A Secure End-to-End System for M-Commerce: Research Paper CS03-24-00

Francois Kritzinger Duncan Truter *Supervisor: Professor Ken MacGregor*

October 12, 2003

Abstract

our project was to implement a Secure End-to-End M-Commerce system. We have called it SeMCom. We have implemented a Client application, a transaction server, a virtual wallet on the transaction server for the client, and a bank server. Semcom conforms to all of the security requirements.

All of the links in our system have end-to-end security, and conform to the requirements for data integrity, Authentication, Non-Repudiation and data confidentiality.

SeMCom conforms to the Secure Electronic Transaction standard (SET), but is also a much simpler implementation than most of the existing implementations today.

1 Introduction

Mobile commerce or M-commerce can be described as the act of performing an electronic transaction that has financial implications from a mobile device such as a cellular phone or Personal Digital Assistant (PDA). An example of this would be the purchase of an item from a shop using a mobile device to perform the entire transaction.

The term "electronic transaction" means that the transaction details are electronically transferred from the user's mobile device, over one or many public cellular and/or computer networks, to the party that is supposed to accept the transaction (which, in our case, is the merchant's bank).

Mobile commerce is a relatively new development, and for this reason there are no industry-wide accepted standards for securing it as of yet. The goals of this project was to design and implement a secure end-to-end m-commerce system based on the proposed architecture (see figure 1).

In principal, the only difference between m-commerce and e-commerce is that the customer is connected to a wireless link as opposed to one consisting of physical wires or cables such as telephone lines and network cables. From this perspective, existing e-commerce systems could be adapted to suit m-commerce without much effort.

Security is a crucial requirement of an m-commerce system due to the fact that the sensitive financial information that these systems transmit travel over untrusted networks where it is essentially fair game for anyone with local or even remote access to any part of the path followed.

The suggested architecture (see figure 1) is significantly simpler than existing e-commerce architectures such as SET (Secure Electronic Transaction). In SET, the financial transactions are performed by the merchant, introducing a level of distrust which adds significant complexity to the system's security. This is illustrated by the sheer size of the SET specification (971 pages [2]).

The goal of this project was to implement and/or acquire the necessary software in order to implement an end-to-end secure m-commerce system on top of the specified architecture. This goal was achieved and resulted in a system called *Semcom (Secure End-to-end M-COMmerce)* which is much simpler in design, but which, at the same time, satisfies most of the requirements of SET.

This paper discusses the development of Semcom and how it manages to satisfy the requirements of a standard as complex as SET while being built on top of an architecture which is much simpler in comparison.

2 Background and Motivation

2.1 Cryptography

Before discussing cryptography, we present a few important definitions:

Plaintext: The original, unmodified mesage or data which is used as input to the encryption algorithm.

Ciphertext: The scrambled, encrypted message that is produced by the encryption algorithm.

Encryption algorithm: The algorithm performs various substitutions and transformations on the plaintext (input) to produce the ciphertext (output).

Secret Key Used as input to the algorithm; affects the exact substitutions and transformations performed by the algorithm on the plaintext.

Decryption algorithm: Essentially the encryption algorithm run in reverse in order to transform the ciphertext back into plaintext (decryption).

Cryptography is the process of transforming the original plaintext message into scrambled, unreadable ciphertext using an encryption algorithm whose mathematical operations on the plaintext are controlled by the encryption key.

An important note is that the security of encryption depends on the secrecy of the key, not the algorithm. Good encryption algorithms are well known to the public, allowing them to be carefully analysed for flaws.

2.2 Conventional/Private Key Cryptography

Conventional encryption algorithms use the same key for encryption and decryption. If more than one party is involved in the operation, this secret key needs to be known to them all, which may cause logistical problems if many parties are involved.

The upside of a shared secret key is that the encryption process significantly faster compared to public-key cryptography. This is due to the fact that public key encryption algorithms need to perform more complex operations in order to function with two different and complimentary keys.

2.3 Public Key Cryptography

There are several algorithms that make use of this cryptographic method. In Public key cryptography, the secret information is not the algorithm used to encrypt and decrypt the data, but the keys used. RSA, for example, is a very well-known and established public key algorithm that generates keys and uses them for encryption and decryption.

