear text value of the credit card number is received by the e-Business, a message digest of the clear text credit card SHOULD be produced (creating refund.ccIndex[n]) and compared with the puchase.ccIndex[n] value from the original purchase transaction. This will enforce a business policy that refunds will only be given to the card used in the original purchase. A policy such as this can be considered a best practice as it will avoid the problem of refund fraud and opportunity for internal abuse by customer support personnel. The following gives an example of this type of process:

At time of original purchase:

Clear text card number purchase.ccNumber: “4111111111111111”
Purchase Mask purchase.ccMask0: “xxxxxxxxxxxx1111”
Purchase Key purchase.ccIndex0:
“aL+zlvNa84dvxQlmWz3COgkwqrE=

At refund time:

Card number presented refund.ccNumber: “4111111111111111”
Refund Mask refund.ccIndex0: “aL+zlvNa84dvxQlmWz3COgkwqrE=

Since the refund.ccIndex0 is the same as purchase.ccIndex0, the e-Business can be assured that the credit card number received for the refund is the same credit card account number used for the original purchase. If the refund.ccIndex0 is NOT the same as the purchase.ccIndex0, the e-Business knows this is NOT the same credit card account used for the original transaction. This assumes that the same message digest algorithm is used to create the ccIndex[n] values.

In a case where the date and time of the original purchase is included during the formation of the ccIndex[n] value, the date and time are retrieved from the merchant’s database, appended to the card number provided by the consumer, [SHA] digest computed, and then compared to the ccIndex[n] value stored in the merchant’s order database. If the computed digests match where the date and time are known to the merchant, the credit card presented for the refund can be declared as the same used during the original purchase.

Technology Issues

This section outlines a set of related technology issues.

General Assumption

A few assumptions related to the pre-existence of systems and use of commerce site products from others.

It should be assumed that many e-Businesses that desire to implement the Electronic Citadel model for protection of consumer credit card data MAY have pre-existing systems.

E-Business management and IT departments SHOULD un-constrained when making selections of commerce site server products or shopping-cart system software products.

The Electronic Citadel model and systems MUST be easily integrated into existing system topologies or used with commerce or database software products from others.

Most e-Businesses will have already decided on the types and models of relational database software products in their enterprises. The only requirement on these systems is that they MUST be capable of storing and retrieving a blob of binary data (chunk of cipher text).

Performance

One of the core requirements (objectives listed in section 3.3.1) is to do nothing that limits the ability of an e-Business to accept and process transactions. Cryptographic software performance can take valuable time from the processing of a customer order. This objective can be interpreted to mean that the processes associated with the Electronic Citadel and e-Business system MUST be optimized to assure that the least amount of actual time is used during the time a consumer is waiting to have their sales order accepted or rejected.

The cryptographic software used to generate [SHA] ccIndex[n] value and [3DES] ccCipher[n] SHOULD be implemented asynchronously to the process of the consumer's order. This is to say that a separate, independent process should be handed the clear text data to process while the e-Bsuiness site is free to conclude the transaction with the consumer.

Further, the data storage functions (interfaces to store and retrieve the data) SHOULD be implemented in a similar asynchronous scheme.

The mechanism for calculating a message digest of a credit card number (index item) during the refund process MUST be done synchronously to the refund process. This assumes that the consumer is in a telephonic session with a support representative and the card number being presented for the refund MUST be compared to the purchase.ccMask[n] (see 3.3.4) prior to accepting or concluding the refund process.

Key Management

This section discusses the handling of encryption keys used in the [3DES] encryption/decryption process. In essence, the key is used by [3DES] to create a cipher text from clear text and, if needed, to create clear text from cipher text.

A very brief understanding of cryptography is needed here. In general, the ability to decrypt data (convert from cipher text to clear text) requires three components: copy of cipher text, algorithm (in this case a copy of [3DES]), and the key such that, cipher text + key + algorithm will always result in clear text. Since hiding the cipher text and algorithm does not guarantee security, secure management of the key is absolutely mandatory and the only true assurance of maintaining secrecy. This principle forms the foundation of the Electronic Citadel.

