IMPLEMENTATION OF A SECURE MOBILE PAYMENT SYSTEM

by

Murat Acar

B.S. Electronics & Communication Engineering, Kocaeli University,1999

Submitted to the Institute for Graduate Studies in

Science and Engineering in partial fulfillment of

The requirements for the degree of

Master of Science

Graduate Program in System & Control Engineering

Boğaziçi University

2007

IMPLEMENTATION OF A SECURE MOBILE PAYMENT SYSTEM

APPROVED BY:

Graduate Program in System & Control Engineering

Boğaziçi University

2007

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Tamer Şıkoğlu for his invaluable guidance and help during the preparation of this dissertation. I would like to mention his patience, giving me inspiration and hope all the time when I needed.

I specially thank to my mother Suzan Acar, my father Ahmet Acar and my sister Nilüfer Tek, Serkan Tek and my grandmother Nazire Calkı for giving support of any kind in my life.

ABSTRACT

IMPLEMENTATION OF A SECURE MOBILE PAYMENT SYSTEM

Over the last decade, with the rapid development of internet and mobile technologies new concepts got into our lives. Several convenience which hasn’t been existed before became indispensable part of our lives. The widespread of ADSL and wireless Networks and with WiMAX technology, mobile internet and value added mobile services will usher a new era. Nowadays, although the convenience and advantages of e-trade is being used by so many people, the security risk of credit card transactions over internet keeps major community away from e-trade opportunities. Today’s mobile technology, spread of mobile devices and easier internet access alternatives, gives opportunity of mobile payment more secure and convenient.

In this Project, we realized a payment system over mobile platforms without using credit card and cash. In this mobile payment system in order to maintain security, a group of algorithms and security blocks are used.

ÖZET

GÜVENLİ BİR MOBİL ÖDEME SİSTEMİNİN GERÇEKLEMESİ

Geçtiğimiz on sene içerisinde internet ve mobil teknolojileri hayatımıza birçok yeni kavramı sokarken, öncesinde alternatifi olmayan birçok kolaylığı da beraberinde getirerek hayatımızın vazgeçilmez birer parçası olmuştur. ADSL ve kablosuz networklerin yaygınlaşması ve yakın bir gelecekte WiMAX ile şehir içi mobil internet ve üzerinde katmadeğerli servisler yine hayatımızda getireceği servislerle bir çığır açacaktır. Günümüzde e-ticaretin getirdiği bir çok kolaylık ve avantajlar birçok kişi tarafından benimsenmiş olsa da ödemelerin internet üzerinden kredi kartı ile yapılmasının ciddi güvenlik sorunlarına sebep olam ihtimali büyük çoğunluk e-ticareti hayatına sokmamayı tercih ediyor. Mobil ciharların bu kadar yaygınlaşması ve internet erişiminin de gün be gün daha kolay erişim imkanları, ödemelerinde cep telefonları üzerinden güvenli bir şekilde yapılmasını mümkün kılmaktadır.

Bu projede, mobil platformlar üzerinden, direk kredi kartı veya para kullanmadan, ödemelerin yapılmasını sağlayan bir sistem gerçekleştirdik. Mevcut ödeme yöntemlerinden daha güvenli bir platform olarak mobil ödeme sistemini inşa etmek üzere çeşitli güvenlik algoritmaları ve yapıları kullanılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

M_i	Message Input
F	Function
K_i	Constant
C	Crypted Output
S	8-bit lookup table
$\lll_s$	A left bit rotation by s places
⊟	Addition
RFID	Radio Frequency Identification
PDA	Personal Digital Assistant
POS	Point of Sale
SMS	Short Message Service
GSM	Global System for Mobile Communication
WAP	Wireless Application Protocol
AES	Advanced Encryption Standard
WLANs	Wireless Local Area Networks
AP	Access Point
LAN	Local Area Network

MD5	Message-Digest algorithm five
NIST	National Institute of Standards and Technology
SHA	Secure Hash Algorithm
NSA	National Security Agency
3DES	Triple Data Encryption Standard
VPN	Virtual Private Network
SSL	Secure Sockets Layer
TLS	Transport Layer Security
PKI	Public Key Infrastructure
RSA	An Algorithm for Public-Key Encryption
DSA	Digital Signature Algorithm
RC2/4	Rivest Cipher Algorithm
IDEA	International Data Encryption Algorithm
MAC	Message Authentication Code
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
NNTP	Network News Transfer Protocol
ARM	Advanced RISC Machines
OTP	One Time Password

1. INTRODUCTION

Exchange of banknotes, coins and checks are used to be the medium of payment but nowadays being replaced by smartcards, mobile terminals and internet. The slope of this movement going from the exchange of physical goods towards the exchanging of information between customer and supplier in a secure way. Payment with (credit) cards involves exchanging between customer's and the supplier's bank accounts by a third party regional or global card organization.

The e-commerce moved to this payment process further. All trade is done over internet and customer and supplier does the exchange without any physical contact. The recent development of high-speed mobile data networks has created a new channel for commerce, while more sophisticated mobile devices are enabling the virtual exchange of payment information known as proximity payments.

Change from physical exchange to virtual payment's advantages come up very vital benefits for customers whereas payment service providers has to be more aware of any stage of payment due to keep security level as high as possible and interoperatibility. The advent of mobile payments has added another layer of complexity through the use of constrained devices with different capabilities and network limitations.

Despite the differences, the success of mobile payments is contingent on the same factors that have fuelled the growth of physical world non-cash payments, namely: security, interoperability, privacy, global acceptance, and ease of use. In the meantime, high-speed data networks, such as 2.5 and 3G, with more sophisticated data-enabled wireless devices, have the potential to transform payment. Color screens, greater bandwidth, and more compelling content are converging to create an environment where consumers feel more comfortable transacting on the move. In addition, new wireless protocols, such as Bluetooth, infrared and radio frequency identification (RFID), are enabling short range wireless device-to-device payments. Although we are at an early stage in the development of mobile payments, a number of factors are threatening to arrest the development of this new medium, including the proliferation of competing network standards, as well as incompatible operating systems and devices. Another major factor is the lack of secure and interoperable standards for mobile payments [1].

In this Thesis, our aim is to design and implement a secure mobile payment system. In this academic study based on academic research and modeling of consumer, merchant and bank based system. We will study encryption algorithms, software languages security levels on mobile applications and other security techniques. We will evaluate the techniques, applications and implement the better approaches.

2. BACKGROUND

2.1. Mobile Commerce

Telecommunications, Internet, and mobile computing are merging their technologies to form a new business called mobile commerce. According to NetLingo, a dictionary of Internet terms, mobile commerce is a form of electronic commerce because transactions are generated from a product or service that exists electronically on the Internet. Known as the next-generation of e-commerce, m-commerce enables users to access the Internet without having to find a place to plug in [2].

2.2. Mobile Payment

2.2.1. Definition of mobile payment

Mobile payment can be defined as a payment that is carried out with a handheld device such as a mobile phone or a PDA (personal digital assistant). Payment involves a direct or indirect exchange of monetary values between parties. Handheld devices can be used at real POS (point of sale), in e-commerce and in m-commerce. According to the mobile payment forum, payment has evolved from the physical exchange of notes and coins, to writing checks, and to transferring payment card details either in person or at distance, over the phone or the Internet. This evolution has involved a shift from the physical transference of tangible tokens of value to an exchange of information between parties. The emergence of e-commerce has further digitized the payment process, where payment details are sent over open networks with no physical contact between the buyer and the seller. The recent development of high-speed mobile data networks has created a new channel for commerce, where more sophisticated mobile devices are enabling the virtual exchange of payment information. The advent of mobile payments has added another layer of complexity through the use of mobile devices with different network capabilities and limitations. [2]

2.2.2. Definitions of micro-payment and macro-payment

According to the worldwide web consortium, micro-payments are systems “capable of handling arbitrarily small amounts of money”. Another definition is given by Erin Joyce of atnewyork.com; she defines micro-payment as “an electronic transaction or payment in the range of about $10 to about one-tenth of a cent, which travels over the Internet or public network infrastructure”. Accordingly, in light of the last definition, macro-payment is an electronic transaction or payment superior to $10 [2].

2.2.3. Mobile global market

EMC, a leading researcher and publisher of intelligence about wireless markets, estimates that there are now more than one billion wireless phone subscribers worldwide. By 2005, 50 per cent of all calls in the world will be wireless. Today, mobile commerce looks in many ways much like electronic commerce did in 1995. For example, electronic commerce sales were evaluated to $ 200 million US in 1995 and mobile commerce sales reached $ 500 million US in 2003. Yet, in 2002 e-commerce sales had grown to $ 843 billion US and they are estimated to $ 6501 billion US in 2006 [2].

2.2.4. Supporting Technologies For Mobile Payment

Different supporting technologies for mobile payment are presented as follows:

2.2.4.1 SMS. Short Message Service (SMS) is a text message service that enables short messages of generally no more than 140-160 characters in length to be sent and transmitted from a mobile phone. SMS was introduced in the GSM (Global System for Mobile Communication) system and later supported by all other digital-based mobile communications systems. Unlike paging, but similar to e-mail, short messages are stored and forwarded at SMS centers, which means you can retrieve your messages later if you are not immediately available to receive them. SMS messages travel to the cell phone over the system's control channel, which is separate and apart from the voice channel [2]. Several telecommunication carriers have recently begun offering premium rate short messages, which through higher pricing and revenue sharing allow companies to be paid for their services by sending a short message. A short code is a five or six digit code instead of a 10-digit mobile phone number, to which a text message is sent. They are associated with a particular brand or campaign and are easier to memorize for consumers. They are useful for mobile commerce applications deployed with SMS because the consumer sends a key word to a specific short code to start a “mobile shopping” session [2].

2.2.4.2 WAP. Wireless Application Protocol (WAP) is the de facto worldwide standard for providing Internet communications and advanced telephony services on digital mobile phones, pagers, personal digital assistants and other wireless terminals. The WAP is a standard developed by the WAP Forum, a group founded by Nokia, Ericsson, Phone.com (formerly Unwired Planet), and Motorola. WAP defines a communication protocol as well as an application environment. In essence, it is a standardized technology for cross-platform distributed computing [2].

2.2.4.3 Barcodes and SMS WAP. When issuing a payment in mobile commerce applications, the consumer must receive a transaction confirmation. This confirmation is necessary to receive the goods and services that were purchased with the mobile phone. A way for merchants to validate transactions is to scan a barcode received on the consumer's mobile phone.

An implementation of this technology is to use a SMS WAP approach. This is a SMS containing a link to a WAP site. This link can be invisible and the WAP content is directly downloaded on the consumer's mobile phone. In this case, a SMS would be sent containing a link on a unique barcode. This barcode is automatically downloaded on the consumer's mobile phone and can be scanned with a normal barcode scanner used by the merchant. In the cases where the mobile phone and the scanner would be incompatible, the merchant enters the number below the barcode [2].

2.3. Mobile Security

Security is a complex, multifaceted challenge, which we at Symbol sometimes depict as six 'pillars' in a 'temple of security'. Each pillar represents a column that supports the system objective, which is Information Assurance. At the same time, each supporting column presents a hurdle or closed gate to an attacker, and we aim to strengthen each deterrent hurdle. We can think of the hurdles as a series of obstacles, each of which an attacker would have to overcome. We can identify no absolutes in these fields; only doors that tend to 'close off avenues' more or less well and force terrorists to go elsewhere. The requirements for the system include: Confidentiality, Integrity, Authentication, Non-Repudiation, Authorization/Access Control, and Availability [3].

2.3.1. Confidentiality

The first security pillar for the system, or hurdle for an attacker, is the most obvious. Confidentiality is the pillar that holds personal or restricted information that only a few, privileged persons may or should know. For many things, such as PIN numbers, we go to extra effort to conceal information to retain its confidentiality. We are well aware of the need to conceal information, from childhood with those imaginary ‘Captain Midnight’ decoder rings, to the infamous German Enigma Cipher of WWII, to the Advanced Encryption Standard (AES), four released this year by the United States National Institute of Standards and championed by Presidential Science Advisor, Jack Marburger [3].

2.3.2. Integrity

Concealed information, however, could still be corrupted. Hence the second pillar. A zero or two missing from your bank account number is a real problem. So, we must guarantee that the data hasn’t been tampered with, amended, altered or damaged in any way. It must remain consistent and completely intact for any applicable duration, possibly indefinitely, for the information to remain valid and accurate [3].