The algorithms used in public key cryptography make use of two different and complimentary keys; one is called the *private key*, and the other is called the *public key*. These keys are complementary in the sense that a message encrypted with one can only be decrypted by the other. For example, a message encrypted with the private key can only be decrypted with the public key, and *vice versa*. It is not computationally feasible to derive one key from the other. the RSA algorithms of Public key cryptography has been tested a lot, and has been found to be secure for keys of a large enough size.

Usually, the private key is kept secret by the user, while the public key is well-known and available to the public, and needs to be known to any party that needs to be able to send encrypted messages to the user. The public key may be easily avaliable, but the private key would be required to decrypt the message so this method is secure. It can be done the other way around, (the sender uses their own private key to encrypt the message) but the application for that is to sign a message because anyone would be able to decrypt the message with the sender's public key, so that is not secure, but proves the identity of the sender.

Key size is a very important issue: 128 bits used to be considered secure, but now the current key size that is considered 'unbreakable' is a 1024-bit key [2].

2.4 Data Encryption Standard (DES)

DES is one of the most commonly used and most analysed encryption standards in existance. Its underlying algorithm is the Data Encryption Algorithm (DEA). It is a block cipher with a block size of 64 bits and has a key length of 56 bits.

2.4.1 Background and Security Issues

The main vulnerability of DES has proven to be the key length, which had been a cause for concern from the beginning. All of the successful attempts at cracking DES had been through the use of brute force attacks, which becomes easier and easier to succeed with as computers become more powerful and less expensive. It has been shown that a machine can theoretically be constructed that would be able to crack DES in around three-and-a-half hours and would cost a relatively modest one million US dollars to construct [3].

Regardless of these security issues around its key length, today DES is *the* most widely used encryption scheme [2]. It is widely used in the financial industry to protect sensitive online applications [3].

Although the short DES key length presents a major vulnerability, the algorithm itself has proven to be very secure: because DES is such a widely deployed standard, it has become the most-studied encryption algorithm in existence and, despite numerous attempts using numerous approaches, no one has so far publicly acknowledged having discovered a fatal weakness in the algorithm [2].

2.5 Triple DES (3DES)

Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985, and was incorporated as part of DES in 1999 with the publication of FIPS PUB 46-3 [1] [2].

The only difference between DES and 3DES is that 3DES uses three keys and performs three executions of the DES algorithm.

The 3DES encryption process is as follows:

$$C = E_{K_3}[D_{K_2}[E_{K_1}[P]]]$$

where

$$\begin{aligned} C &= \text{ciphertext} \\ P &= \text{plaintext} \\ E_K[X] &= \text{encryption of } X \text{ using key } K \\ D_K[Y] &= \text{decryption of } Y \text{ using key } K \end{aligned}$$

The decryption process is the same as above, but with the three keys used in reverse:

$$P = D_{K_1}[E_{K_2}[D_{K_3}[C]]]$$

As can be seen from the above, 3DES uses an Encrypt-Decrypt-Encrypt (EDE) process. The only significance of the decryption in the middle is that it makes 3DES fully backwards-compatible with single DES, which is equivalent to a 3DES encryption with all three keys being the same:

$$C = E_{K_1}[D_{K_1}[E_{K_1}[P]]] = E_{K_1}[P]$$

If three *different* keys are used, then the effective key length is 168 bits. This is a substantial improvement over the 56-bit key length used by DES and makes the successful use of brute force attacks almost impossible. Furthermore, because 3DES uses the same underlying algorithm as DES (DEA), it can claim the same resistence to cryptanalysis based on the algorithm as DEA [2].

2.6 Message Authentication

Encryption provides protection against passive attacks such as the release of message contents. On the other hand, message authentication provides protection against active attacks such as the falsification of data and transactions, whereas encryption provides protection against passive attacks such as the release of message contents.

A message is said to be authentic if it is genuine and came from its alleged source [2]. Message authentication is a procedure that allows communicating entities to verify that received messages are authentic.

The two most important things to verify are that the message has not been altered or modified after being sent, and that the message is in fact from an authentic source. It may also be necessary to verify the timeliness (i.e. that it has not been artificially delayed or replayed) and/or sequence of the message relative to other messages recieved from the source in question [2].

2.6.1 Message Authentication Codes (MACs)

MACs are a commonly used technique for message authentication in which a small block of data (the message authenication code) is appended to the message before it is sent. The MAC is calculated as a function of the message M and a secret key K which is shared between the two communicating parties A and B. The idea is that every possible message has a unique MAC, and that it be impossible to generate M from its MAC algorithmically (called the "one-way" property).