The key MUST be available to create cipher text as required by the sales order process. The key MUST be stored in an encrypted form and only retrieved (decrypted) into a memory location as a result of receiving an authenticated request from a known host. As an example, the key is stored as a keyCipher value and is requested by a known system and process. The key MUST be recoverable during the actual recovery period as set by business policy. In essence, if there are 12 recovery periods (numbered 0 through 11) based on the months of the year, key[0] MUST be maintained during recovery period 0. Once recovery period 0 ends and period 1 begins, key[0] is destroyed using [wipe] and key[1] becomes the current key.

Reuse or recycling of keys MUST NOT be permitted.

Details about the Electronic Citadel key management process will be described in the following chapter.

Citadel Method and system

From the preceding sections, the reader should understand that the Electronic Citadel is based on these basic facts:

Numerous threats face e-Businesses from both inside and outside of their companies.

Threats and vulnerabilities change constantly.

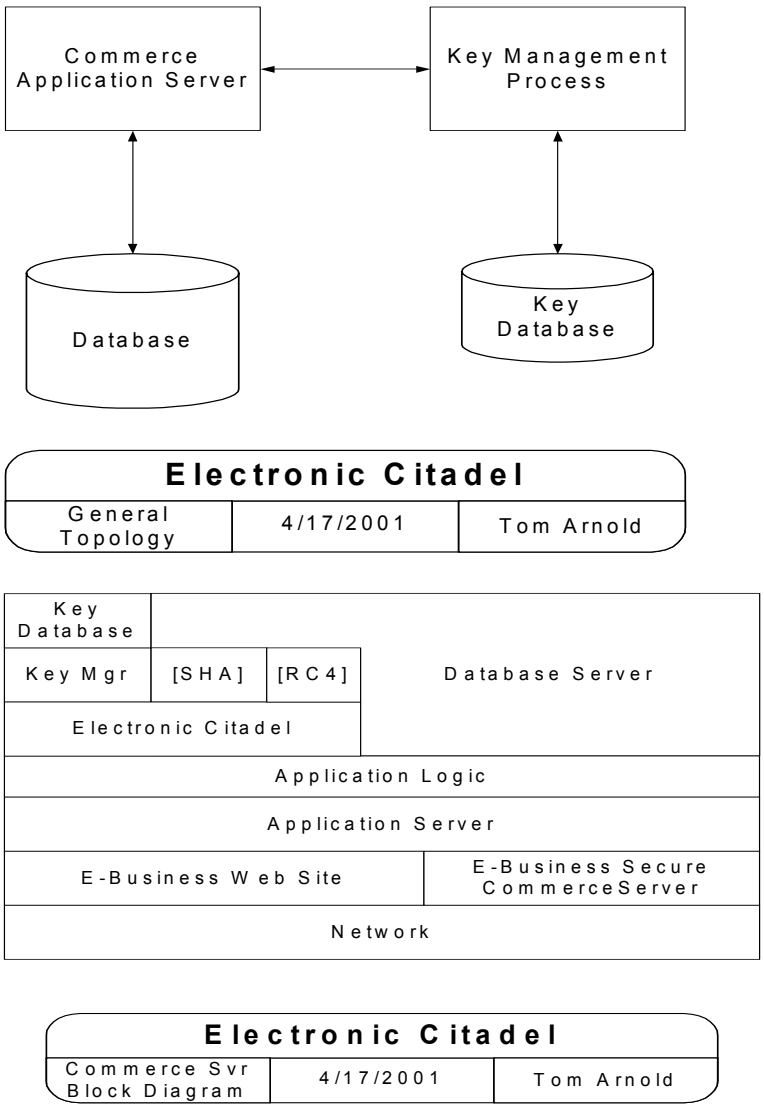
Businesses have limited resources – security isn't the only thing they have to do. Businesses can't build the all-seeing and all-knowing security system for now and into the future.

Business operations and customer expectations are important when considering security systems.

The Electronic Citadel method and system is presented in this section. The architecture describing the system is by no means the only possible solution. It is one example consistent with the Electronic Citadel requirements. This chapter describes the overall topology of the system at a very high level, and then examines individual subsystems. Database attributes as well as input and output data are included to assist in understanding of the model.

High-level components

The Electronic Citadel is a set of interoperating subsystems and routines that may be distributed among application servers. The key management process and associated key database would most likely be shared resources across a secured network environment. They are considered separate in this example. They key management process could co-exist with the commerce application server.