2.3.3. Authentication

At this point, we have protected against eavesdroppers and the data are intact, but we still don’t know who sent it, or, from the sender’s view, who actually received it. That was a fundamental problem of WEP, five the initial security mechanism designed for modern WLANs (Wireless Local Area Networks.) There was no mutual authentication between the client, such as a mobile terminal, and the network Access Point (AP). Today an industry has emerged around building network security mechanisms for wireless LANs that address all levels of intensity. This is where our tools have been most relevant and effective [3].

2.3.4. Non-repudiation

This pillar involves the participation requirements and protocols in any given transaction. Because of our reliance on electronic information exchange, we must provide authentication methods strong enough that the originator cannot deny participation in the transaction [3].

2.3.5. Authorization and access control

In security systems, varied rights and privilege(s) policies must also be administered, whether for an individual, a computer, or a piece of baggage authorized to fly aboard the aircraft. Remember that locking a front door to a house does very little good if a window or a back door is left open. This pillar suggests that we look deeper than the obvious keys and locking mechanisms. What is the protocol if a person appears at the door posing as a relative? Keys and locks are no longer useful if a resident is duped into letting in an otherwise unauthorized infiltrator. Suppose a burglar uses a house key stolen from your office desk? Access control must be set to strict criteria, not just superficial obstacles barring entrance [3].

2.3.6. Availability

However, a system that can be readily crippled is not really secure. A practical design must provide some degree of service even when confronted with massive denial-of-service attacks, or accidental failure. This is the sixth and final pillar, or final hurdle for an attacker. In addition, various cryptographic tools are the indispensable mortar reinforcing these pillars, from encryption algorithms, to message digests, to digital signatures [3].

2.4. Security Algorithms

2.4.1. MD5:

In cryptography, MD5 (Message-Digest algorithm five) is a widely used cryptographic hash_function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD5 hash is typically a 32-character hexadecimal number.

MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5; while it was not a clearly fatal weakness, cryptographers began to recommend using other algorithms, such as SHA-1.

2.4.1.1 MD5 Vulnerability. Because MD5 makes only one pass over the data, if two prefixes with the same hash can be constructed, a common suffix can be added to both to make the collision more reasonable. Because the current collision-finding techniques allow the preceding hash state to be specified arbitrarily, a collision can be found for any desired prefix; that is, for any given string of characters X, two colliding files can be determined which both begin with X. All that is required to generate two colliding files is a template file, with a 128-byte block of data aligned on a 64-byte boundary that can be changed freely by the collision-finding algorithm [4].

2.4.1.2 MD5 Applications. MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it. Unix-based operating systems include MD5 sum utilities in their distribution packages, whereas Windows users use third-party applications [4].

However, now that it is easy to generate MD5 collisions, it is possible for the person who creates the file to create a second file with the same checksum, so this technique cannot protect against some forms of malicious tampering. Also, in some cases the checksum cannot be trusted (for example, if it was obtained over the same channel as the downloaded file), in which case MD5 can only provide error-checking functionality: it will recognize a corrupt or incomplete download, which becomes more likely when downloading larger files[4].

MD5 is widely used to store passwords. A number of MD5 reverse lookup databases exist, which make it easy to decrypt password hashed with plain MD5. To prevent such attacks a salt can be added to passwords before hashing them. Also, it is a good idea to apply the hashing function (MD5 in this case) more than once see key strengthening. It increases the time needed to encode a password and discourages dictionary attacks.

2.4.1.3 Algorithm.

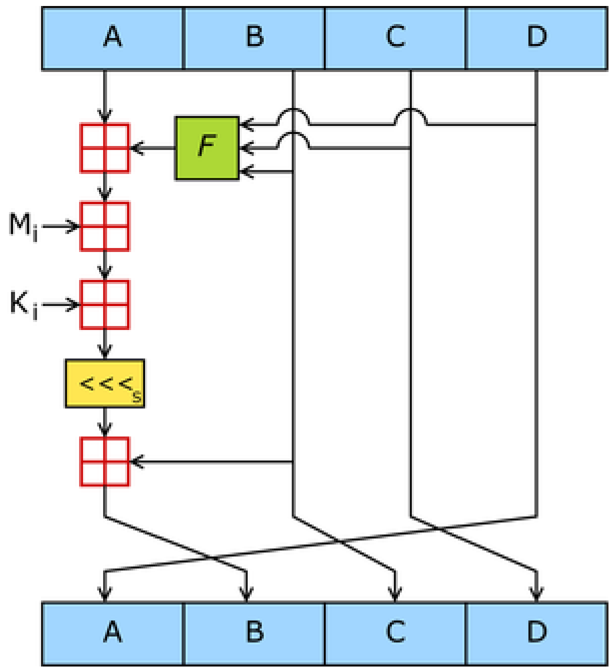


Figure 2.1. One MD5 operation

MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. *F* is a nonlinear function; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation.

$\lll_s$ denotes a left bit rotation by s places; s varies for each operation. $\boxplus$ denotes addition modulo 2^{32}[4].

2.4.2. SHA-1

The original specification of the algorithm was published in 1993 as the *Secure Hash Standard*, FIPS PUB 180, by US government standards agency NIST (National Institute of Standards and Technology). This version is now often referred to as "SHA-0". It was withdrawn by the NSA shortly after publication and was superseded by the revised version, published in 1995 in FIPS PUB 180-1 and commonly referred to as "SHA-1". SHA-1 differs from SHA-0 only by a single bitwise rotation in the message schedule of its compression function; this was done, according to the NSA, to correct a flaw in the original algorithm which reduced its cryptographic security. SHA-1 appears to provide greater resistance to attacks, supporting the NSA's assertion that the change increased the security.

SHA-1 (as well as SHA-0) produces a 160-bit digest from a message with a maximum length of $2^{64} - 1$ bits, and is based on principles similar to those used by Professor Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms [5].

2.4.2.1 Algorithm. The SHA (Secure Hash Algorithm) hash functions, namely SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512, are five related cryptographic hash functions designed by the National Security Agency (NSA) and collectively published as a US government standard. SHA-1 is employed in a large variety of popular security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec. It was considered to be the successor to MD5, an earlier, widely-used hash function. Both are reportedly compromised.

The other four variants (SHA-224, SHA-256, SHA-384, and SHA-512) are sometimes collectively referred to as SHA-2 functions or simply SHA-2. No attacks have yet been reported on the SHA-2 variants, but since they are similar to SHA-1, researchers are worried, and are developing candidates for a new, better hashing Standard [5].

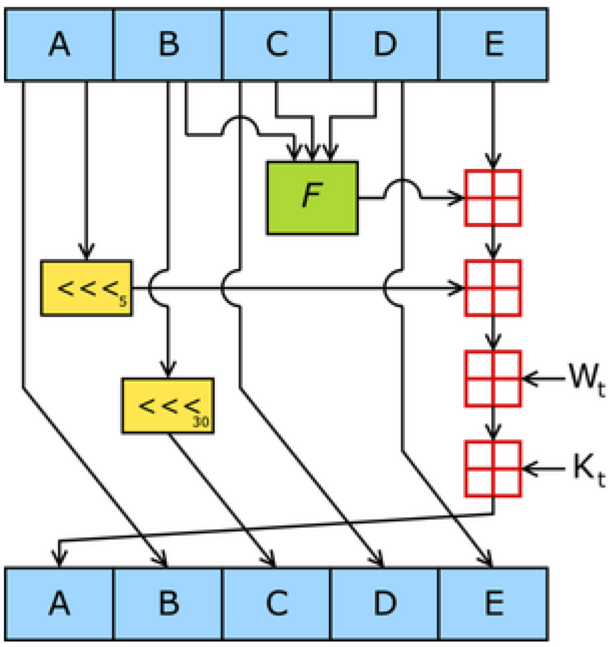


Figure 2.2. One iteration within the SHA-1 compression function.

A, B, C, D and E are 32-bit words of the state; *F* is a nonlinear function that varies; $\lll_n$ denotes a left bit rotation by *n* places; *n* varies for each operation. $\boxplus$ denotes addition modulo 2^{32}. K_t is a constant.

2.4.2.2 Cryptanalysis of SHA-1. In the light of the results on SHA-0, some experts suggested that plans for the use of SHA-1 in new cryptosystems should be reconsidered. After the CRYPTO 2004 results were published, NIST announced that they planned to phase out the use of SHA-1 by 2010 in favour of the SHA-2 variants [5].

NIST has published four additional hash functions in the SHA family, each with longer digests, collectively known as SHA-2. The individual variants are named after their digest lengths (in bits): "SHA-256", "SHA-384", and "SHA-512". They were first published in 2001 in the draft FIPS PUB 180-2, at which time review and comment were accepted. FIPS PUB 180-2, which also includes SHA-1, was released as an official standard in 2002. In February 2004, a change notice was published for FIPS PUB 180-2, specifying an additional variant, "SHA-224", defined to match the key length of two-key Triple DES.

SHA-256 and SHA-512 are novel hash functions computed with 32- and 64-bit words, respectively. They use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds. SHA-

224 and SHA-384 are simply truncated versions of the first two, computed with different initial values.

These new hash functions have not received as much scrutiny by the public cryptographic community as SHA-1 has, and so their cryptographic security is not yet as well-established. Gilbert and Handschuh (2003) have studied the newer variants and found no weaknesses [5].

2.4.2.3 SHA sizes.

Table 2.1 SHA Sizes

Algorithm	**Output size (bits)**	**Internal state size (bits)**	**Block size (bits)**	**Max message size (bits)**	**Word size (bits)**	**Passes**	**Operations**
SHA-0	160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rotl
SHA-1	160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rotl
SHA-256/224	256/224	256	512	$2^{64} - 1$	32	64	+,and,or,xor,shr,rotr
SHA512/384	512/384	512	1024	$2^{128} - 1$	64	80	+,and,or,xor,shr,rotr

2.4.2.4 Applications. A prime motivation for the publication of the Secure Hash Algorithm was the Digital Signature Standard, in which it is incorporated. The SHA hash functions have been used as the basis for the SHACAL block ciphers.

2.4.2.5 Example hashes. The following are some examples of SHA digests. ASCII encoding is assumed for all messages [5].

SHA1 ("The quick brown fox jumps over the lazy dog")

= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

Even a small change in the message will, with overwhelming probability, result in a completely different hash due to the avalanche effect. For example, changing `d` to `c`:

SHA1 ("The quick brown fox jumps over the lazy cog")

= de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3

The hash of the zero-length message is:

SHA1 ("") = da39a3ee 5e6b4b0d 3255bfef 95601890 afd80709

2.4.2.6 A description of SHA-1. Pseudo code for the SHA-1 algorithm follows [5]:

```
Note: All variables are unsigned 32 bits and wrap modulo 2^32 when calculating

Initialize variables:
h0 := 0x67452301
h1 := 0xEFCDAB89
h2 := 0x98BADCFE
h3 := 0x10325476
h4 := 0xC3D2E1F0

Pre-processing:
append the bit '1' to the message
append k bits '0', where k is the minimum number >= 0 such that the resulting message length (in bits) is congruent to 448 (mod 512)
append length of message (before pre-processing), in bits, as 64-bit big-endian integer

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w(i), 0 ≤ i ≤ 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w(i) := (w(i-3) xor w(i-8) xor w(i-14) xor w(i-16)) leftrotate 1
```

```
Initialize hash value for this chunk:

a := h0

b := h1

c := h2

d := h3

e := h4


Main loop:

for i from 0 to 79

    if 0 ≤ i ≤ 19 then

        f := (b and c) or ((not b) and d)

        k := 0x5A827999

    else if 20 ≤ i ≤ 39

        f := b xor c xor d

        k := 0x6ED9EBA1

    else if 40 ≤ i ≤ 59

        f := (b and c) or (b and d) or (c and d)

        k := 0x8F1BBCDC

    else if 60 ≤ i ≤ 79

        f := b xor c xor d

        k := 0xCA62C1D6
```

```
    temp := (a leftrotate 5) + f + e + k + w(i)

    e := d

    d := c

    c := b leftrotate 30

    b := a

    a := temp


Add this chunk's hash to result so far:

h0 := h0 + a

h1 := h1 + b
```

```
        h2 := h2 + c
        h3 := h3 + d
        h4 := h4 + e

    Produce the final hash value (big-endian):
    digest = hash = h0 append h1 append h2 append h3 append h4
```

Note: Instead of the formulation from the original FIPS PUB 180-1 shown, the following equivalent expressions may be substituted into the appropriate ranges of the above pseudocode for improved efficiency:

```
    (0  ≤ i ≤ 19): f := d xor (b and (c xor d))          (alternative)
    (20 ≤ i ≤ 39): f := b xor c xor d                    (unchanged)
    (40 ≤ i ≤ 59): f := (b and c) or (d and (b or c))    (alternative 1)
    (40 ≤ i ≤ 59): f := (b and c) + (d and (b xor c))    (alternative 2, due to Colin Plumb)
    (40 ≤ i ≤ 59): f := (b and c) or (d and (b xor c))   (alternative 3, due to Jeffrey Riaboy)
    (60 ≤ i ≤ 79): f := b xor c xor d                    (unchanged)
```

Figure 2.3. Pseudo code of SHA-1 Algorithm

2.4.3. 3DES (Triple DES)

In cryptography, **Triple DES** is a block cipher formed from the Data Encryption Standard (DES) cipher by using it three times.