The process is as follows: A generates the MAC for a message M and appends it to the message and sends it over the network. Upon receipt, B removes the MAC from the message, and performs its own calculation of the MAC based on the contents of M and K. It then compares its version of the MAC to the one received with the message. If they are identical, the message is authentic, and B can be certain of the following facts:

1. The message has not been altered.
2. A was the sender of the message.
3. The message sequence number is correct (if the message did in fact contain a sequence number).

B can be certain of point 1 because if an attacker had modified the message, she would have had to modify the MAC as well, which would have had to be computed from the newly modified message. This would only have been possible if the attacker had known the secret key.

Point 2 is a certainty because it is assumed that only A knows the secret key, and as such is the only entity able to generate the correct MAC for M.

Point 3 is certain for the same reason as point 1. B can be certain that the messages have not been reordered or lost [2].

Any number of algorithms could be used to generate this MAC, but DES is the one recommended by FIPS PUB 113. An encrypted version of the message is generated using DES, but only the first N bytes are used as the MAC.

LibX9 uses MACs generated with the DES algorithm for message authentication, as recommended in X9.9 [5] (see §2.15).

2.7 Hash functions and Digital Signatures

Hash functions take as a parameter a variable length of bits, and output a fixed size value called a hash code. The hash code can't be used to work out the actual message itself. These are used as follows: a message is written, and when it is ready to be sent its hash value is taken. this is appended to the message and the resulting message is encrypted. Now if the message is changed in transit, the hash code will almost certainly be invalid. Algorithms that are commonly used for this, are MD5 and SHA-1. Sometimes a hash algorithm is used as a digital signature. if confidentiality isn't an issue, they can be used like this: a person writes a message, hashes it and encrypts the hash with his private key. This shows everyone that he wrote it, and that the message is unchanged.

Unlike traditional paper signatures, digital signatures authenticate both the user *and* the document. Each digital signature is unique to the document being signed, and associates the user with that document. For this reason, digital signatures are more secure than paper signatures.

2.8 Digital Certificates

Because anyone can generate key pairs, there are ways to compromise the security of public key cryptography. For example, a malicious party could set up an impostor server and then provide their public key to the user. Digital certificates prevent this kind of fraud by providing a secure, authenticated way of distributing public and private keys. Digital certificates are also used to authenticate the parties invlovled in the transaction so that everyone can be confident that they know who they are communicating with.

A digital certificate contains its owner's identitiy, public key and other information that is needed to authenticate the certificate. The certificate itself is encrypted with a *certificate authority's (CA)* private key. Third parties such as VeriSign, RSA Security and Thawte act as CAs, providing a "respected, independent resource to issue keys and certificates to their holders." [9]

Digital certificates can be used for authentication. If entities A and B establish a connection, sometimes only one-way authentication is required, and sometimes two-way (or even 3-way) authentication is required. This is a description of these methods:

1-way authentication: A authenticates itself to B by sending B its certificate and proving that it has the secret key matching the public key on the certificate. This is useful for applications where the client has nothing to lose if someone else masquerades as the server and gets some of his confidential information. This is not the case in this application. If A is pretending to be someone else and sending that person's certificate, it will be easy enough to find out: when B sends any encrypted data to A (encrypted with A's public key) A won't be able to decrypt it.

2-way authentication: where the client AND the server authenticate themselves to each other by sending each other their certificates and proving their identities by means of the secret keys as described above. This is sufficient for the purposes of our application.

2.9 SIM cards

In a secure mobile syste, a *Subscriber Identity Module (SIM)* card is often used to authenticate the user of the mobile device (the client). These cards are used to store the client's authentication information, such as her private key that is used for encryption and for digital signature generation. In order to authenticate the user to be able to use the device, the a *PIN (Personal Identification Number)*, will be prompted for, which will then be used to access their private key.

SIM cards are tamper-proof. The data stored on a SIM card is encrypted and can only be unlocked with a PIN. For this reason, the use of SIM cards for user authentication in an m-commerce system is very reliable and secure.

2.10 Wireless Transport Layer Security (WTLS)

Wireless Transport Layer Security (WTLS) is a communications security protocol that is similar to *Secure Sockets Layer (SSL)*, but has been made simpler and more lightweight in order to be more suitable for the wireless environment with its lower data rates, higher latency and weaker computational power.