In laying out the Electronic Citadel, one primary consideration was to design a software architecture that would integrate closely with existing e-business commerce applications. The general topology diagram illustrates a system where the only external process supports the key management processes (including key generation, key encryption, key decryption, and key deprecation) and the key database. The cryptographic components of the Electronic Citadel are implemented in software contained in the application server. This allows the Electronic Citadel software architecture to interoperate with the e-business's order management and commerce application platform. For example, if the application server platform were Java based, the Electronic Citadel application would be a package of Java classes that could be

imported into the core commerce application. Since performance is a major issue related to any software routine associated with the acceptance and processing of electronic commerce transactions, this architecture offers the most advantages and allows the Electronic Citadel to scale and be deployed with the application server environment.

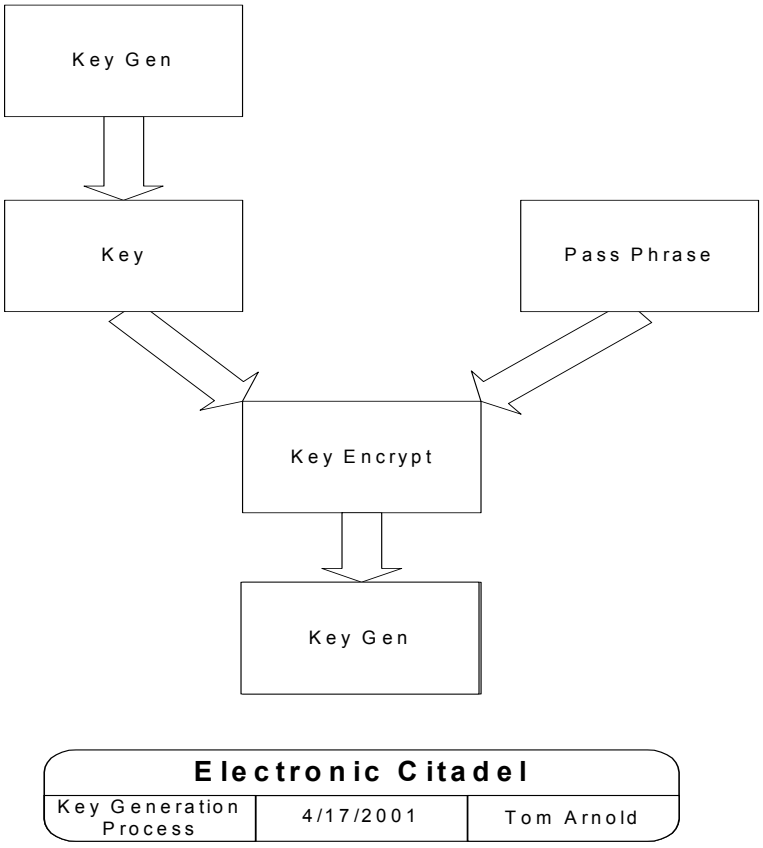
System Processes

This section describes the core functions in greater depth.

Key Generation

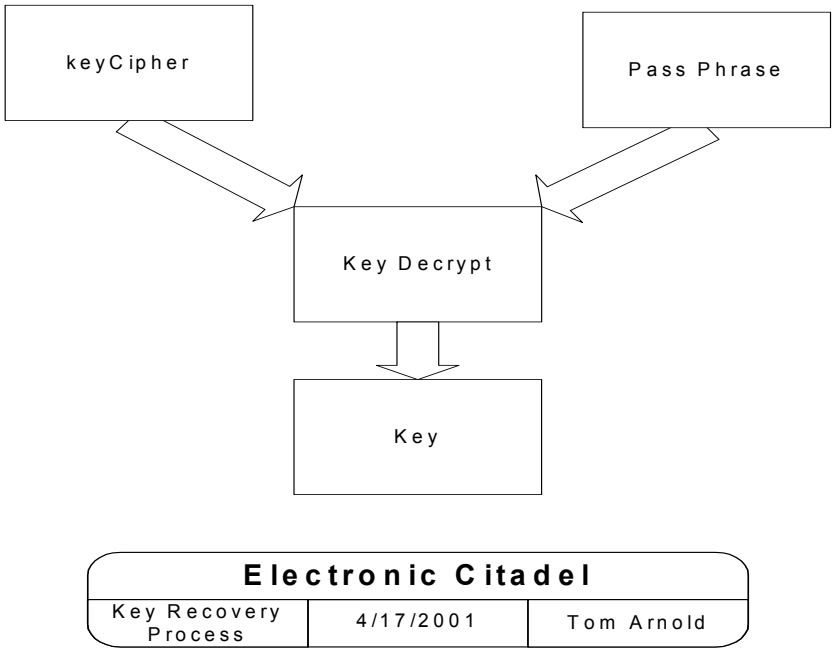
This process generates a key for use by [3DES] for encryption and decryption of the target data. Algorithms used to generate the key can be acquired from [Applied]. The key that is generated should be at least 128 bits in length. If the cipher text of the target data needs to be stored for a period longer than 5 years, this key length should be increased to 256 bits or beyond.