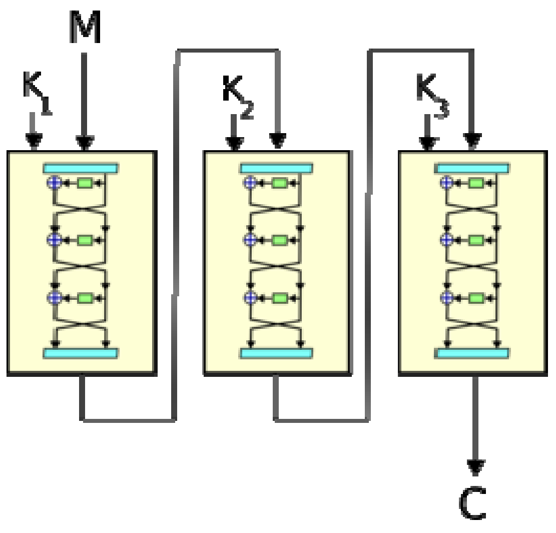


Figure 2.4. 168 bit 3DES with 64 bits block size, 48 DES-equivalent rounds [6]

2.4.3.1 Algorithm. When it was found that a 56-bit key of DES is not enough to guard against brute force attacks, TDES was chosen as a simple way to enlarge the key space without a need to switch to a new algorithm. The use of three steps is essential to prevent meet-in-the-middle attacks that are effective against double DES encryption. Note that DES is not a group; if it were one, the TDES construction would be equivalent to a single DES operation and no more secure.

The simplest variant of TDES operates as follows: DES(k_3;DES(k_2;DES(k_1;M))), where M is the message block to be encrypted and k_1, k_2, and k_3 are DES keys. This variant is commonly known as EEE because all three DES operations are encryptions. In order to simplify interoperability between DES and TDES the middle step is usually replaced with decryption (EDE mode): DES(k_3;DES^{-1}(k_2;DES(k_1;M))) and so a single DES encryption with key k can be represented as TDES-EDE with $k_1 = k_2 = k_3 = k$. The choice of decryption for the middle step does not affect the security of the algorithm [6].

In general TDES with three different keys (3TDES) has a key length of 168 bits: three 56-bit DES keys (with parity bits 3TDES has the total storage length of 192 bits), but due to the meet-in-the-middle attack the effective security it provides is only 112 bits. A variant, called two-key TDES (2TDES), uses $k_1 = k_3$, thus reducing the key size to 112 bits and the storage length to 128 bits. However, this mode is susceptible to certain chosen-plaintext or known-plaintext attacks and thus it is officially designated to have only 80-bits of security [6].

TDES is slowly disappearing from use, largely replaced by its natural successor, the Advanced Encryption Standard (AES). One large-scale exception is within the electronic payments industry, which still uses 2TDES extensively and continues to develop and promulgate standards based upon it (e.g. EMV, available from EMVCo). This guarantees that TDES will remain an active cryptographic standard well into the future [6].

By design, DES and therefore TDES suffer from slow performance in software; on modern processors, AES tends to be around six times faster. TDES is better suited to hardware implementations, and indeed where it is still used it tends to be with a hardware implementation (e.g., VPN appliances and the Nextel cellular and data network), but even there AES outperforms it. Finally, AES offers markedly higher security margins: a larger block size, potentially longer keys, and as of 2006, no known public cryptanalytic attacks [6].

2.4.4. SSL

Secure Sockets Layer (SSL), are cryptographic protocols which provide secure communications on the Internet for such things as web browsing, e-mail, Internet faxing, and other data transfers. There are slight differences between SSL 3.0 and TLS 1.0, but the protocol remains substantially the same. The term "TLS" as used here applies to both protocols unless clarified by context [7].

The SSL protocol allows client/server applications to communicate in a way designed to prevent eavesdropping, tampering, and message forgery. SSL provides endpoint authentication and communications privacy over the Internet using cryptography. Typically, only the server is authenticated (*i.e.*, its identity is ensured) while the client remains unauthenticated; this means that the end user (be that a person, or an application such as a web browser), can be sure of whom they are "talking" to. The next level of security - both ends of the "conversation" being sure of whom they are "talking" to - is known as mutual authentication. Mutual authentication requires public key infrastructure (PKI) deployment to clients [7].

SSL involves three basic phases:

1. Peer negotiation for algorithm support
2. Public key encryption -based key exchange and certificate-based authentication
3. Symmetric cipher -based traffic encryption

During the first phase, the client and server negotiation uses cryptographic algorithms. Current implementations support the following choices:

- for public-key cryptography: RSA, Diffie-Hellman, DSA or Fortezza;
- for symmetric ciphers: RC2, RC4, IDEA, DES, Triple DES, AES or Camellia;
- for one-way hash functions: MD2, MD4, MD5 or SHA.

<u>2.4.4.1 How it works.</u> The SSL protocol exchanges records; each record can be optionally compressed, encrypted and packed with a message authentication code (MAC). Each record has a *content_type* field that specifies which upper level protocol is being used [7].

When the connection starts, the record level encapsulates another protocol, the handshake protocol, which has *content_type* 22.

The client sends and receives several handshake structures [7]:

- It sends a *ClientHello* message specifying the list of cipher suites, compression methods and the highest protocol version it supports. It also sends random bytes which will be used later.

- Then it receives a *ServerHello*, in which the server chooses the connection parameters from the choices offered by the client earlier.

- When the connection parameters are known, the client and server exchange certificates (depending on the selected public key cipher). These certificates are currently X.509, but there is also a draft specifying the use of OpenPGP based certificates.

- The server can request a certificate from the client, so that the connection can be mutually authenticated.

- The client and server negotiate a common secret called the "master secret", possibly using the result of a Diffie-Hellman exchange, or simply encrypting a secret with a public key that is decrypted with the peer's private key. All other key data is derived from this "master secret" (and the client- and server-generated random values), which is passed through a carefully designed "pseudorandom function".

SSL have a variety of security measures [7]:

- Numbering all the records and using the sequence number in the MACs.
- Using a message digest enhanced with a key (so only with the key can you check the MAC). This is specified in RFC 2104.
- Protection against several known attacks (including man in the middle attacks), like those involving a downgrade of the protocol to a previous (less secure) version or a weaker cipher suite.
- The message that ends the handshake ("Finished") sends a hash of all the exchanged data seen by both parties.

- The pseudorandom function splits the input data in half and processes each one with a different hashing algorithm (MD5 and SHA), then XORs them together. This provides protection if one of these algorithms is found to be vulnerable.

2.4.4.2 Applications. SSL runs on layers beneath application protocols such as HTTP, FTP, SMTP and NNTP, and above the TCP or UDP transport protocol, which form part of the TCP/IP protocol suite. While it can add security to any protocol that uses reliable connections (such as TCP), it is most commonly used with HTTP to form HTTPS. HTTPS is used to secure World Wide Web pages for applications such as electronic commerce. SMTP is also an area in which TLS has been growing and is specified in RFC 3207. These applications use public key certificates to verify the identity of endpoints [7].

An increasing number of client and server products support SSL natively, but many still lack support. As an alternative, users may wish to use standalone SSL products like Stunnel. Wrappers such as Stunnel rely on being able to obtain a SSL connection immediately, by simply connecting to a separate port reserved for the purpose. For example, by default the TCP port for HTTPS is 443, to distinguish it from HTTP on port 80. However, in 1997 the Internet Engineering Task Force recommended that application protocols always start unsecured and instead offer a way to upgrade to SSL - which a pure wrapper like Stunnel cannot cope with [7].

SSL can also be used to tunnel an entire network stack to create a VPN, as is the case with OpenVPN.

SSL operates in modular fashion. It is extensible by design, with support for forward and backward compatibility and negotiation between peers [7].

2.4.5. AES

<u>2.4.5.1 Development.</u> Unlike AES' predecessor DES, Rijndael is a substitution permutation network, not a Feistel network. AES is fast in both software and hardware, is relatively easy to implement, and requires little memory. As a new encryption standard, it is currently being deployed on a large scale [8].

<u>2.4.5.2 Description of the cipher.</u> AES is not precisely Rijndael as Rijndael supports a larger range of block and key sizes; AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits, whereas Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits [8].

The key is expanded using Rijndael's key schedule.

Most of AES calculations are done in a special finite field.

AES operates on a 4×4 array of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). For encryption, each round of AES (except the last round) consists of four stages [8]:

1. `AddRoundKey`: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.
2. `SubBytes` a non-linear substitution step where each byte is replaced with another according to a lookup table.
3. `ShiftRows`: a transposition step where each row of the state is shifted cyclically a certain number of steps.
4. `MixColumns`: a mixing operation which operates on the columns of the state, combining the four bytes in each column using a linear transformation.

The final round replaces the `MixColumns` stage with another instance of `AddRoundKey`.

2.4.5.3 The `AddRoundKey` step.

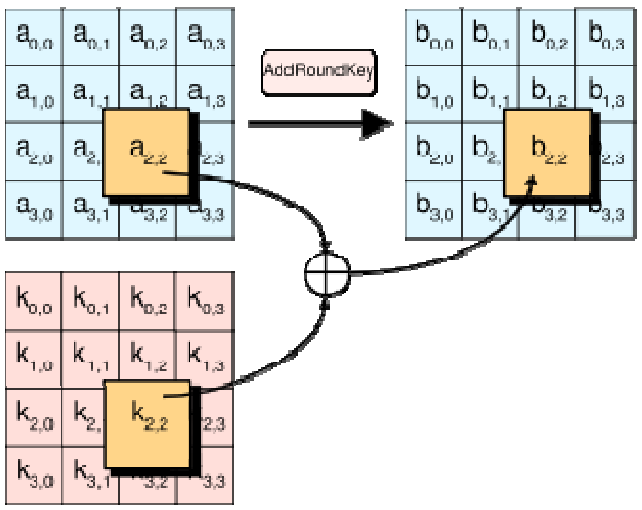


Figure 2.5. In the `AddRoundKey` step, each byte of the state is combined with a byte of the round subkey using the XOR operation.

In the `AddRoundKey` step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR [8].

2.4.5.4 The SubBytes step.

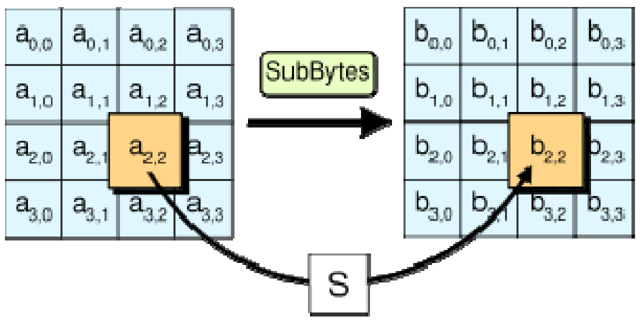


Figure 2.6. In the `SubBytes` step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, *S*; $b_{ij} = S(a_{ij})$.

In the `SubBytes` step, each byte in the array is updated using an 8-bit S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the inverse function over **GF**(2^8), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points [8].

2.4.5.5 The `ShiftRows` step.

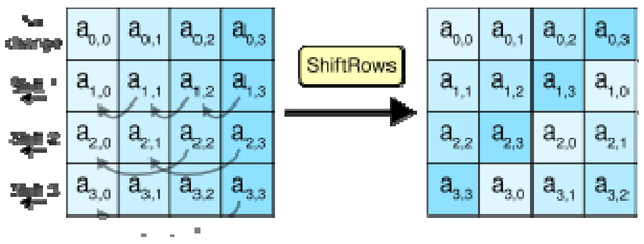
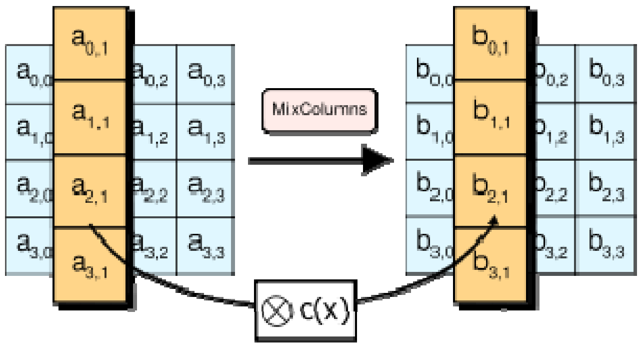


Figure 2.7. In the `ShiftRows` step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

The `ShiftRows` step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For the block of size 128 bits and 192 bits the shifting pattern is same. In this way, each column of the output state of the `ShiftRows` step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). In case of 256 bit block first row is unchanged and the shifting for second, third and fourth row is 1 byte, 2 byte and 4 byte respectively -

although this change only applies for the Rijndael cipher when used with a 256-bit block, which is not used for AES [8].