WTLS, like SSL, makes use of public key encryption and hash codes to ensure secure wireless data transmission. It is becoming the standard for providing authentication, data integrity, and privacy to wireless applications.

2.11 Secure Electronic Transaction (SET)

SET is an open security specification which was designed with the primary goal of enabling secure credit card transactions on the Internet. It was created in response to a call for security standards by MasterCard and Visa in 1996. Some of the companies that were involved in the development of the initial specification are IBM, Microsoft, Netscape, RSA, Terisa, and Verisign. The concept has since been through numerous tests, and the first wave of SET-compliant products started appearing in 1998 [2].

SET is not a payment system, and can better be described as a set of security protocols and formats that facilitates the use of existing credit card payment systems on an open, public network, such as the Internet, in a secure fasion [2].

The requirements that SET fulfills are as follows:

Confidentiality of payment and ordering information: It is necessary to assure customers that this information is secure and only accessible by the intended recipient.

Integrity of all transmitted data: It is necessary to ensure that no undetected changes in the content of transmitted messages occur during transmission. SET uses digital signatures (similar to MACs) to this end.

Authentication of cardholder: SET uses digital signatures and digital certificates for this purpose.

Authentication of merchant: Cardholders need to be albe to identify merchants with whom they can conduct secure transactions. SET uses digital signatures and certificates to this end.

Best security practices and system design techniques: SET is a well-tested specificaton based on highly secure cryptographic algorithms and protocols.

Independent of transport security mechanisms: SET can securely operate over TCP/IP, but, at the same time, does not interfere with the use of other security mechanisms such as IPSec and SSL/TLS.

Interoperability among software and network providers: The SET protocols and formats are independent of the underlying hardware platform, operating system, and Web software [2].

The customer's payment and order information get transmitted from the customer's device (e.g. computer) to the merchant, which then connects to the third party responsible for processing mechant payment messages. Thus, at some point during the transaction, the merchant is in posession of the customer's payment (credit card) information. It would not be acceptable for the merchant to be able to read this information. Additionally, the third party payment processing entity should not be able to read the the customer's order information. This introduces a challenging problem which is solved in SET by a combination of a hashing and encryption called a *dual signature*. A detailed explanation of how dual signatures work is beyond the scope of this report. However, the bottom line is that, with the use of dual signatures, the order and payment information can be sent from the customer to the merchant to the payment processor, without the merchant being able to read the payment information or the payment processor being able to read the order information. Dual signatures also ensure that it is impossible to disassociate the order and payment information once they have been "signed" and associated with each other [2].

This example serves to illustrate one of the many complexities imposed by the architecture of existing e-commerce systems, of which SET is one of the most prominent.

2.12 ANSI X9-Series of Standards

The X9 Committee is accredited by ANSI and develops technical standards for the financial services industry. Its mission is to *"develop, establish, publish, maintain, and promote standards for the Financial Services Industry in order to facilitate delivery of financial products and services"* [8]. Some of its most noteworthy strategic objectives are to:

- Provide a common source for all standards affecting the Finance Services Industry.
- Promote use of Financial Services Industry standards.
- Participate and promote the development of international standards [8].

Its inter-industry voting membership includes over 300 organizations representing investment bankers, banks, software and equipment manufacturers, printers, credit unions, depositories, government regulators, associations, consultants, among others. Its standards are reviewed by ANSI before publication and its operations are audited by ANSI every five years.

This project implements a number of these X9 standards in libx9, including:

X9.26-1990 (Financial Institution Sign-On Authentication for Wholesale Financial Transactions)

X9.9-1986 (Financial Institution Message Authentication (Wholesale)

X9.23-1988 (Financial Institution Encryption of Wholesale Financial Messages)

2.13 X9.23 (Encryption)

2.13.1 Scope

X9.23 (Financial Institution Encryption of Wholesale Financial Messages) specifies a method for the encryption of wholesale financial messages (such as wire transfers), or encryption elements within messages, in order to provide confidentiality. The encryption algorithm specified by X9.23 is DES. However, in light of single DES's vulnerability to brute force key guessing attacks, Semcom uses 3DES for its implementation of X9.23 instead.