The pass phrase should be a string that is used by the key encrypt function to encrypt the key value.



Key Recovery

Key recovery is the process used by the Electronic Citadel to extract a key from a keyCipher (produced in the key generation process). The key decrypt process to extract the key value uses a pass phrase and reference to the keyCipher. To facilitate application server performance and avoid the timely process of extracting the key value with each transaction being processed, this process should be used when the e-business commerce application server is initialized (or booted). The key value should be stored in memory and used by the Electronic Citadel [3DES] function. The following diagram illustrates the key recovery process.



Recovery Period Process

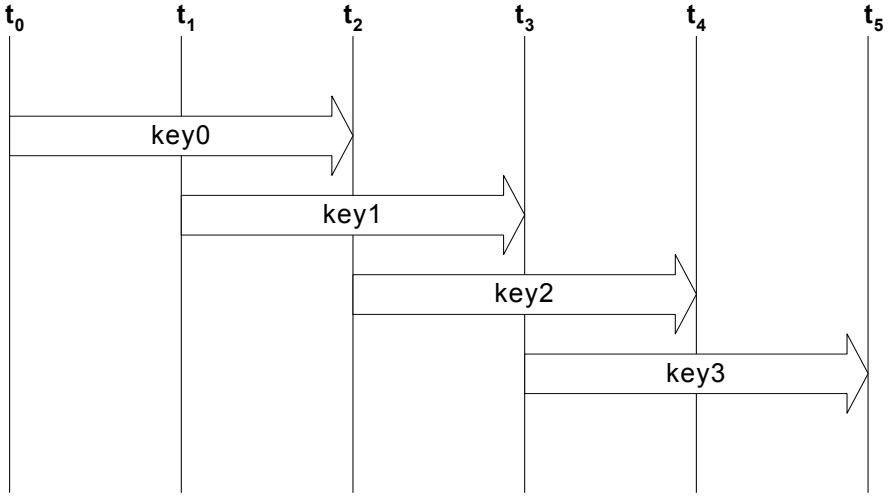
The recovery period process is the embodiment of a method to manage keys. The period refers to an interval of time that a key is maintained before being deprecated and replaced by another key. This method of managing encryption keys is one of the unique, distinguishing features of the Electronic Citadel.

To explain this process, let's consider the period of time that a credit card account number needs to be maintained in a state where the clear text value can be recovered. As you'll recall from the earlier discussions related to selling and capture processes (sections 3.3.2 and 3.3.3 respectively), a clear text credit card account number needs to be recovered from the transaction database at a time when a capture batch is prepared.

Let's assume that a merchant calculates the outer-most bounds of this period of time as 60 days. Their internal statistics say that well over 99% of customer orders will either be fulfilled or canceled within 60 days of the purchase transaction. Over 90% of the orders will achieve this final state within 45 days. Given this data, the recovery period illustrated in the following diagram represents a 60-day period of time. The interval from t_0 to t_1 is 30 days; from t_1 to t_2 is 30 days; and so on. The first key value (key0) is generated and used for encryption and decryption of credit card account data from t_0 to t_1. At t_1, a new key value (key1) is generated and used to encrypt and decrypt

transactions, while key0 is used only to recover transactions from the prior interval. At t_2, a key2 is generated and used for encryption and decryption of transactions, key1 is used only to decrypt transactions from the t_1 to t_2 period, key0 is deprecated and destroyed. By destroying key0, the Electronic Citadel implements a mechanism of one-way encryption to provide the ultimate security for target data.