2.4.5.6 The `MixColumns` step.



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Figure 2.8. In the `MixColumns` step, each column of the state is multiplied with a fixed polynomial *c(x)*.

In the `MixColumns` step, the four bytes of each column of the state are combined using an invertible linear transformation. The `MixColumns` function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with `ShiftRows`, `MixColumns` provides diffusion in the cipher. Each column is treated as a polynomial over **GF**(2^8) and is then multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = 3x^3 + x^2 + x + 2$. The `MixColumns` step can also be viewed as a matrix multiply in Rijndael's finite field [8].

2.4.5.7 Optimization of the cipher. On systems with 32-bit or larger words, it is possible to speed up execution of this cipher by converting the `SubBytes`, `ShiftRows` and `MixColumns` transformations into tables. One then has four 256-entry 32-bit tables, which utilizes a total of four kilobytes (4096 bytes) of memory--a kilobyte for each table. A round can now be done with 16 table lookups and 12 32-bit exclusive-or operations, followed by four 32-bit exclusive-or operations in the `AddRoundKey` step [8].

If the resulting four kilobyte table size is too large for a given target platform, the table lookup operation can be performed with a single 256-entry 32-bit table by the use of circular rotates [8].

2.4.5.8 Security. As of 2006, the only successful attacks against AES have been side channel attacks. The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for US Government non-classified data. In June 2003, the US Government announced that AES may be used for classified information [8]:

"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use." [8]

This marks the first time that the public has had access to a cipher approved by NSA for TOP SECRET information. Many public products use 128-bit secret keys by default; it is possible that NSA suspects a fundamental weakness in keys this short, or they may simply prefer a safety margin for top secret documents (which may require security decades into the future).

The most common way to attack block ciphers is to try various attacks on versions of the cipher with a reduced number of rounds. AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. As of 2006, the best known attacks are on seven rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys.

Some cryptographers worry about the security of AES. They feel that the margin between the number of rounds specified in the cipher and the best known attacks is too small for comfort. The risk is that some way to improve these attacks might be found and that, if so, the cipher could be broken. In this meaning, a cryptographic "break" is anything faster than an exhaustive search, so an attack against 128-bit key AES requiring 'only' 2^{120} operations would be considered a break even though it would be, now, quite unfeasible. In practical application, any break of AES which is only this 'good' would be irrelevant. For the moment, such concerns can be ignored. The largest publicly-known brute force attack has been against a 64 bit RC5 key by distributed.net (finishing in 2002; Moore's Law implies that this is roughly equivalent to an attack on a 66-bit key as of December 2005).

Unlike most other block ciphers, AES has a very neat algebraic description.[4] This has not yet led to any attacks, but some researchers feel that basing a cipher on a new hardness assumption is risky. This has led Ferguson, Schroeppel, and Whiting to write, "...we are concerned about the use of Rijndael [AES] in security-critical applications."

In 2002, a theoretical attack, termed the "XSL attack", was announced by Nicolas Courtois and Josef Pieprzyk, showing a potential weakness in the AES algorithm. Several cryptography experts have found problems in the underlying mathematics of the proposed attack, suggesting that the authors may have made a mistake in their estimates. Whether this line of attack can be made to work against AES remains an open question. For the moment, the XSL attack against AES appears speculative; it is unlikely that anyone could carry out the current attack in practice [8].

2.4.5.9 Side channel attacks. Side channel attacks do not attack the underlying cipher, but attack implementations of the cipher on systems which inadvertently leak data.

In April 2005, D.J. Bernstein announced a cache timing attack that he used to break a custom server that used OpenSSL's AES encryption. The custom server was designed to give out as much timing information as possible, and the attack required over 200 million chosen plaintexts. Some say the attack is not practical over the internet with a distance of one or more hops; Bruce Schneier called the research a "nice timing attack." [8]

In October 2005, Adi Shamir and two other researchers presented a paper demonstrating several cache timing attacks (PDF file) against AES. One attack was able to obtain an entire AES key after only 800 writes, in 65 milliseconds. This attack requires the attacker to be able to run programs on the same system that is performing AES encryptions [8].

2.5. Development Platform

Mobile payment system is developed with C# in Microsoft .NET.

The system is run and tested with:

1. Microsoft Mobile Emulator Manager which is the mobile emulator solution where mobile payment system is run and tested.
2. Active Synch which is used as a solution for synchronizing data flow in between mobile device and PC.

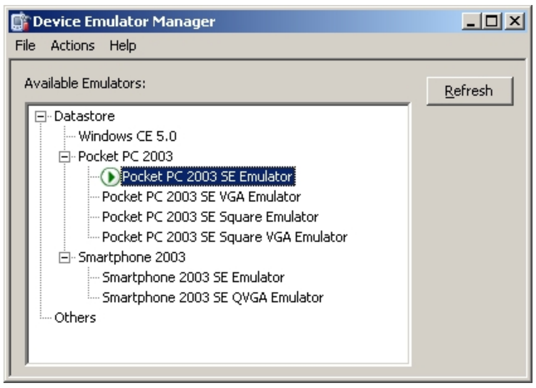

Device Emulator Manager
File Actions Help
Available Emulators:
Refresh
Datastore
Windows CE 5.0
Pocket PC 2003
Pocket PC 2003 SE Emulator
Pocket PC 2003 SE VGA Emulator
Pocket PC 2003 SE Square Emulator
Pocket PC 2003 SE Square VGA Emulator
Smartphone 2003
Smartphone 2003 SE Emulator
Smartphone 2003 SE QVGA Emulator
Others

Figure 2.9. Device Emulator Manager

The Microsoft Device Emulator (Figure 2.9) version 1.0 is a desktop application that emulates the behavior of a Windows CE- or Windows Mobile-based hardware platform. Using the Device Emulator, you can run, test, and debug a run-time image without the need for a physical device.

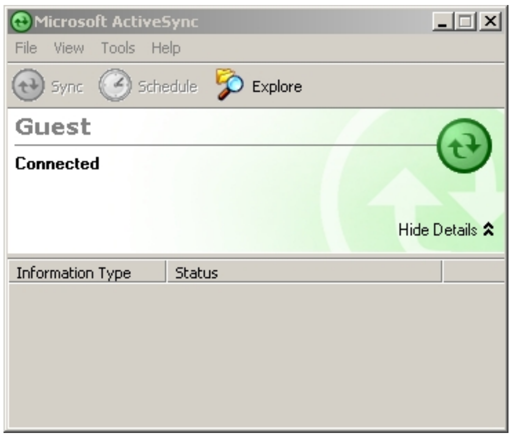


Figure 2.10. ActiveSync

ActiveSync (Figure 2.10) allows you to create a synchronization relationship between your mobile device and PC using a cable, cradle, Bluetooth, or infrared connection. ActiveSync can also make it possible for your device to connect to other resources through your PC.

2.5.1. Features of Mobile Device Emulator

Emulator runs code compiled for ARM processors rather than x86 processors. In most cases, you can run the same binaries on the emulator as you do on the device.

Mobile device emulator supports synchronizing with ActiveSync. You can use the emulator with a full ActiveSync partnership. With this feature you can debug applications that are synchronizing, or use real synchronized data from within the emulator.

Supports more development environments, including Visual Studio 2005, Visual Studio .NET 2003, and embedded Visual C++ 4.0 (Service Pack 4)—all using ActiveSync. Supports GAPI. You can write and debug games on this emulator. (Figure 2.11)

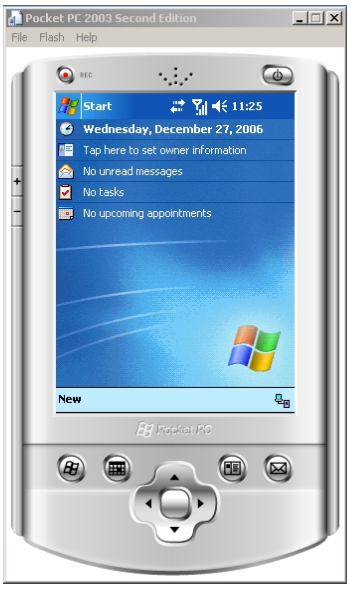

Pocket PC 2003 Second Edition
File Flash Help
Start
11:25
Wednesday, December 27, 2006
Tap here to set owner information
No unread messages
No tasks
No upcoming appointments
New

Figure 2.11. Pocket PC Emulator 2003

3.DESIGN AND IMPLEMENTATION OF A SECURE MOBILE PAYMENT SYSTEM

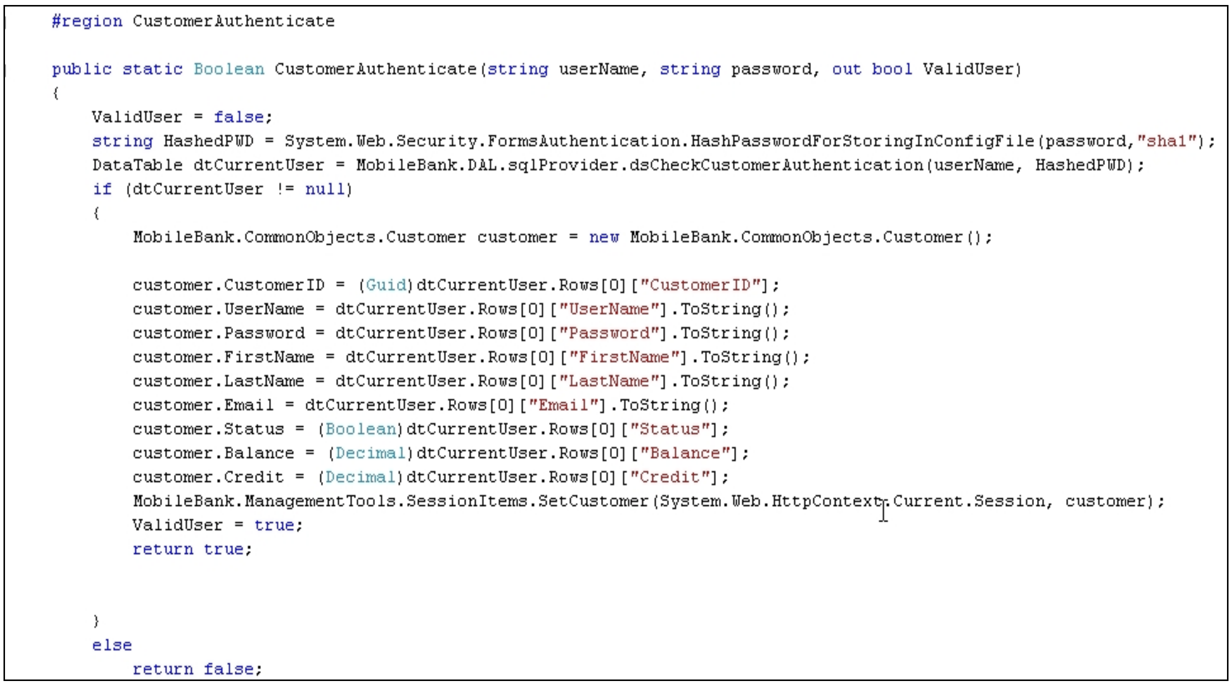
3.1. Security Features

3.1.1. SHA1

Authentication is done by username and password. User password is hashed with SHA1 algorithm and hashed outcome is stored in the database in order to secure password data in database. (Figure 3.1)

In cases, where application and databases located in different servers, hashing of password with SHA1 prevents account data to be hacked on any intrusion attempt.

In mobile bank application, we hashed password, also accounts username can be hashed and stored with SHA1.