2.14 X9.26 (Sign-on Authentication)

X9.26 specifies peer entity sign-on authentication. In the following paragraphs, the *requestor* is the entity requesting sign-on authentication, while the *grantor* is the entity who grants or denies authentication. The protocol specified by X9.26 uses the TVP as a "challenge" from the grantor to the requestor, who is then required to reply with a "response" which contains the TVP encrypted with the shared secret key using 3DES in ECB mode. If the grantor decrypts this TVP and the result matches the generated TVP, sign-on authentication is successful.

2.15 X9.9 (Message Authentication)

X9.9 specifies a method to authenticate financial messages including funds transfers (e.g. wire transfers), letters of credit, security transfers, loan agreements, and foreign exchange contracts. The authentication algorithm specified by X9.9 is based on the Data Encryption Algorithm (DEA), which is used by 3DES, in ECB mode.

The main protection provided by X9.9 is that against accidental or deliberate alteration or duplication of messages between the originator and the recipient. Furthermore, the proper use of X9.9 implicitly verifies the identity of the originator.

Protection against message duplication is provided by the inclusion of the date of generation and a *unique* message identifier (MID). In the case of libx9, the MID is a sequence number, which is how it provides protection against duplication.

The message originator generates a Message Authenticaton Code (MAC) by applying the authentication algorithm to the message with a secret key shared between originator and recipient as input.

The MAC is to be included in the message before being sent to the recipient. The recipient then computes its own version of the MAC and compares it to the attached MAC. The equality of the computed MAC to the attached MAC will constitute authentication of the message.

Upon receipt of the message, the recipient removes the MAC and computes its own version over the rest of the message. If the received MAC is identical to the newly computed version, authentication is successful.

Authentication may fail due to a number of reasons such as message structure rule violations (e.g. nested field delimiters or missing fields), or an attached MAC that does not match the computed MAC.

3 Method/Approach

3.1 Architecture

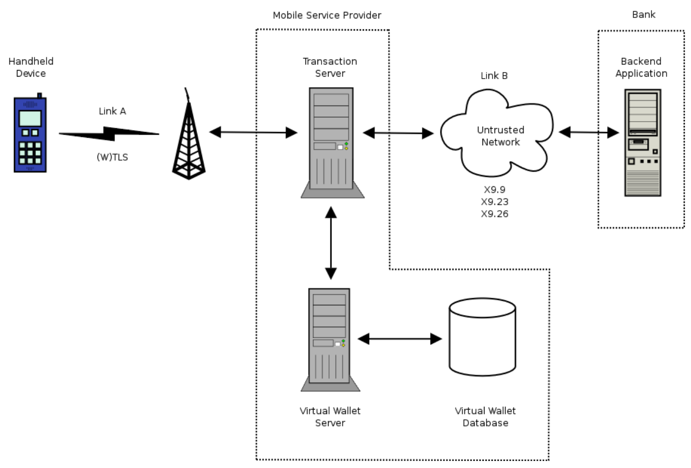

Figure 1: System Architecture

The connection between the mobile device and the transaction server is a wireless or cellular connection, so it may use cellular protocols such as GSM or UMTS, or computer communication protocols such as TCP/IP over WI-FI (IEEE 802.11). This connection shall henceforth be referred to as "link *A*".

The connection between the Transaction Server and the bank application is over an untrusted TCP/IP computer network such as the Internet. This connection shall henceforth be referred to as "link *B*".

The system we implemented consists of four main parts (see figure 1):

The Mobile Application: We implemented this on a PC using OpenSSL, because we didn't have access to a mobile device or a mobile device emulator. this is link A

Transaction Server: The transaction server (TS) resides on the MSP's premises. It will accept requests from the mobile device. It will be responsible for the authentication of the customer, the retrieval of the customer's credit details from the Virtual Wallet Server, and the transferral of currency to the value of the transaction amount into the merchant's bank account over link B.

Virtual Wallet Server: The virtual wallet server's principal responsibility is to respond to the transaction server's requests for customer credit information by retrieving this information from the virtual wallet database to which it is connected and sending this information back to the transaction server.

Bank/Backend Server: This is the application running on the merchant's bank's server. It processes electronic funds transfer messages received from the Transaction Server.

3.2 General Operation

When the customer is ready to make her purchase, she enters the transaction amount and the merchant's bank's identifier and account number into the handheld application which sends this information to the transaction server over a TLS (Transport Layer Security) connection. The TLS connection is established using authentication data (e.g. an encryption key, digital certificates, etc.) stored on the smart or SIM card which would typically be inside the handheld device.