In this example, any credit card account numbers accepted during the first period (time between t_0 to t_1) can *never* be restored to clear text. This condition extends to any copies of database backup tapes or even full disk storage devices that might be removed from a storage subsystem. Since the key cannot be recovered, the clear text would only be retrieved through a brute force attack on the cipher text. Given the strength of the original key (e.g., length of key), this type of attack could take many years to complete. This added measure of security reduces further the likelihood of an intruder being able to acquire the clear text version of the target data from either on-line databases or backup tapes.



Electronic Citadel		
Recovery Period Process	4/17/2001	Tom Arnold

One might ask, why overlap the keys? This is to allow for continuous transaction processing during the brief period of time while the next encryption key is being generated. This is very important since most e-business systems cannot accept any downtime, no matter how brief. Further, the process of generating a key, creating the keyCipher object, and storing the keyCipher object in a key database can take several minutes. If the issue of allowing system downtime while creating a new key value is not

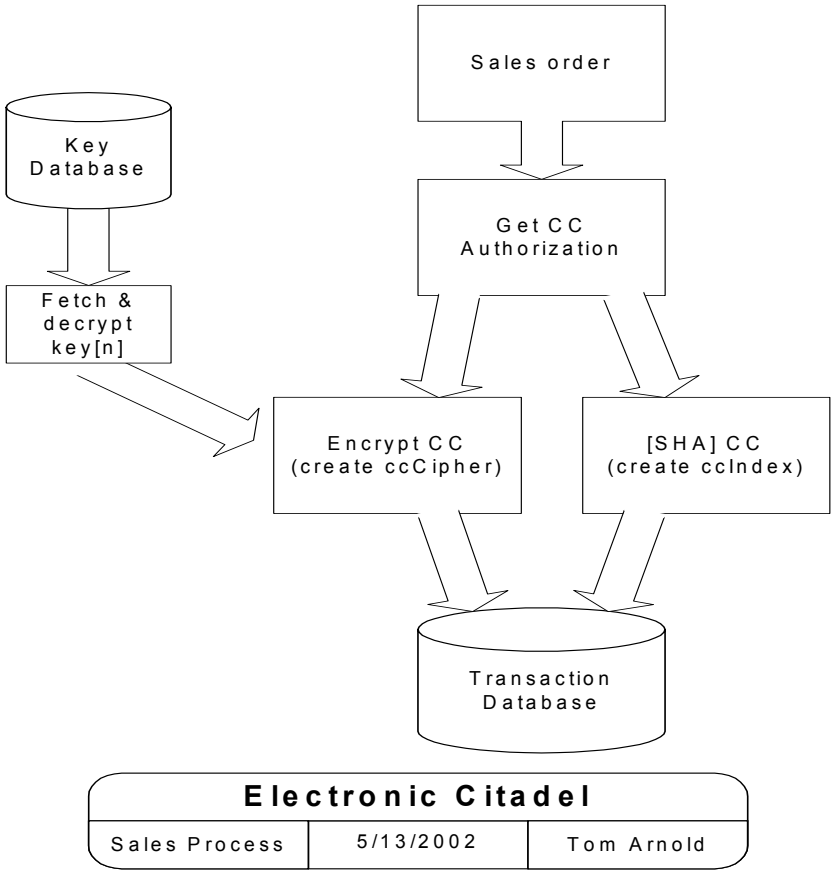
a concern, this diagram could be redrawn to accommodate a single key for any recovery period.

It is important to note that the creation of an index and mask as provided by the Electronic Citadel method and system provides an important tool whereby a merchant can still verify the correctness of a target data item presented at a time outside of the recovery period. An example would be processing a refund request and the need to validate the credit card number presented at refund was the same card used to make the original purchase.

Selling Process

The selling process has been included to describe how an e-business's commerce application server would process a customer purchase transaction. This is presented to further the illustration of the Electronic Citadel. It is fully feasible to utilize this security model in different business situations.

The sales process is illustrated in the following drawing:



In this case, the sales order represents the step in the transaction process where the e-business commerce application server has received a credit card from the customer. The step involving getting the credit card authorization from a bank processor is actually out of scope of the system and is illustrated as a point of reference to the larger selling process cycle.