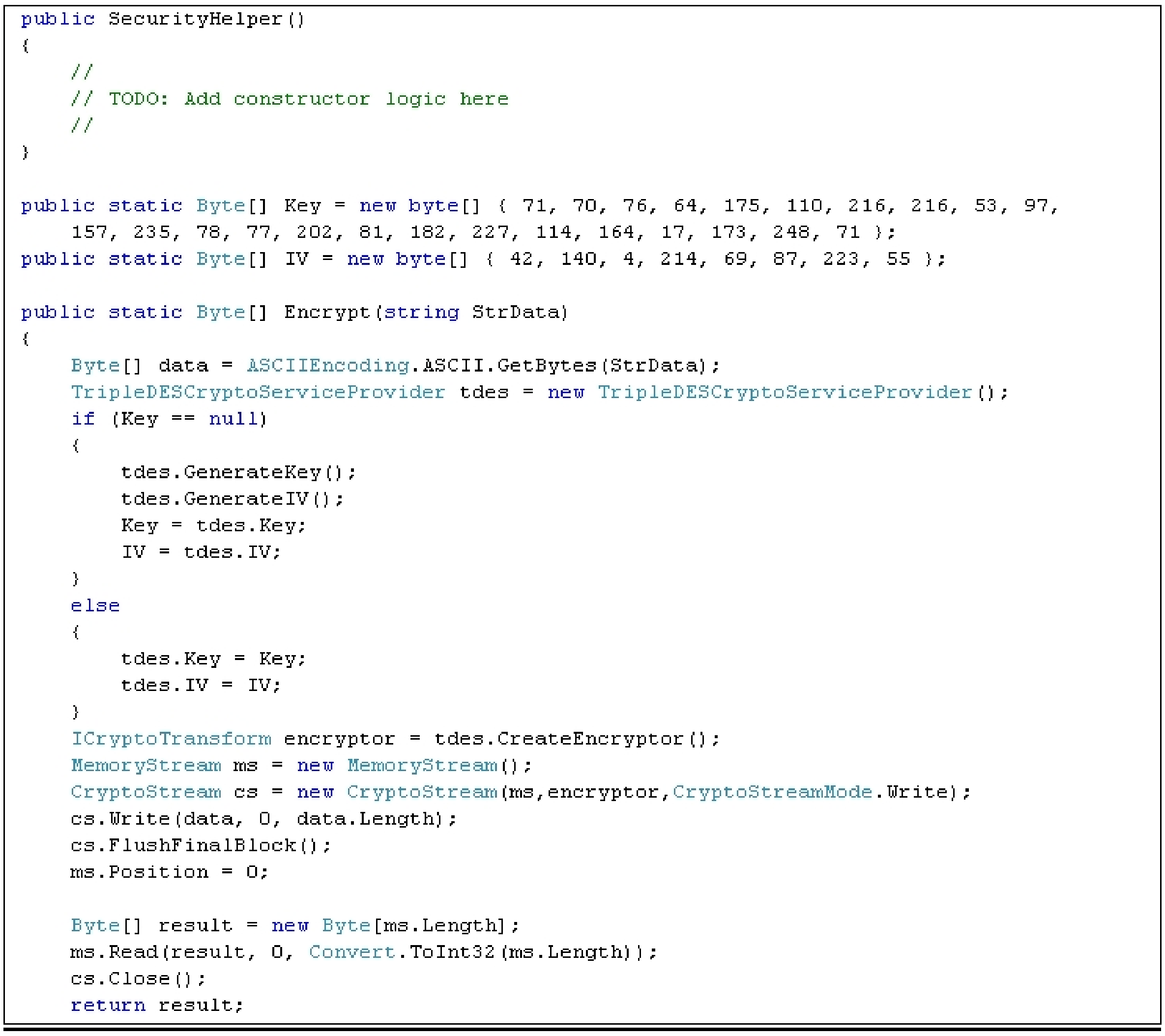


```
#region CustomerAuthenticate

public static Boolean CustomerAuthenticate(string userName, string password, out bool ValidUser)
{
    ValidUser = false;
    string HashedPWD = System.Web.Security.FormsAuthentication.HashPasswordForStoringInConfigFile(password,"sha1");
    DataTable dtCurrentUser = MobileBank.DAL.sqlProvider.dsCheckCustomerAuthentication(userName, HashedPWD);
    if (dtCurrentUser != null)
    {
        MobileBank.CommonObjects.Customer customer = new MobileBank.CommonObjects.Customer();

        customer.CustomerID = (Guid)dtCurrentUser.Rows[0]["CustomerID"];
        customer.UserName = dtCurrentUser.Rows[0]["UserName"].ToString();
        customer.Password = dtCurrentUser.Rows[0]["Password"].ToString();
        customer.FirstName = dtCurrentUser.Rows[0]["FirstName"].ToString();
        customer.LastName = dtCurrentUser.Rows[0]["LastName"].ToString();
        customer.Email = dtCurrentUser.Rows[0]["Email"].ToString();
        customer.Status = (Boolean)dtCurrentUser.Rows[0]["Status"];
        customer.Balance = (Decimal)dtCurrentUser.Rows[0]["Balance"];
        customer.Credit = (Decimal)dtCurrentUser.Rows[0]["Credit"];
        MobileBank.ManagementTools.SessionItems.SetCustomer(System.Web.HttpContext.Current.Session, customer);
        ValidUser = true;
        return true;


    }
    else
        return false;
```

Figure 3.1. Implementation of SHA-1 on Mobile Payment system.

3.1.2. 3DES

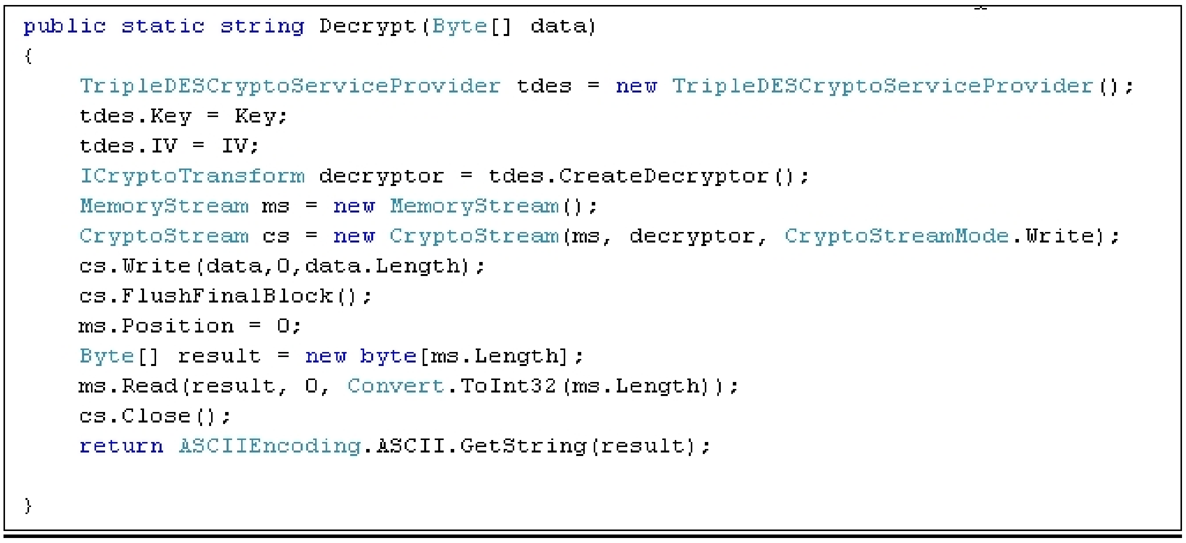
All data flow between mobile terminal, merchant and bank is being encrypted with 3DES algorithm in order to maintain a high level security for the data being exchanged between clients and the server. A 192 bit random number array is selected as 3DES key. (Figure 3.2)



```
public SecurityHelper()
{
    //
    // TODO: Add constructor logic here
    //
}

public static Byte[] Key = new byte[] { 71, 70, 76, 64, 175, 110, 216, 216, 53, 97,
    157, 235, 78, 77, 202, 81, 182, 227, 114, 164, 17, 173, 248, 71 };
public static Byte[] IV = new byte[] { 42, 140, 4, 214, 69, 87, 223, 55 };

public static Byte[] Encrypt(string StrData)
{
    Byte[] data = ASCIIEncoding.ASCII.GetBytes(StrData);
    TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
    if (Key == null)
    {
        tdes.GenerateKey();
        tdes.GenerateIV();
        Key = tdes.Key;
        IV = tdes.IV;
    }
    else
    {
        tdes.Key = Key;
        tdes.IV = IV;
    }
    ICryptoTransform encryptor = tdes.CreateEncryptor();
    MemoryStream ms = new MemoryStream();
    CryptoStream cs = new CryptoStream(ms,encryptor,CryptoStreamMode.Write);
    cs.Write(data, 0, data.Length);
    cs.FlushFinalBlock();
    ms.Position = 0;

    Byte[] result = new Byte[ms.Length];
    ms.Read(result, 0, Convert.ToInt32(ms.Length));
    cs.Close();
    return result;
```

Figure 3.2. Implementation of 3DES Encryption on Mobile Payment system.

Crypted data is being received from other party is decrypted as shown in Figure 3.3 below:



```
public static string Decrypt(Byte[] data)
{
    TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
    tdes.Key = Key;
    tdes.IV = IV;
    ICryptoTransform decryptor = tdes.CreateDecryptor();
    MemoryStream ms = new MemoryStream();
    CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Write);
    cs.Write(data,0,data.Length);
    cs.FlushFinalBlock();
    ms.Position = 0;
    Byte[] result = new byte[ms.Length];
    ms.Read(result, 0, Convert.ToInt32(ms.Length));
    cs.Close();
    return ASCIIEncoding.ASCII.GetString(result);

}
```

Figure 3.3. Implementation of 3DES Decryption on Mobile Payment system.

3.1.3. SSL

Mobile Payment System is 128 bit SSL certified.

3.1.4. One Time Password

The purpose of a one-time password (OTP) is to make it more difficult to gain unauthorized access to restricted resources, like a computer account. Traditionally static passwords can more easily be accessed by an unauthorized intruder given enough attempts and time. By constantly altering the password, as is done with a one-time password, this risk can be greatly reduced.

There are basically three types of one-time passwords: the first type uses a mathematical algorithm to generate a new password based on the previous, a second type that is based on time-synchronization between the authentication server and the client providing the password, and a third type that is again using a mathematical algorithm, but the new password is based on a challenge (e.g. a random number chosen by the authentication server or transaction details) and a counter instead of being based on the previous password.

Mobile Payment system generated OTP as taking a memory location from system memory block and gets time information of the system. Basically random alphanumeric characters are being selected and randomly queued as a byte array. This array maintains the timing interval for checking the system time. According to the seed integer data developed, .Net random function generates one time password.

3.2. Payment Processes

All stores serving mobile payment opportunity to their customers has a unique ShopID given from the mobile payment Bank.

The customer can buy a product/service with their mobile phone or PDA without paying cash and using their credit card in this mobile payment system (Figure 3.4)

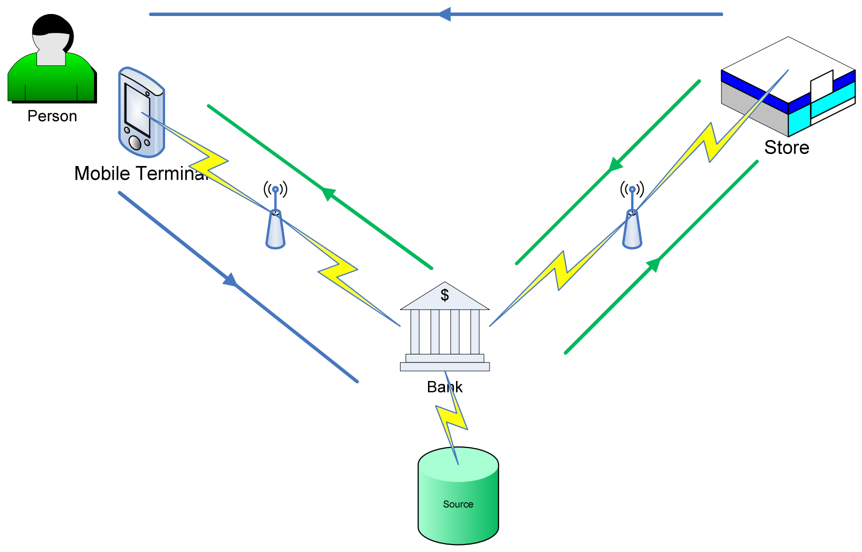


Figure 3.4. Mobile payment system

This mobile payment processes are explained below:

3.2.1. Authentication

User requests mobile payment password with mobile device (GSM, PDA, etc.). Mobile user enters UserID and sends a request to the server.

Bank sends a one time password to the GSM number of the customer who has the UserID sent within the password request.