Upon receipt of this information, the transaction server sends a request in the form of a token representing the identity of the customer to the virtual wallet server. The identification token is derived from the device ID of the handheld device used by the customer. Currently this token is in the form of a username.

The virtual wallet server looks up the customer's username in the virtual wallet database to which it is connected, and if such a virtual wallet exists, sends back the customer's credit balance. If the customer has sufficient credit, the transaction server transfers the money into the merchant's bank account by connecting to the bank server, which returns an indication of success to the transaction server. This completes the transaction.

3.3 Security

As explained by the problem definition, the security requirements of connection B are:

1. confidentiality
2. peer entity authentication
3. message (data) authentication
4. integrity

It is assumed that the virtual wallet server is located on the service provider's premises along with the transaction server, which means that the security of the connection between them is not as critical as the others.

3.3.1 Connection A

Connection A was implemented using OpenSSL. Requirement 1, confidentiality, is satisfied by the session key that is established in an OpenSSL link.

Requirement 2, peer entity authentication, is satisfied by the use of X.509 digital cetificates.

Requirement 3 and 4 are satisfied by hashing algorithms: a hash is used to ensure message integrity, and a hash is signed with the private key of the signer, which can be verified with that person's public key.

3.3.2 Connection B

Requirement 1 (confidentiality) is addressed by encryption. Encryption is provided by the X9.23 standard (§2.13). The X9.23 standard specifies the means for using encryption to "scramble" all communications so that unauthorized parties are unable to interpret them. Only those parties who know the secret key are able to decipher the transmissions.

Requirement 2, peer entity authentication, is provided by X9.26 (§2.14), which provides sign-on authentication. Through sign-on authentication, we can determine whether or not the party requesting a connection is who they claim to be.

Requirement 3 and 4 are in the domain of message authentication (§2.6), which is provided by the X9.9 standard (§2.15) in this implementation. Through the use of X9.9 we can detect when received messages have been modified and also whether or not the received message is from an authorized source.

4 Results

Semcom satisfies the four fundamental requirements of a secure system:

Integrity of messages is provided by X9.9 (message authentication). The slightest modification of the message will be detected.

Confidentiality is provided by X9.23 (encryption). Messages can only be read by a party if it posesses the secret key.

Authentication: Peer entity authentication is implemented by X9.26, while message authentication is implemented by X9.9.

The code which satisfies these requirements was implemented on top of highly secure cryptographic algorithms (3DES), protocols (X9), and implementations (OpenSSL/libcrypto). Testing proved that these tools were used correctly and that the protocols and algorithms that had to be implemented produced the correct results.

In addition, the system satisfies most of SET's requirements:

Confidentiality of payment and ordering information: As was explained above, confidentiality is provided by encryption. Payment information is kept on the MSP's servers, and thus does not get transmitted over the network. Only the merchant's bank account number and the transaction amount is sent over the network. However, these are protected by encryption. Order information does not get transmitted over the network either, because the customer only sends the transaction amount to the MSP, which is protected by the security protocols and mechanisms implemented on link A.

Integrity of all transmitted data: The provision of message integrity was discussed above.

Authentication of cardholder: If a customer does not have a pre-arranged credit account with her MSP, the transaction would never be able to be initiated.

Authentication of merchant: As the merchant is not involved in the transaction, there is no need to verify this. If the merchant does not have an account at one of the banks supported by the system, then the transaction would naturally never be initiated. If this does happen for some reason, the bank server would simply reject the request, and the transaction would fail.

Best security practices and system design techniques: As mentioned above, Semcom makes use of well-known and proven algorithms and standards, the implementation of which has been tested and shown to be correct.

Protocol independent of transport security mechanisms: Because the X9 standards operate at the application layer, they do not interfere with anything at or lower than the network layer. Thus, Semcom is fully capable of co-existing with security mechanisms such as SSL/TLS or IPSec.

Semcom satisfies six out of the seven the requirements of SET, one of the most prominent e-commerce security standards. Not only does it satisfy them, but it even obsoletes some of them due to the simplicity of its architecture. This simplicity is a result of the fact that the merchant's involvement in the transaction is totally removed and that the MSP is trusted by the customer.

With Semcom's architecture, the customer authorizes the transaction through her mobile service provider (MSP), cutting out the middleman (the merchant). The customer is effectively doing the banking portion of the transaction herself, which is a much simpler solution than that provided by standards like SET. This is achieved by having the MSP handle the funds transfer on behalf of the customer.

This would require the MSP to have a prior arrangement with the merchant's bank. This greatly simplifies the security infrastructure needed to fulfill the system's security requirements, because it is assumed that the customer trusts her MSP. Complex mechanisms such as SET's dual signature are not necessary with such a simple architecture.

Link A is an end-to-end secure connection. The SSL record protocol ensures that data is encrypted, as each encrypted SSL record is fragmented and sent over TCP.

Link B is certainly an end-to-end secure connection. Packets sent over link B are encrypted at all times until they reach their destination. X9.23 specifies application-layer encryption, which means that IP routing headers are not encrypted. This allows the encrypted packet to be routed accross large networks without the need to be decrypted at intermediate nodes in order to get routing information. Thus, given that link A was known to be end-to-end secure, and considering that the transaction server is under the control of the trusted MSP, the system could be declared end-to-end secure.

Semcom's simplicity makes it easier and cheaper to implement and maintain, and also places no burden on the merchant, who has no involvement in the transaction besides verifying that the transaction amount has been deposited into his account.

Not only does Semcom's architecture fulfill the requirements of SET, but it also adds end-to-end security.

5 Conclusion

Semcom has been shown to be a secure m-commerce system which satisfies four of the fundamental requirements of secure computer systems as well as six out of the seven requirements of SET. However, Semcom's architecture is significantly simpler than that of SET due to the elimination of the merchant's involvement in the transaction, and the introduction of the MSP which is trusted by the customer. This makes a tremendous difference in the complexity of the required hardware and software infrastructure.

Semcom's simplicity makes it easier and cheaper to implement and maintain, and also places no burden on the merchant, who has no involvement in the transaction besides verifying that the transaction amount has been deposited into his account.

Perhaps the most important advantage of Semcom is that it provides end-to-end security.

References

[1] Federal Information Processing Standards (FIPS). (1999, October 25). *FIPS Publication 46-3: Data Encryption Standard*

[2] William Stallings (2003). *Network Security Essentials: Applications and Standards (Second edition)*
`ISBN: 0-13-120271-5`

[3] Tropical Software. *DES Encryption*
`http://www.tropsoft.com/strongenc/des.htm`

[4] Internet RFC/STD/FYI/BCP Archives. (2000, May). *RFC 2828 (RFC2828) - Internet Security Glossary*
`http://www.faqs.org/rfcs/rfc2828.html`

[5] X9, under procedures of ANSI (1986, August 15). *Financial Institution Message Authentication (Wholesale)*

[6] X9, under procedures of ANSI (1988, May 16). *Financial Institution Encryption of Wholesale Financial Messages*

[7] X9, under procedures of ANSI (1990, February 28). *Financial Institution Sign-On Authentication for Wholesale Financial Transactions*

[8] The X9 World Wide Web Site. *X9.org*
`http://www.x9.org/`

[9] Phone.com. 2000. *Understanding Security on the Wireless Internet.*
`http://www.phone.com/pub/Security_WP.pdf`

[10] Gemplus. *Mobile Commerce Security: Essential and Available*
`http://www.wmrc.com/businessbriefing/pdf/mcommerce2001/tech/Gemplus.pdf`

[11] Stephen Byrne. *Building Trusted and Secure m-Commerce Services–Innovative Security Technology to Enable and Secure Transactions*
`http://www.wmrc.com/businessbriefing/pdf/mcommerce2001/book/byrne.pdf`

[12] ANAM Wireless Internet Solutions. *M-Commerce Security Issues and Measures*
`http://www.ecai.ie/ecai-sec-mcom.pdf`

[13] Mark Osborne. *WAP, m-commerce and security*
`http://www.kpmg.co.uk/kpmg/uk/image/mcom5.pdf`

[14] American Bar Association: Section of Science and Technology: Information Security Committee.
Digital Signature Guidelines Tutorial
`http://www.s2.chalmers.se/iths/pdf/Digital%20signature%20tutorial.pdf`

[15] James Messham. *M-COMMERCE SECURITY*
`http://www.tdap.co.uk/uk/archive/billing/bill(fml_0012).html`

[16] Eric Olden. (2000). *Securing m-Commerce*
`http://eai.ebizq.net/web_integration/olden_1.html`