The fetch and decrypt key step would most likely have been performed at the time that a commerce application server is initialized. In actuality, this step would have to be performed at a time after a key is published. Since the fetching of a key involves having to use a pass phrase and go through a key decrypt process to yield the actual key value, most e-business sites may want to consider storing the encryption key in a memory location on the secure commerce application server machine.

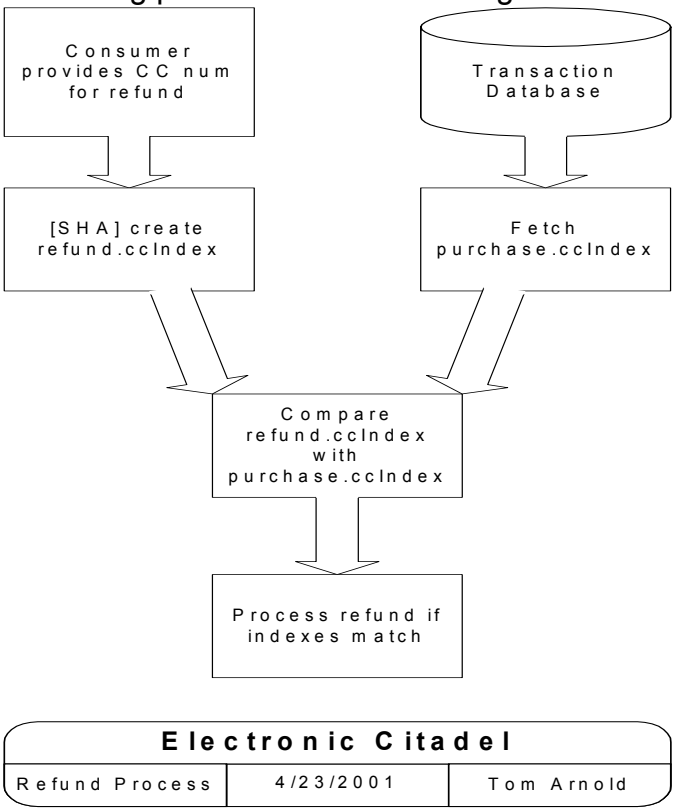
Assuming the credit card account number has received an authorization and is ready for storage, the value is submitted to the Electronic Citadel interface and a cipher text and index is passed back to the calling application. In terms used before, this step

would yield the ccCipher and ccIndex values for the clear text credit card number. The original target data MUST never be passed back to the calling application. Call with clear text, receive index item and cipher text back.

Refund Process

The refund process illustrates how the Electronic Citadel might be implemented in an electronic commerce environment within an e-business. The diagram at the end of this section illustrates the process.

Assume that the business has a policy that refunds can only be accepted on the credit card account number from the original purchase. This is a common policy for combating possible internal fraud against the e-business.



In this case, the merchant customer support person has determined that a refund should be processed for a customer. The merchant has received the full credit card account number from the customer. The support person into a user interface on the merchant's commerce application server then keys this value. Prior to submitting the

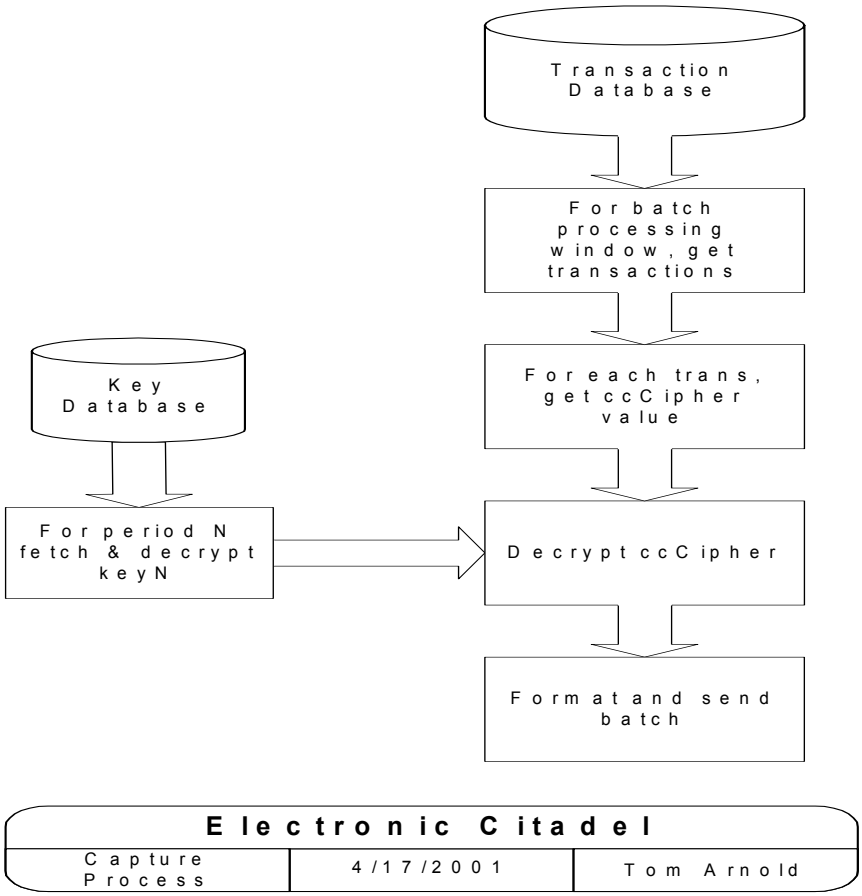
refund request to the bank processor, the application requests the Electronic Citadel to create an index item from the refund card number. This creates and returns to the application a refund.ccIndex value. The application can then fetch the purchase.ccIndex value from the e-business's transaction database, compare the two values, and determine if this is the same card number as presented during the original transaction. If the refund.ccIndex and purchase.ccIndex values do NOT match, the application knows to immediately notify the support person and stop further processing.

A process similar to this can be constructed to support any target data that might need to be referenced in the future. Imagine using this in a medical record system to validate social security numbers.

Capture Process

The capture process has been included to describe how an e-business's commerce application server would retrieve the credit card number so that additional processing can be performed. As with the prior sections, this is presented to illustrate the Electronic Citadel. It is feasible to utilize this security model in different business situations.

In this example, the e-business has concluded shipping the product or service to the consumer in fulfillment of a sales order. The e-business system must retrieve the credit card account number as clear text, append this with associated transaction data, format a batch file for transfer to a bank processor, and then send the batch file to the bank. This process illustrates how the Electronic Citadel can retrieve a clear text version of the target data within the appropriate recovery period.



At a specified batch processing time, all transactions that are queued for processing obtain the stored ccCipher value, and period identifier from the transaction database. For each ccCipher and Recovery Period Identifier, the appropriate key is obtained from the key database, and the ccCipher value is decrypted, giving the clear text credit card number. This is formatted into a batch file and transferred to the bank processor. As a reminder, any work or temporary files generated during the capture process MUST be destroyed using [WIPE] once out of scope.

Proposed Electronic Citadel Software interface

This section describes the public interface for the Electronic Citadel class libraries. Protected and private classes and members are not described in this paper and are dependent on the underlying security libraries being implemented.
For instance, it is not beyond reason to substitute algorithms like [DES] or [AES] for [3DES]. This decision should be fully left up to system performance requirements as

well as budget constraints (as some cryptographic tools must be licensed from the patent holder).

The following is provided as a template of a Java interface and not intended to be a complete class listing. At some future time, a reference implementation of the Electronic Citadel may be posted as an open software reference.

```
package com.companyDomainName.ecitadel.util;

public interface Ecitidel {

// For a given recovery period, generate
// and store key in key database
public void keyGen (int period,
        String passPhrase);

// Retrieve a Base64 key given period and pass phrase.
public String keyRetrieve (int period,
            String passPhrase);

// Deprecate key for a given period. Wipe the key out.
public void keyDeprecate (int period);

// Get a index value of target string as Base64 encoded string
public String getIndex (String targetValue);

// Get Base64 cipher text of targetData string, using key
// and algorithm as DES, 3DES, AES
public String encrypt (String key,
            String clearTarget,
            String algorithm);

// Get clear text of cipher text using a key value
// and algorithm as DES, 3DES or AES
public Sting decrypt (String key,
            String cipherText,
            String algorithm);

}
```

References

[3DES] ANSI X9.17, “American National Standard for Financial Institution Key Management”, American Bankers Association, 1995.

[5,715,399] US Patent, “Secure method and system for communicating a list of credit card numbers over a non-secure network” 5,715,399, US Patent and Trademark Office, May 30, 1995.

[Applied] Schneier, Bruce, Applied Cryptography, Second Edition: Protocols, algorithms, and source code in C, John Wiley and Sons, Inc., 1996.

[CERT] CERT® Coordination Center, http://www.cert.org, Carnegie Mellon University.

[CYBNOTE] National Infrastructure Protection Center, “Cyber Notes”, http://www.nipc.gov/cybernotes/2001/cyberissue2001-06.pdf, Issue 2001-06, March 26, 2001

[DES] FIPS 46-2, “Data Encryption Standard (DES)” Federal Information Processing Standards Publication 46-2, *http://www.itl.nist.gov/fipspubs/fip46-2.htm,* 1993

[MD4] Rivest, R., “The MD4 Message Digest Algorithm”, RFC 1320, Internet Engineering Task Force, April 1992.

[MD5] Rivest, R., “The MD5 Message Digest Algorithm”, RFC 1321, Internet Engineering Task Force, April 1992.

[NIPC 01-003] National Infrastructure Protection Center, “Update to NIPC Advisory 00-060 "E-Commerce Vulnerabilities", http://www.nipc.gov/warnings/advisories/2001/01-003.htm, FBI, March 8, 2001

[NIPC] National Infrastructure Protection Center, http://www.nipc.gov, FBI

[RC4] Rivest, R, “The RC4 Encryption Algorithm”, RSA Data Security, Inc., March 1992.

[RFC 2119] Bradner, S, “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, BCP-4, IETF, March 1997.

[SHA] FIPS 180-1, “Secure Hash Standard” Federal Information Processing Standards Publication 180-1, *http://www.itl.nist.gov/fipspubs/fip180-1.htm,* 1995

[SSL] Rescorla, E., “HTTP over TLS”, RFC 2818, IETF, May 2000
Dierks, T. et. al., “The TLS Protocol Version 1.0”, RFC 2246, IETF, January 1999

[WIPE] National Computer Security Center, “A Guide to Understanding Data Remembrance in Automated Information Systems”, NCSC-TG-025 Version 2, Sep 1991.

About the Author

Mr. Arnold (toma@coolgamestuff.com) is Chief Software Architect and VP of Product Strategy for the Merchant Services Division of InfoSpace, Inc. He has an extensive background in Internet e-Business systems and electronic commerce, having published several white papers, designed and consulted to some of the most successful electronic businesses, and provided policy guidance to the US Government policy makers.

Prior to joining InfoSpace, he was Chief Technical Officer for CyberSource Corporation where he designed and deployed the full suite of Internet Commerce Services for the Company, including payment processing, fraud detection, electronic software distribution, and other commerce systems. He has a strong background in software engineering and the development of systems technology within both the public and private sectors over the past 16 years. Prior to CyberSource, Mr. Arnold managed an applications development team at Silicon Graphics, Inc., building the next generation of electronic sales, software distribution and software licensing systems.

Prior to Silicon Graphics, Inc., Mr. Arnold lead a software development team building on-line transaction processing and database systems at NASA / Ames Research Center.

Mr. Arnold's diverse career and background include design of automated manufacturing systems, deployment of early law enforcement and criminal intelligence computer systems, to working as a police officer investigating crimes in the late 1970s. In May 25, 1999, Mr. Arnold gave expert testimony to the US House of Representatives, Committee on Commerce, Subcommittee on Telecommunications, Trade and Consumer Protection to assist in their deliberations on the proposed SAFE Act, related to the use of encryption technology to secure and protect Internet e-Business trading.. Then, on June 10, 1999, Mr. Arnold testified before the US Senate, Committee on Banking on the security and technology impact of the proposed Export Administration Act of 1999. Since that time, he has been consulted by numerous regulatory agencies including the Department of Commerce, Department of Treasury, Department of Justice, and Fair Trade Commission on security, Internet commerce, identity theft, fraud, consumer protection and consumer privacy.

Acknowledgements

The author wishes to thank Jason Eaton, Jungawunga; Mike Jiminez, CyberSource; Jessy Rendleman (a great engineer); Lauren Hall, Microsoft; Fred Hoch, SIIA; John Pettitt, Cloudview (a great CTO); Jennifer Jennings, CyberSource; Paul Guthrie (another great CTO); Wes Wilhelm and Allen Jost from HNC (gave a timely review and great comments); and several others.