One time password assures that authentication will be achieved with the person who has the mobile number associated with the concerned UserID. (Figure 3.5)

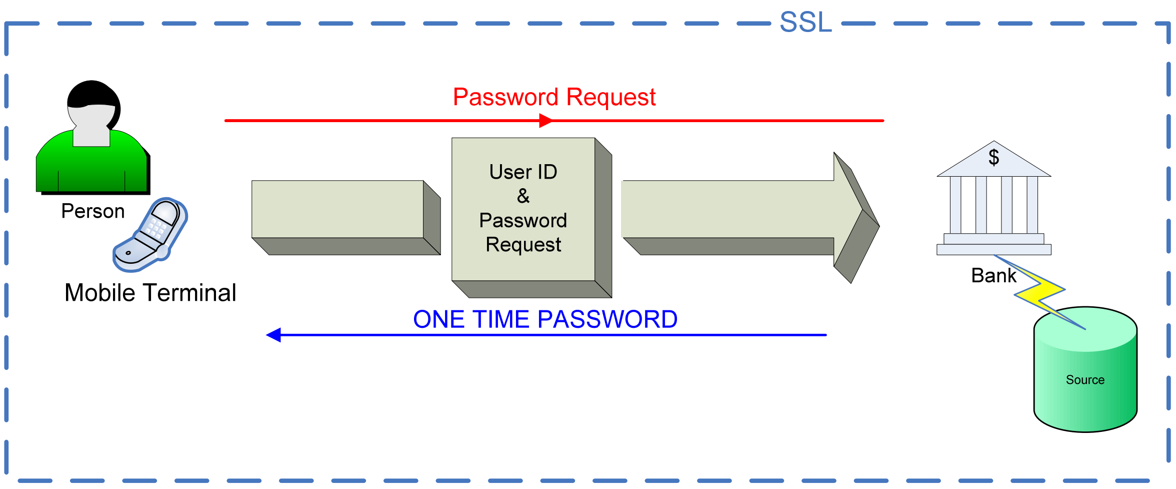


Figure 3.5. One-time password process in authentication

User browses the mobile payment address on mobile terminal. http://10.0.0.1/mobilbank/ShopOperation.aspx

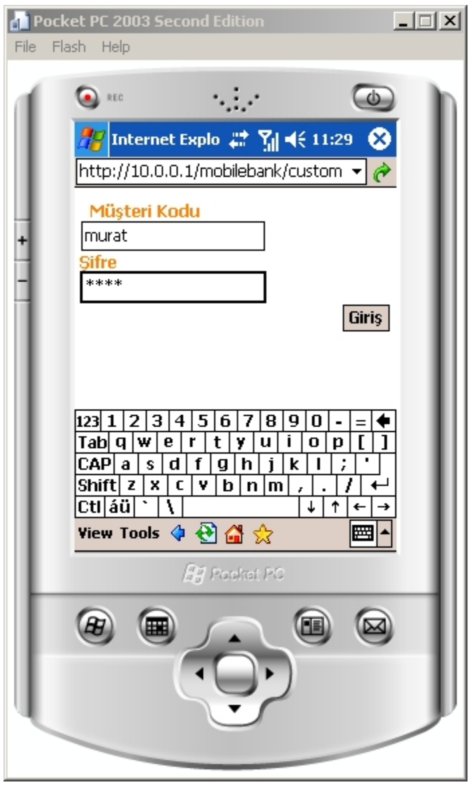


Figure 3.6. Login screen on PDA

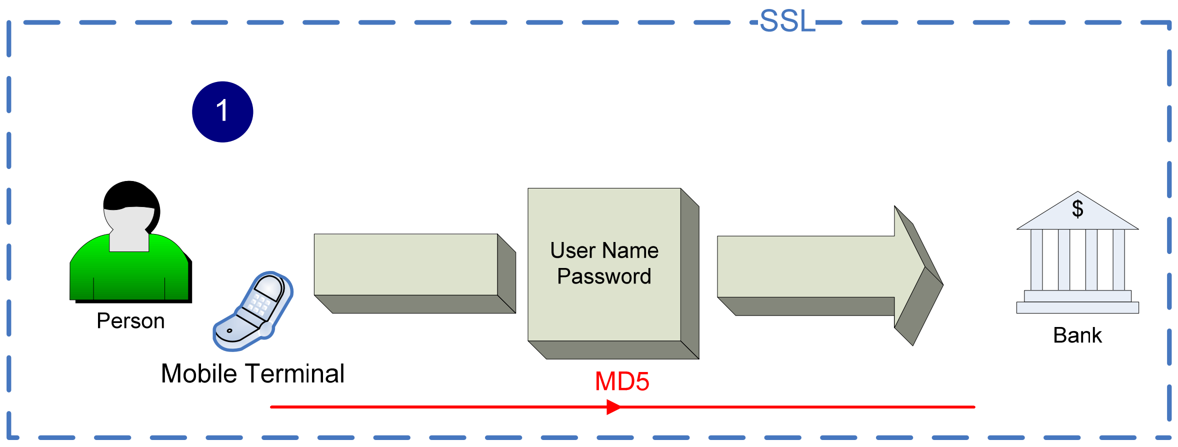
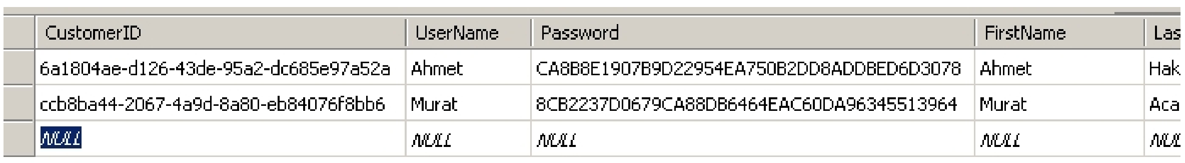


Figure 3.7. Authentication on mobile device.

Customer is asked to enter user ID and password. (Figure 3.6) The username and hashed password data transferred to bank. . (Figure 3.7)



CustomerID	UserName	Password	FirstName	Las
6a1804ae-d126-43de-95a2-dc685e97a52a	Ahmet	CA8B8E1907B9D22954EA750B2DD8ADDBED6D3078	Ahmet	Hak
ccb8ba44-2067-4a9d-8a80-eb84076f8bb6	Murat	8CB2237D0679CA88DB6464EAC60DA96345513964	Murat	Aca
NULL	NULL	NULL	NULL	NUL

Figure 3.8. Hashed password stored in database.

The user name and hashed password is verified with the database (Figure3.8) data and a session is created between mobile terminal and bank server. (Figure 3.9)

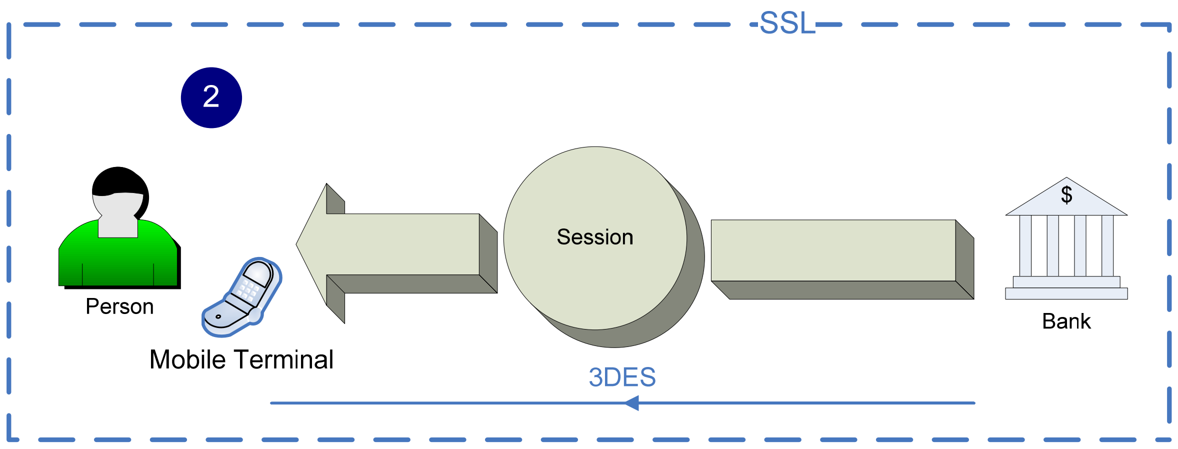


Figure 3.9.Session is created between bank and mobile terminal.

3.2.2. Ordering Request

Customer logs into the mobile payment interface, enters the ShopID and the amount of money for the payment. The mobile terminal passes this data to the bank. (Figure 3.10)

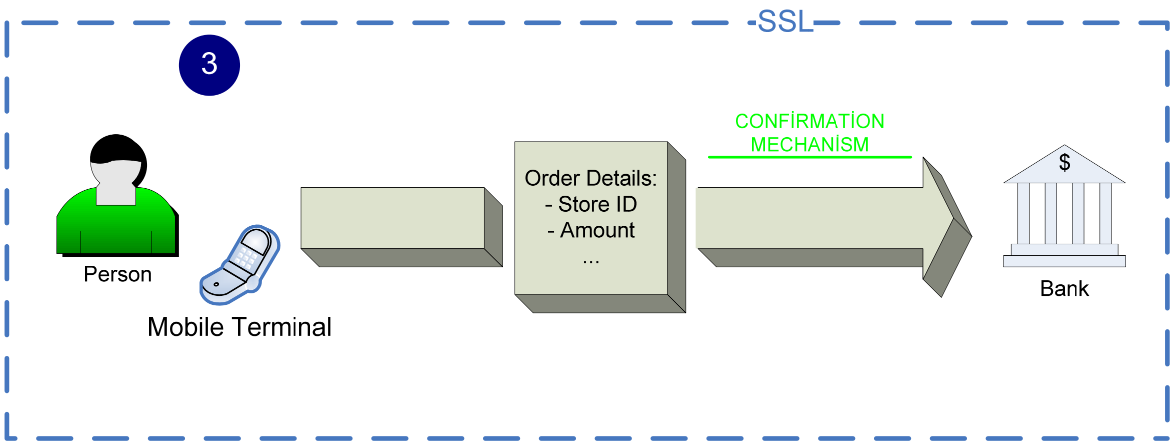


Figure 3.10. Ordering from mobile terminal.

3.2.3. Order Confirmation

Once the transaction accomplished, bank sends the confirmation message to the mobile terminal of customer. (Figure 3.11)

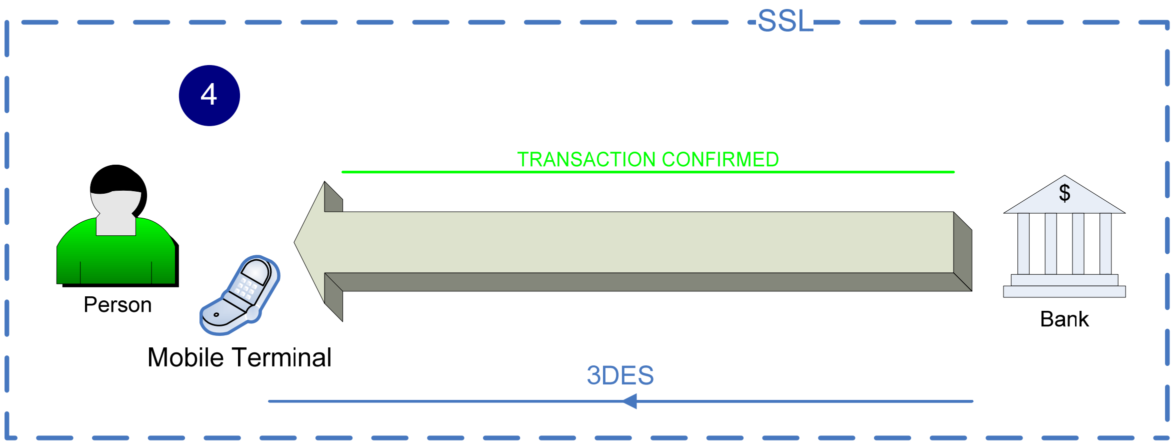


Figure 3.11. Transaction confirmation.

3.2.4. Shop Interface

The shop gets authenticated to the shop interface of mobile payment system with ShopID and password. This authentication is done with SHA1 hashing.

All transaction data concerning that shop is shown in real time in the shop terminal. Customer transaction data is confirmed by shop through the system and afterwards the purchased good/service is given to customer together with the bill (Figure 12).

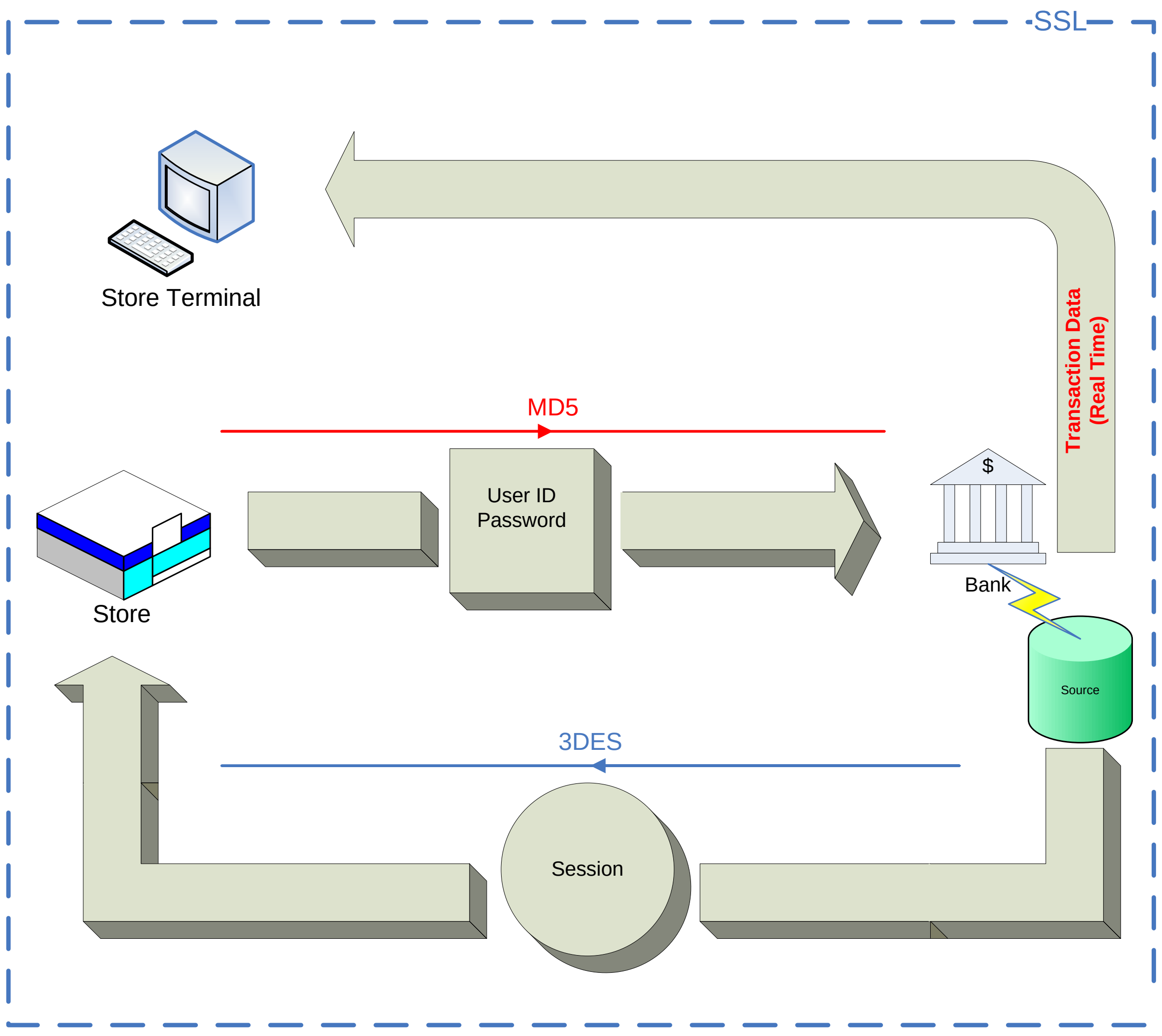


Figure 3.12. Shop mobile payment management process.

3.3. Application Fields:

The developed mobile payment system can be used in a variety of application fields.

3.3.1. Banks:

Banks can serve mobile line subscribers mobile payment opportunity. Account data of the customers can be matched and synchronized with their mobile phone numbers and mobile bank UserID. The shops which are in this system can serve customers without using credit card or cash.

In case a secure mobile payment system is assured by the banks, the customers will prefer using this payment medium rather than credit cards which have so fragile security system.

The mobile terminals (GSM phones, PDA's) are more likely to be electronic wallets in near future.

3.3.2. GSM Operators

GSM Operators can serve mobile payment, whether they match credit card info of the customers with their mobile payment account so that whenever the customer issues a transaction, the payment is done from his/her credit card. Operator can also issue the transaction to the mobile bill of the customer.

In Turkey, Turkcell is a good example for mobile payment service. The service is called "Mobil Ödeme". The customers can use SMS, WAP and voice message to but a service/product. Several organizations from different sectors that got into this mobile payment system are: Airways, fast-food restaurants, online shops, Turkcell billing, marine transport companies, natural gas providers, press distributor organizations.

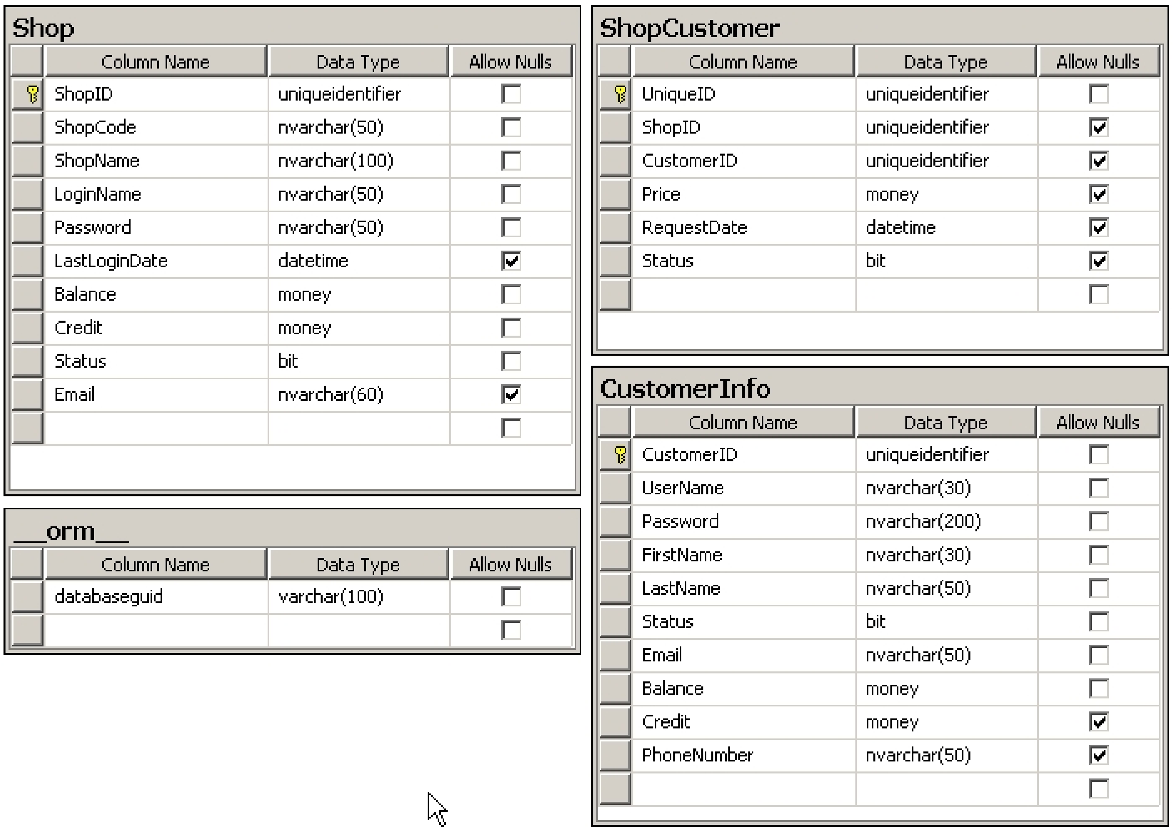
3.3.3. Stock Exchange Agents

Stock Exchange operations can be served by stock operation agent companies. The customers can give buy/sell orders from their mobile terminals.

The data flow in stock exchange is rather rapidly than other sectors data flow. Therefore, mobile medium for knowledge distributions and transactional order issuing over mobile medium is a vital solution for stock exchange operations.

3.4. Database Tables

We used Ms-SQL 2005 database management system. Figure 3.13 shows database tables of the developed system.



Shop

Column Name	Data Type	Allow Nulls
ShopID	uniqueidentifier	☐
ShopCode	nvarchar(50)	☐
ShopName	nvarchar(100)	☐
LoginName	nvarchar(50)	☐
Password	nvarchar(50)	☐
LastLoginDate	datetime	☑
Balance	money	☐
Credit	money	☐
Status	bit	☐
Email	nvarchar(60)	☑
		☐

__orm__

Column Name	Data Type	Allow Nulls
databaseguid	varchar(100)	☐
		☐

ShopCustomer

Column Name	Data Type	Allow Nulls
UniqueID	uniqueidentifier	☐
ShopID	uniqueidentifier	☑
CustomerID	uniqueidentifier	☑
Price	money	☑
RequestDate	datetime	☑
Status	bit	☑
		☐

CustomerInfo

Column Name	Data Type	Allow Nulls
CustomerID	uniqueidentifier	☐
UserName	nvarchar(30)	☐
Password	nvarchar(200)	☐
FirstName	nvarchar(30)	☐
LastName	nvarchar(50)	☐
Status	bit	☐
Email	nvarchar(50)	☐
Balance	money	☐
Credit	money	☑
PhoneNumber	nvarchar(50)	☑
		☐

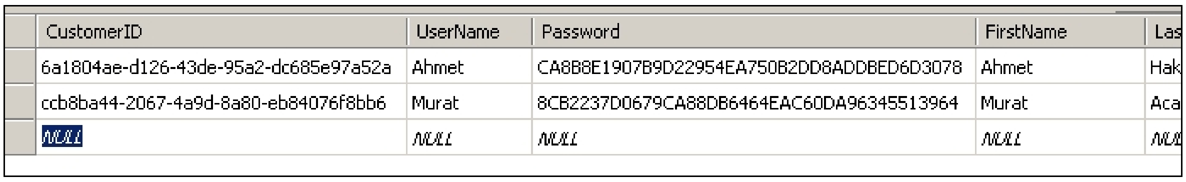
Figure 3.13 Database tables of Mobile Payment System

4.TESTS

4.1. Hashing Test

The password data is being stored in MS-SQL 2005. The SHA-1 hashing algorithm produces the “8CB2237D0679CA88DB6464EAC60DA96345513964” as hashed word for “12345”.

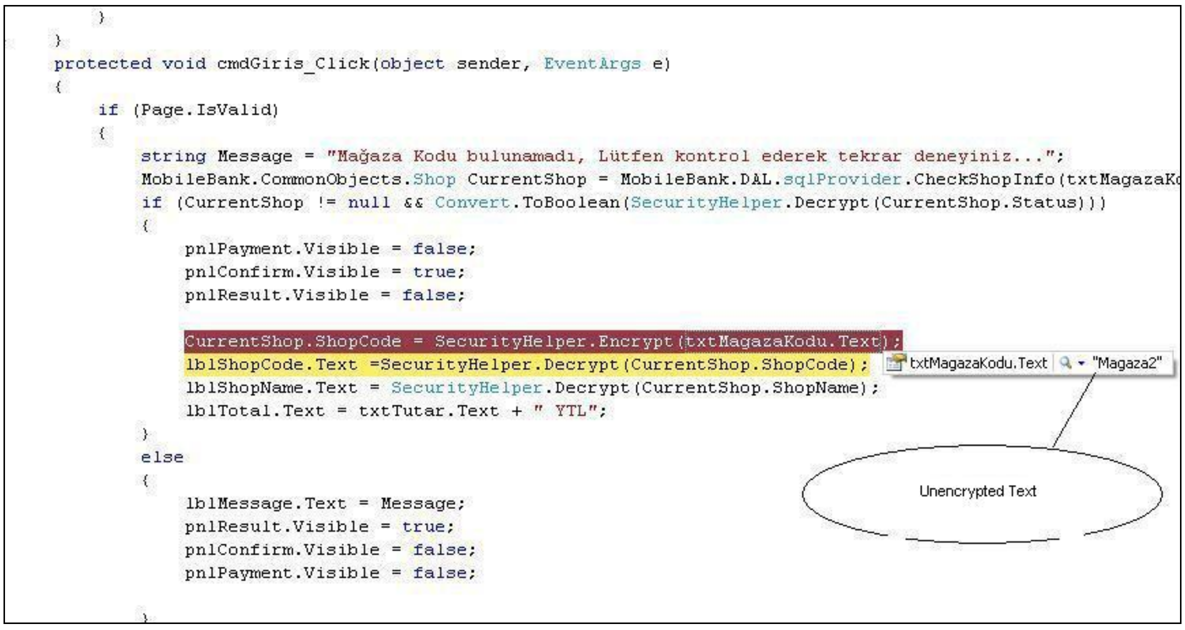
The hashed code in database is also the same. (Figure 4.1)



CustomerID	UserName	Password	FirstName	Las
6a1804ae-d126-43de-95a2-dc685e97a52a	Ahmet	CA8B8E1907B9D22954EA750B2DD8ADDBED6D3078	Ahmet	Hak
ccb8ba44-2067-4a9d-8a80-eb84076f8bb6	Murat	8CB2237D0679CA88DB6464EAC60DA96345513964	Murat	Aca
NULL	NULL	NULL	NULL	NUL

Figure 4.1. Password is stored hashed with SHA-1.

4.2. TriDES Test

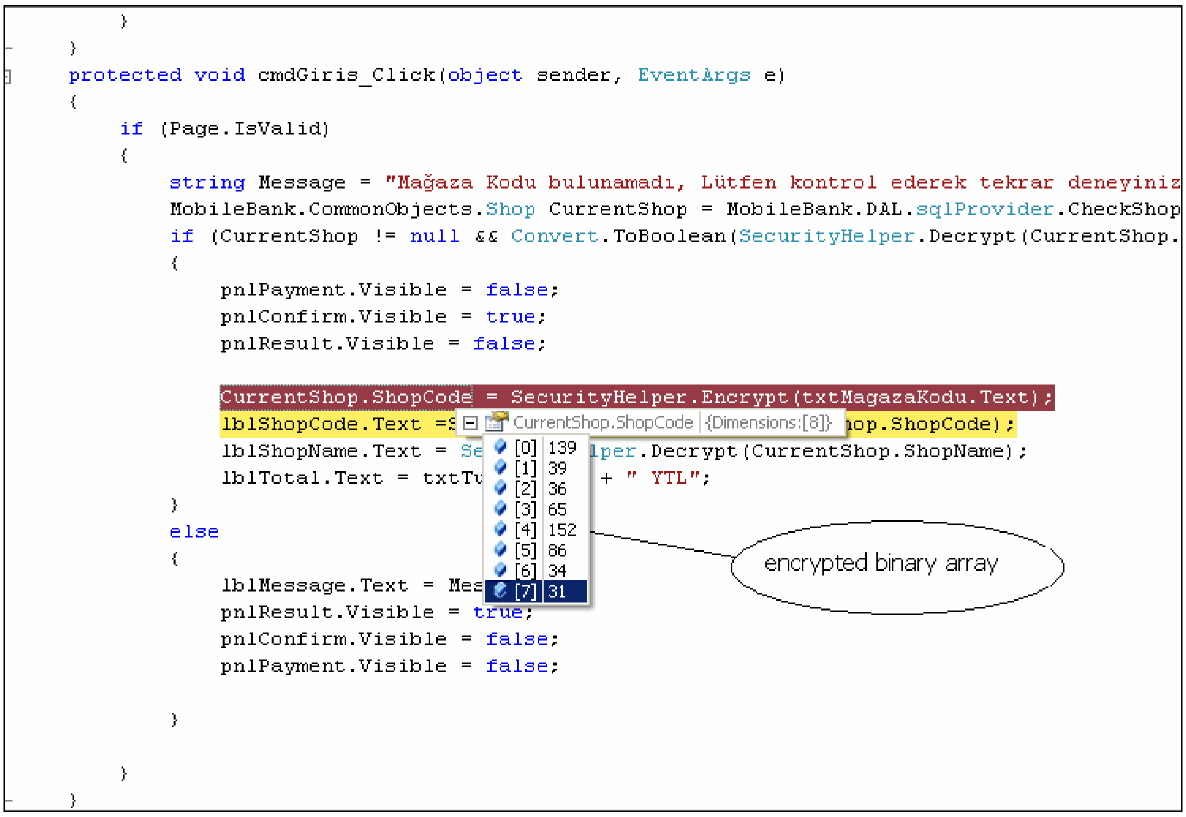


```
        }
    }
    protected void cmdGiris_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            string Message = "Mağaza Kodu bulunamadı, Lütfen kontrol ederek tekrar deneyiniz...";
            MobileBank.CommonObjects.Shop CurrentShop = MobileBank.DAL.sqlProvider.CheckShopInfo(txtMagazaKo
            if (CurrentShop != null && Convert.ToBoolean(SecurityHelper.Decrypt(CurrentShop.Status)))
            {
                pnlPayment.Visible = false;
                pnlConfirm.Visible = true;
                pnlResult.Visible = false;

                CurrentShop.ShopCode = SecurityHelper.Encrypt(txtMagazaKodu.Text);
                lblShopCode.Text =SecurityHelper.Decrypt(CurrentShop.ShopCode);   txtMagazaKodu.Text  "Magaza2"
                lblShopName.Text = SecurityHelper.Decrypt(CurrentShop.ShopName);
                lblTotal.Text = txtTutar.Text + " YTL";
            }
            else
            {
                lblMessage.Text = Message;
                pnlResult.Visible = true;
                pnlConfirm.Visible = false;
                pnlPayment.Visible = false;

            }
```

Figure 4.2. The Un-encrypted Text

In order to test 3des encryption functionality we tracked encryption and decryption results. txtMagazaKodu.Text is assigned “Magaza” as unencrypted. The

CurrentShop.ShopCode is the encrypted “txtMagazaKodu.Text” with 3DES and crypted word is shown in Figure 4.3.



```
        }
    }
    protected void cmdGiris_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            string Message = "Mağaza Kodu bulunamadı, Lütfen kontrol ederek tekrar deneyiniz
            MobileBank.CommonObjects.Shop CurrentShop = MobileBank.DAL.sqlProvider.CheckShop
            if (CurrentShop != null && Convert.ToBoolean(SecurityHelper.Decrypt(CurrentShop.
            {
                pnlPayment.Visible = false;
                pnlConfirm.Visible = true;
                pnlResult.Visible = false;

                CurrentShop.ShopCode = SecurityHelper.Encrypt(txtMagazaKodu.Text);
                lblShopCode.Text =S ⊟ CurrentShop.ShopCode {Dimensions:[8]} hop.ShopCode);
                lblShopName.Text = Se  [0] 139  lper.Decrypt(CurrentShop.ShopName);
                lblTotal.Text = txtTu  [1] 39    + " YTL";
            }                          [2] 36
            else                       [3] 65
            {                          [4] 152
                                       [5] 86
                lblMessage.Text = Mes  [6] 34
                pnlResult.Visible = true; [7] 31
                pnlConfirm.Visible = false;
                pnlPayment.Visible = false;

            }

        }
    }
```

Figure 4.3. The encrypted binary array of txtMagazaKodu.

4.3. One-Time Password Test

On customer authentication stage, several one time passwords requested as a stress test. All one time passwords created uniquely. Received new created one time passwords, didn’t cause any login problems.

5.CONCLUSIONS

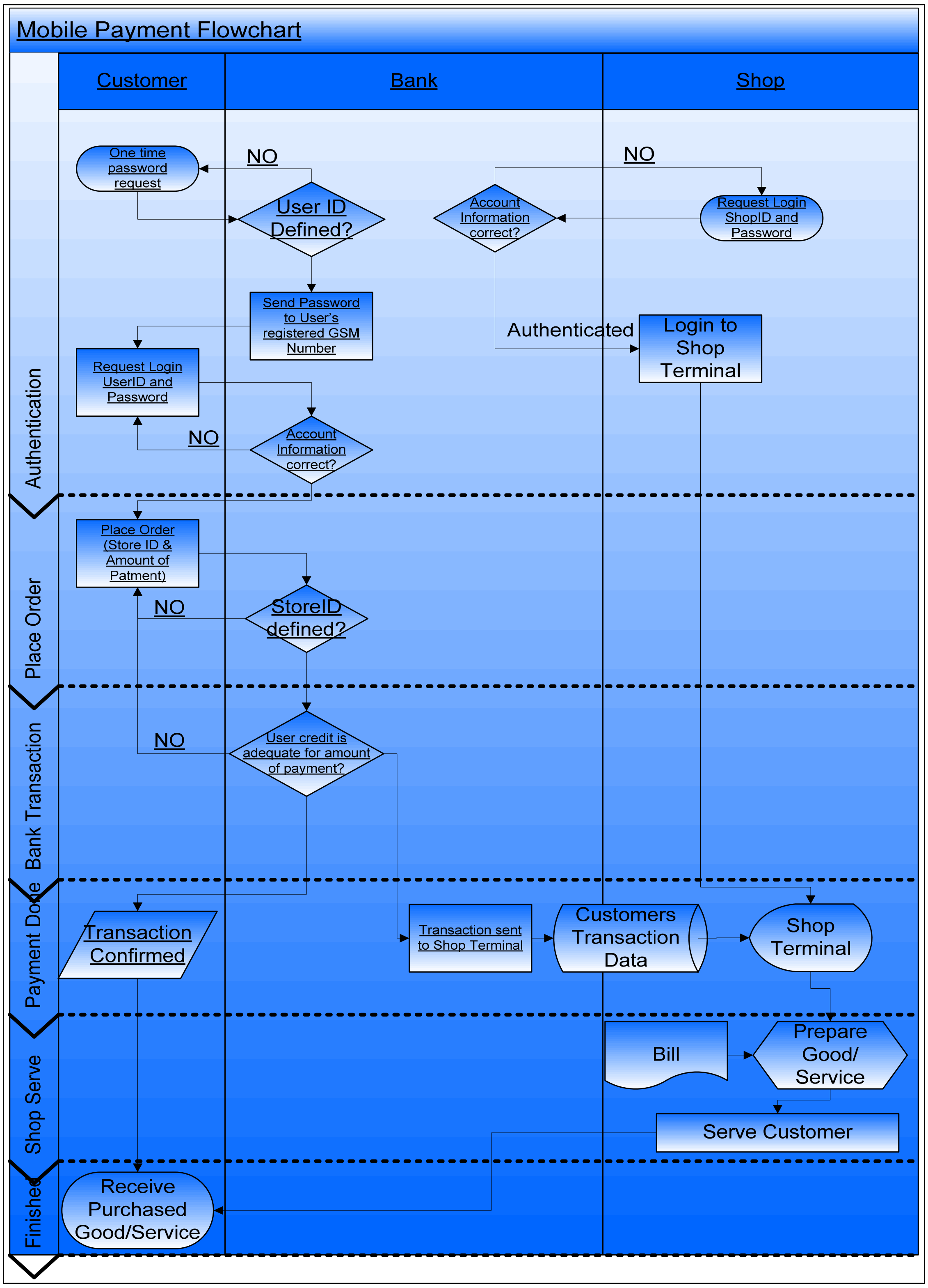
In this thesis, a secure mobile payment (MP) system is developed. Mobile payment system gives customers the opportunity of purchasing good and services from the shops without using credit card and cash. MP uses some security standards in order to serve secure transactions and mobile payment services.

The SHA-1 hashing, 3DES cryptography and one-time password over SSL methods are used in order to achieve secure system.

5.1. Future Work

- Hashing method can be upgraded to SHA-512
- AES can be implemented to ensure more secure cryptography
- Personalized graphical selection indicators can be implemented during the authentication process.

6.APPENDIX A: FLOWCHART


Mobile Payment Flowchart
Customer
Bank
Shop
Authentication
Place Order
Bank Transaction
Payment Done
Shop Serve
Finished
One time password request
NO
User ID Defined?
Send Password to User's registered GSM Number
Request Login UserID and Password
NO
Account Information correct?
NO
Account Information correct?
Request Login ShopID and Password
Authenticated
Login to Shop Terminal
Place Order (Store ID & Amount of Patment)
NO
StoreID defined?
NO
User credit is adequate for amount of payment?
Transaction Confirmed
Transaction sent to Shop Terminal
Customers Transaction Data
Shop Terminal
Bill
Prepare Good/ Service
Serve Customer
Receive Purchased Good/Service

7. REFERENCES

1. Mobile Payment Forum White Paper, "Enabling Secure, Interoperable, and User-friendly Mobile Payments ", December 2002,http://www.mobilepaymentforum.org/

2. Emilie Valcourt, Jean-Marc Robert and Francis Beaulieu, "Investigating mobile payment: supporting technologies, methods, and use", 2005, IEEE

3. Jerome Swartz, "Security systems for a mobile world", Symbol Technologies, Inc., One Symbol Plaza, Holtsville, NY 11742, USA

4. http://en.wikipedia.org/wiki/Md5

5. http://en.wikipedia.org/wiki/SHA-1

6. http://en.wikipedia.org/wiki/3DES

7. http://en.wikipedia.org/wiki/SSL

8. http://en.wikipedia.org/wiki/One_time_password

9. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard