DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

JAVA CARD BASED SECURITY APPLICATION MODULE

by
Soner SEZGİN

February, 2006
İZMİR

JAVA CARD BASED SECURITY APPLICATION MODULE

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in
Computer Engineering, Computer Engineering Orientation Program

by
Soner SEZGİN

February, 2006
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"JAVA CARD BASED SECURITY APPLICATION MODULE"** completed by **Soner SEZGİN** under supervision of **Assoc. Prof. Dr. Yalçın ÇEBİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

....

Doç. Dr. Yalçın Çebi

Supervisor

....

Prof. Dr. Alp R. KUT

(Jury Member)

....

Yrd. Doç. Dr. Zafer Dicle

(Jury Member)

Prof.Dr. Cahit HELVACI

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank to my supervisor, Assoc. Prof. Dr. Yalçın ÇEBİ who has reinforced me to complete the thesis. Even in the most badly times that I used to think about giving up, he encouraged and motivated me, so that I could finish.

I also would like to thank to my colleagues, Burak AYDIN, Sezai ÇAM and other Kentkart employees for their help and motivation.

Finally, I give my deepest gratitude to my family Sevgi - Nurettin SEZGİN and Nursen BİLGİÇ for their endless support and encouragement.

Soner SEZGİN

JAVA CARD BASED SECURITY APPLICATION MODULE

ABSTRACT

The amount of data we have is getting higher along with the growing technology. Internet and other digital media make it possible to share data faster and easier, as a manner of this, it is needed to protect some data. There are wide variety of software on the market specialized on data protection, but it is proven by research that the most powerful technique is using an external hardware module.

The smart card is one of the most feasible hardware which can be used as such module. Java cards would be the best choice, because of the ability of application development in it and also it is possible to get engineering samples with small amounts.

This study gives information about infrastructure of such a personal security application module.

Keywords: Smart card, java card, security application module

JAVA CARD TABANLI GÜVENLİK UYGULAMA MODÜLÜ

ÖZ

Sahip olduğumuz bilgi miktarı gelişen teknolojiye bağlı olarak hızla artmaktadır. Internet ve diğer sayısal ortamlar bilgi paylaşımını hızlandırmakta ve kolaylaştırmakta, buna bağlı olarak bazı bilgilerin korunması ihtiyacı ortaya çıkmaktadır. Bu ihtiyaca yönelik olarak piyasada çok fazla güvenlik yazılımı bulunmaktadır ancak araştırmalar göstermiştir ki en güvenli teknikler harici bir donanım tarafından desteklenen tekniklerdir.

Böyle bir donanım için kullanılabilecek seçeneklerin başında akıllı kartlar gelmektedir. Üzerinde uygulama geliştirilebilmesine olanak sağlayan, kolaylıkla ve az miktarlada temin edilebilen java cardlar donanımsal güvenlik modülü gerçeklemesi için en uygun üründür.

Bu çalışma, ucuz ve kolay kullanılabilir kişisel bir güvenlik uygulama modülü geliştirilmesi için altyapı sağlamak amacıyla yapılmıştır.

Anahtar Sözcükler: Akıllı kart, java card, güvenlik uygulama modülü

CONTENTS

CHAPTER ONE
INTRODUCTION

1.1 Understanding SAM

A Secure Application Module (SAM) is a secure device used to perform transactions (Gemplus, 1998). It can be used in:

- Payment applications (between th terminal and the host)
- Other applications such as loyality, identity, healtcare, driver license, logical and pysical access control, etc.

The SAM is a secure module which can be integrated in a terminal to authenticate and/or validate user information which can be carried in any form of card or any other media. Sensitive data such as keys and secret codes are stored in SAM. The SAM performs all mirror functions of the user card and/or the terminal.

The advantages of this are

- Sensitive operations are performed by the SAM rather than the terminal which in many cases cannot be considered as trustworthy.
- Cryptography no longer is needed to be implemented by system developer. Because of it is taken care of by the SAM

The SAM has functionalities like:

- Holds secure keys and algorithms (and their updates)
- Verifies cards and products
- Holds acceptance and capability criteria
- Signs modified products and transactions
- Sensitive command control
- MAC computation for batch signature
- SAM personalization commands

SAM can be in different formats. Most common types are ISO7816 format for bi-slot terminals or in a plug-in sim card format. This type of SAMs are called terminal SAM and used in POS terminals. Their functionality and speed are limited but sufficient for required operations. Some of these operations are checking existing certification codes generated for previous transaction and generating new certification codes for current transactions. The new generated certification codes are kept in terminal and when the number of certification codes reaches predefined limit they are transferred to the host computer.

A system is a combination of hundreds of terminals and single (in some system more then one) host computer. Some faster and efficient SAMs are required for batch applications in Host computers. The certification codes collected from terminals are controlled in host computer to check if the code is generated by SAM which is member of same system. Merchant may try to generate fake certification code to get some extra profit. The host batch process prevents this type of tampering. Hardware supported fast encrypt/decrypt engines are used in this type of SAM. Host SAMs are usually tamper proof black boxes which has PCI interface for high speed data process.

Basic relationship between the different types of user cards and SAM is shown in the figures given below.

USER CARD	TERMINAL APPLICATION	SAM
Certificate generation =>	<= Request for certificate Request for certificate check =>	Certificate check ? -OK | return TRUE - Not OK | return FALSE + reset ⇐

Figure 1.1 - Sample SAM process for smart card

USER CARD	TERMINAL APPLICATION	SAM
	<= Read certificate	
Certificate =>		
	Request for certificate check =>	
		Certificate check ? -OK | return TRUE - Not OK | return FALSE + reset ⇐
	Request for new certificate =>	
		<= Certificate generation
	<= Write new certificate	
New certificate <=		

Figure 1.2 - Sample SAM process for dummy memory card

Security Application Module is used to perform transactions. Electronic purse scheme is selected as an example to handle what kinds of transactions are performed and how they are performed in SAM. Security level applied to the transaction is related with the importance of the data. Electronic purse applications have money related transactions and it has the highest security level in every step. Different types of SAMs are used for each step of electronic purse scheme. The types of SAMs are explained in section 4 in detail. SAMs are named according to their physical form and where they are used. SAM for HOST (HOST SAM), SAM for personalization and SAM for terminal are some examples.

1.2 Objectives

There are a wide variety of security mechanisms available to the designer. The most important mechanisms are based on the use of cryptographic algorithms. By this means it is possible to create services for

- Authentication
- Data Integrity
- Confidentiality
- Non – repudiation

In some cases a single mechanism can provide a number of security services. For instance a digital signature can offer data integrity with source authentication and non - repudiation.

The important point is that all these cryptographic controls involve key management. This requires the secure distribution of cryptographic keys into the various entities and the need for these entities to provide a tamper resistant environment.

Personal usage of this type of environment is not easy to achieve. Even it is possible, key management and interfacing may be cruel problem. This project gives information about how to design a personal security application module with open platform java card.

CHAPTER TWO
INTRODUCTION TO SMART CARD

2.1 Smart Card Definition

Smart card has lots of definition in different sources. The easiest definition is "A credit-card sized tamper resistant plastic card that contains a microprocessor that can store and process data" (Glossary, n.d.). Even in the easiest definition it is mentioned that smart cards are credit card sized with an embedded microprocessor chip. The microprocessor's memory module can hold many times the information contained in a credit card's magnetic stripe. "Smart cards" will offer even more conveniences that Magnetic stripe card can't. A single card will store information –credit card, e-cash, health insurance data, motor vehicle permits, currency exchange rates, loyalty programs. Data is also far more secure, since access to files can be protected by keys. Most smart card microprocessors come with built in encryption engines based on powerful algorithms.

Beginning in the mid-1990s, a number of very significant breakthroughs occurred in the chip card industry with the introduction of open systems specifications for Application development. The three leading technologies in this area are Java Card™, Windows® Powered Smart Cards, and MULTOS™. These technology specifications provide an important contribution to the solution towards the multi-Application chip card vision, such as common programming standards allowing Application portability between different card specific implementations.

2.2 The History of Smart Cards

The proliferation of plastic cards started in the USA in the early 1950s. The low price of the synthetic material PVC enabled the production of robust, long-lasting cards, much more suitable for use in everyday life than the previously conventional paper on cardboards equivalents, which were not equipped for coping with mechanical and climatic damage.

The first all-plastic card for trans-regional payment was issued by Dinners Club in 1950. The entry of VISA and Mastercard in to arena led to a very rapid proliferation of plastic money, first in the USA and, a few years later, in Europe and the rest of the world (Rankl & Effing, 1997).

The cards' functions were quite simple. They served as data-carriers protected against forgery and tampering. The card caries the data like issuer's name as surface printed, card number and/or card holder name as embossed and a free signature field. The system's security depended on the care of the retail staff. With increasing proliferation, these basic features no longer proved sufficient. The card's functions had to be extended and improved.

The first improvement consisted of a magnetic stripe on the back of the cards. This allowed further digitized data to be stored in machine-readable form, in addition to the visual data and embossed data. This type of embossed card with a magnetic stripe is still the most commonly used as a method of payment.

Magnetic stripe technology has a crucial weakness, however, in that the stored data on the stripe can be read, deleted and rewritten by anyone with access to the appropriate read/write device. Hence, it is unsuitable for storage of confidential data. Further measures are needed to ensure confidentiality, as well as to protect against tampering. That is why most systems which employ a magnetic stripe card are connected on-line to the system's host computer. However, this generates a considerable data transmission costs. In order to keep cost down, another system had to be found which enables the card transactions of-line, but the system security should not be put at risk.

The development of the smart card leaded new solutions for solving this problem. In 1970, microelectronics had a great progress and this made it possible to integrate data storage and arithmetic logic on a single chip. The idea of incorporating such chip into an ID card was announced, and patent applied for by the German investors Jürgen Dethloff and Helmut Grötrupp in 1968. This is fallowed by a similar application by Kunitaka Arimura in 1970. But, the first real smart card patent announced by Roland Moreno in France in 1974. The great breakthrough was achieved in 1984, France Postal and Telecommunication Service successfully carried out a field trial with telephone cards. By 1986, many millions of French telephone smartcards were in circulation. Their number reached nearly 60 million in 1990, and 150 million are projected for 1996. A comparative pilot project was conducted in Germany in 1984/85. Smart card proved the hero of this plot study (Rankl & Effing, 1997).

Progress was significantly slower in the field of bank cards, in part due to their greater complexity compared with telephone cards. The developments in semiconductor technology lead the development of modern cryptography which can be mathematically calculated by modern hardware and software. The smart card proved to be an ideal medium. It made a high level of security, since it could safely store secret keys and execute cryptographic algorithms.

2.3 Types of Cards and Comparison

A number of commercially available technologies can be considered in the design of a personal identification or credentialing system. Government agencies and private entities have adopted different combinations of identification methods and media.

2.3.1 Embossed Cards

Simple plastic or paper cards with printed visual identification information (e.g., individual name, address, and photo) are used in numerous applications where information is visually verified when the card is presented for identification. Because visual identification is highly dependent on a security officer's ability to recognize images and relies more on individual judgment, visual identification is considered to be one of the least secure identification methods.

2.3.2 Magnetic Stripe Cards

Magnetic stripes have been used on cards since the 1970s for a wide range of applications – from financial credit cards to transit cards to driver's licenses. The magnetic stripe on the back of an ID card is composed of iron-based magnetic particles encased in plastic-like tape. Each magnetic particle in the stripe is a tiny bar magnet about 20-millionths of an inch long. When all of the bar magnets are polarized in the same direction, the magnetic stripe is blank. Information is written on the stripe by magnetizing the tiny bars in either a north or south pole direction with a special electromagnetic writer, called an encoder. Identification information is written to the magnetic media during the personalization process and then read by swipe or insertion readers at the point of interaction. A new magnetic stripe standard for cards will provide more memory capacity than available with previous cards. The user data encoded on magnetic stripes can easily be copied and interpreted using a standard magnetic reader. The data can also be easily transferred to another card. This fact makes magnetic stripe technology most applicable for low security applications. New technology is available, however, that determines the magnetic "fingerprint" of a magnetic stripe card; by adding this as a component of the card data and verifying the fingerprint with a compatible reader, and the magnetic stripe card can be made more secure.

2.3.3 Smart Cards

A smart card includes an embedded computer chip that can be either a microcontroller with internal memory or a memory chip alone. The card connects to a reader with direct physical contact or with a remote contactless electromagnetic interface. With an embedded microcontroller, smart cards have the unique ability to store large amounts of data, carry out their own on-card functions (e.g., encryption and digital signatures) and interact intelligently with a smart card reader. Smart cards are used worldwide in financial, telecommunications, transit, health care, secure identification and other applications. Today’s production microcontroller smart cards can store up to 1GB of usable data. Future versions will surpass this. Through the use of locking mechanisms and encryption, data stored on smart card chips can be made very secure. Smart cards can perform powerful complex operations within their secure internal computing environments including the ability to perform match-on-card biometric operations.

2.3.4 USB

Universal Serial Bus (USB) is a plug-and-play interface between a computer and add-on devices (such as audio players, joysticks, keyboards, telephones, scanners, and printers). With USB, a new device can be added to a computer without having to add an adapter card or even having to turn the computer off. USB supports a data speed of 480 megabits per second. This speed can accommodate a wide range of devices, including MPEG video devices and digitizers. Today, most new computers and peripheral devices are equipped with USB.

USB security tokens are available that can be used to authenticate users to a computer or network. These tokens provide storage for usernames, passwords, biometrics or cryptographic keys.

2.4 Smartcard Enabled Products

This section lists popular security products and explains how smartcards can be used to enhance their security.

2.4.1 Web Browsers (SSL, TLS)

Web browsers use technology such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS) to provide security while browsing the World Wide Web. These technologies can authenticate the client and/or server to each other and also provide an encrypted channel for any message traffic or file transfer. The authentication is enhanced because the private key is stored securely on the smartcard. The encrypted channel typically uses a symmetric cipher where the encryption is performed in the host computer because of the low data transfer speeds to and from the smartcard. Nonetheless, the randomly generated session key that is used for symmetric encryption is wrapped with the partner's public key, meaning that it can only be unwrapped on the smartcard. Thus it is very difficult for an eavesdropper to gain knowledge of the session key and message traffic.

2.4.2 Secure Email (S/MIME, OpenPGP)

S/MIME and OpenPGP allow for email to be encrypted and/or digitally signed. As with SSL, smartcards enhance the security of these operations by protecting the secrecy of the private key and also unwrapping session keys within a secure environment.

2.4.3 Form Signing

Web based HTML forms can be digitally signed by your private key. This could prove to be a very important technology for internet based business because it allows for digital documents to be hosted by web servers and accessed by web browsers in a paperless fashion. Online expense reports, purchase requests, and group insurance forms are some examples. For form signing, smartcards provide portability of the private key and certificate as well as hardware strength non repudiation.

2.4.4 Object Signing

If an organization writes code that can be downloaded over the web and then executed on client computers, it is best to sign that code so the clients can be sure it indeed came from a reputable source. Smartcards can be used by the signing organization so the private key can’t be compromised by a rogue organization in order to impersonate the valid one.

2.4.5 Kiosk / Portable Preferences

Certain applications operate best in a "kiosk mode" where one computer is shared by a number of users but becomes configured to their preferences when they insert their smart card. The station can then be used for secure email, web browsing, etc. and the private key would never leave the smartcard into the environment of the kiosk computer. The kiosk can even be configured to accept no mouse or keyboard input until an authorized user inserts the proper smartcard and supplies the proper PIN.

2.4.6 File Encryption

Even though the 9600 baud serial interface of the smartcard usually prevents it from being a convenient mechanism for bulk file encryption, it can enhance the security of this function. If a different, random session key is used for each file to be encrypted, the bulk encryption can be performed in the host computer system at fast speeds and the session key can then be wrapped by the smartcard. Then, the only way to easily decrypt the file is by possessing the proper smartcard and submitting the proper PIN so that the session key can be unwrapped.

2.4.7 Workstation Logon

Logon credentials can be securely stored on a smartcard. The normal login mechanism of the workstation, which usually prompts for a username and password, can be replaced with one that communicates to the smartcard.

2.4.8 Dialup Access (RAS, PPTP, RADIUS,TACACS)

Many of the common remote access dial-up protocols use passwords as their security mechanism. As previously discussed, smartcards enhance the security of passwords. Also, as many of these protocols evolve to support public key based systems, smart cards can be used to increase the security and portability of the private key and certificate.

2.4.9 Digital Cash

Smart cards can implement protocols whereby digital cash can be carried around on a smart card. In these systems, the underlying keys that secure the architecture never leave the secure environment of hardware devices. Visa Cash, Europay-Mastercard-Visa (EMV), and Proton are examples of digital cash protocols designed for use with smartcards.

CHAPTER THREE
ERROR DETECTION, CORRECTION AND ENCRYPTION

3.1 Error Detection and Correction

Whenever data are transmitted or stored, it should be possible to detect any changes to the data. In particular, stored programs must be protected against corruption, since a single altered bit of program code could cause fault in the program or modify its execution that the required functions could not be done. The EEPROM memory used in smart cards is especially sensitive to external influences, such as heat and voltage fluctuations. Consequently, the sections that perform security related functions must be protected so that undesired changes can be detected by the operating system and their negative effects can be avoided.

Very sensitive contents, such as program code, keys, access conditions must be protected against alteration. Error detection codes (EDCs) are used for this purpose. Error correction codes (ECCs) are an extension of error detection. They make it possible to not only detect errors but to also correct errors to a limited extent.

All of these codes work on the principle of assigning a checksum to the protected data. The checksum is usually stored along with the protected data. It is computed using a generally known algorithm. The data can be checked for changes as necessary by using the EDC. This is done by comparing the stored checksum with the new computed one. Error detection and correction algorithms utilize a wide variety of mathematical procedures. Using such algorithms enormously increases the complexity and size of the program code.

3.1.1 XOR

An XOR checksum is also known as a longitudinal redundancy check (LRC). The XOR checksum computation can be obtained very simply and very quickly. Simplicity and speed are both important criteria for error detection codes used in smart cards. In addition, the algorithm can be implemented very easily. Besides protecting data stored in memory, XOR checksums are also used for data transmission ATR with the T = 1 transmission protocol. An XOR checksum is computed by performing consecutive logical XOR operations on all data bytes. This is performed by byte 1 is XORed with byte 2, the result of this is XORed with byte 3 and so on.

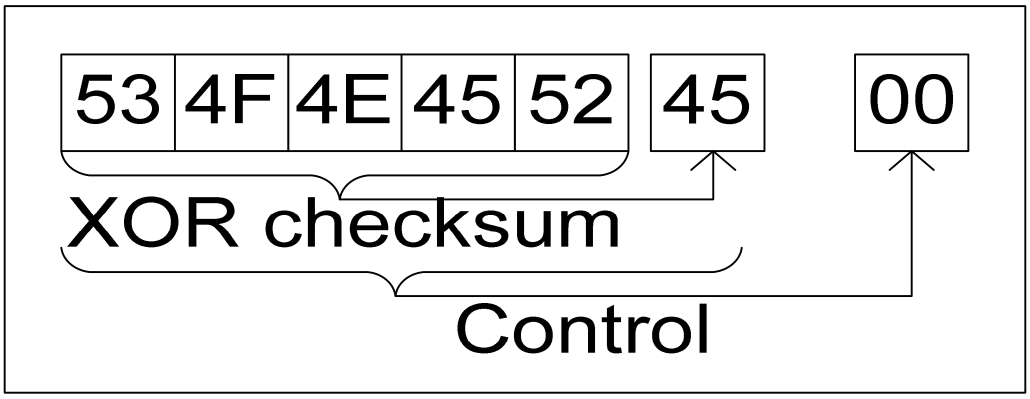

Figure 3.1 - XOR checksum operation

3.1.2 Checksum

The cyclic redundancy check method (CRC) also comes from the field of data communications, but it is significantly better than the XOR method. Still, a CRC checksum is also only an error detection code, so it cannot be used for error correction. The CRC method has been used for a long time in data transmission protocols, such as Xmodem, Zmodem and Kermit, and it is widely used in hard disk drive controllers in a hardware implementation.

A CRC-16 checksum is generated by a 16-bit cyclic feedback shift register, while a 32-bit shift register is used to generate a CRC-32 checksum. The CRC-16 checksum is the most commonly used CRC checksum method for smart cards.

The computation of a CRC checksum proceeds as follows:

- The 16-bit CRC register is set to its initial value
- The data bits are fed into the feedback shift register one after the other, starting with the least-significant bit
- The feedback takes place via bitwise logical XOR operations on the CRC bits.

After all data bits have been fed into the register, the computation is complete and the content of the 16 bits is the desired CRC checksum.

CRC checksum can be verified by again calculating the CRC checksum of the data and comparing the result with the checksum provided with the data. If they are the same, it follows that the data and the checksum have not been altered.

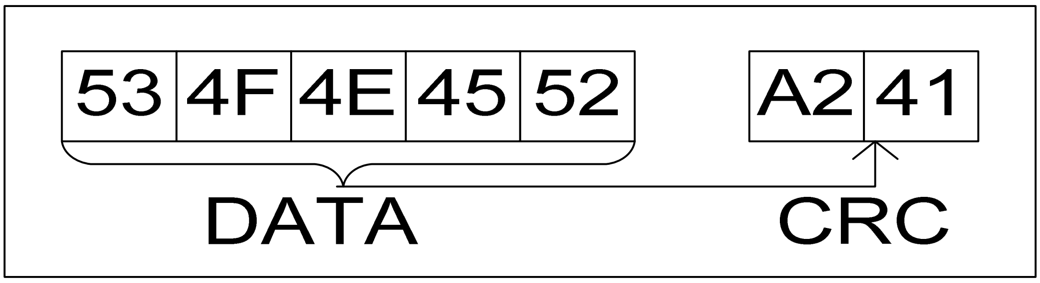

Figure 3.2 - CRC checksum representation

3.1.3 Encryption

In addition to their function as data storage media, smart cards are also used as authorization media and encryption modules. As a result, cryptography achieved a central significance in the early days of smart cards. Nowadays, the procedures and methods of this discipline are firmly established components of smart card technology.

The four objectives of cryptography are maintaining the confidentiality of messages, ensuring the integrity and the authenticity of messages and ensuring the binding force of messages (Rankl & Effing, 2003). These objectives are mutually independent, and they place different demands on the system in question. Confidentiality means that only the intended recipient of a message can decrypt its contents. Authenticity means that the recipient can verify that the received message has not been altered in the course of being transmitted. Nonrepudiation means that the sender can verify that a certain recipient has received a particular message, which means that the message has binding force.

There are three types of data in encryption technology. The first is plaintext, which is unencrypted data. Encrypted data is referred to as ciphertext. Finally there is a key, one or more of which is required for encryption and decryption. These three types of data are processed by an encryption algorithm. The algorithms that are currently used in smart cards are generally block-oriented, which means that the plaintext and cipher text can only be processed in packets with fixed lengths such as 8 bytes with DES.

There are two basic techniques for encrypting information: symmetric encryption (also called secret key encryption) and asymmetric encryption (also called public key encryption.)

3.1.3.1 Symmetric

Symmetric encryption is the oldest and best-known technique. A secret key, which can be a number, a word, or just a string of random letters, is applied to the text of a message to change the content in a particular way. This might be as simple as shifting each letter by a number of places in the alphabet. As long as both sender and recipient know the secret key, they can encrypt and decrypt all messages that use this key.

When using symmetric algorithms, both parties share the same key for encryption and decryption. To provide privacy, this key needs to be kept secret. Once somebody else gets to know the key, it is not safe any more. Symmetric algorithms have the advantage of not consuming too much computing power. A few well-known examples are: DES, Triple-DES (3DES), IDEA, CAST5, BLOWFISH, TWOFISH.

3.1.3.2 Asymmetric

The problem with secret keys is exchanging them over the Internet or a large network while preventing them from falling into the wrong hands. Anyone who knows the secret key can decrypt the message. One alternate is asymmetric encryption, in which there are two related keys a key pair. A public key is made freely available to anyone who might want to send you a message. A private key is kept secret, so that only you know it.

Asymmetric algorithms use pairs of keys. One is used for encryption and the other one for decryption. The decryption key is typically kept secretly, therefore called ``private key'' or ``secret key'', while the encryption key is spread to all who might want to send encrypted messages, therefore called ``public key''. Everybody having the public key is able to send encrypted messages to the owner of the secret key. The secret key can't be reconstructed from the public key. The idea of asymmetric algorithms was first published 1976 by Diffie and Hellmann (Rankl & Effing, 2003).

Any message (text, binary files, or documents) that are encrypted by using the public key can only be decrypted by applying the same algorithm, but by using the matching private key. Any message that is encrypted by using the private key can only be decrypted by using the matching public key.

This means that you do not have to worry about passing public keys over the Internet (the keys are supposed to be public). A problem with asymmetric encryption, however, is that it is slower than symmetric encryption. It requires far more processing power to both encrypt and decrypt the content of the message.

3.1.4 Padding

In smart cards, the DES algorithm is primarily used in the two block-oriented modes. However, since the data communicated to the card do not always fit exactly into a certain number of blocks, it is occasionally necessary to fill up a block. Filling up a data block so that its length is an exact multiple of a given block size is called padding.

The recipient of a padded data block has a problem after the data have been decrypted, since he does not know where the actual data stop and the padding bytes start. One solution to this would be to state the length of the message at the beginning of the message, but this would change the structure of the message, which is generally undesirable. It would also be especially onerous with data that do not always have to be encrypted, since in this case no padding would be needed and thus no length as well. In many cases, therefore, the structure of the message may not be changed.

It is important for the recipient to know whether messages are always padded or padded only if necessary. If padding only takes place when the length of the data to be encrypted is not an integer multiple of the block length, the recipient must take this into account.

3.1.5 Message Authentication Code

A cryptographic message authentication code (MAC) is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC. The MAC value protects both a message's integrity as well as its authenticity, by allowing verifiers to detect any changes to the message content.

While MAC functions are similar to keyed hash functions, they possess different security requirements. To be considered secure, a MAC function must resist existential forgery under chosen message attack. This implies that an attacker be unable to find any two messages M and M^1 which both produce the same MAC under some unknown secret key. A MAC may be considered secure even if the key-holder can efficiently find collisions.

MACs differ from digital signatures, as MAC values are both generated and verified using the same secret key. This implies that the sender and receiver of a message must agree on keys before initiating communications, as is the case with symmetric encryption. For the same reason, MACs do not provide the property of non-repudiation offered by signatures: any user who can verify a MAC is also capable of generating MACs for other messages.

3.1.6 Random Numbers

A random number generator is a computational or physical device designed to generate a sequence of numbers that does not have any easily discernable pattern, so that the sequence can be treated as being random. Random number generators have existed since ancient times, in the form of dice and coin flipping, the shuffling of playing cards.

Random numbers are repeatedly needed in connection with cryptographic procedures. In the field of smart cards, they are typically used to ensure the uniqueness of a session during authentication, as padding for data encryption and as initial values for send sequence counters. The length of the random number needed for these functions usually lies in the range of 2 to 8 bytes. The maximum length naturally comes from the block size of the DES algorithm.

The security of all these procedures is based on random numbers that cannot be predicted or externally influenced. The ideal solution would be a hardware-based random number generator in the card's microcontroller. However, this would have to be completely independent of external influences, such as temperature, supply voltage, radiation and so on, since otherwise it could be manipulated. That would make it possible to compromise certain procedures whose security relies on the randomness of the random numbers. Current random number generators in smart card microcontrollers are generally based on linear feedback shift registers (LFSRs) driven by voltage-controlled oscillators.

CHAPTER FOUR
SECURE APPLICATION MODULE

Secure Application Modules are separate and tamperproof modules for security-critical functions. They provide secure storage and processing, may hold data such as master keys, and carry out encryption. They are concerned particularly with the functions and data handled by card-accepting devices.

They can be smart cards, chips, dongles or black boxes, but they are usually a chip cut out of a smart card with a pair of scissors or a dangerous machine, leaving a rim of plastic around the chip. A chip is best, because it can be put into a device so that it cannot be removed without damaging the device.

The SAM manages keys, by linking the unique numbers stored in the shell using a special numbering system. During personalisation, in terminals and during transactions it ensures that the numbers really are unique. It gives access rights to the shell and creation and modification rights for products, and protects transactions, for example by imposing functional and value limits on terminals.

Security Application Module is used to perform transactions. Different types of SAMs are used for each step of security scheme. SAMs are named according to their physical form and where they are used. SAM for HOST (HOST SAM), SAM for personalization and SAM for terminal are some examples.

- SAM for HOST (HOST SAM): This type of SAM is designed for checking signatures, batches of transactions from the merchants for settling and to generate credit cryptograms. The Host SAM is capable of handling large batches of transactions. Usually, it is made as a tamper-proof black box located in a safe environment.

- SAM for personalization : In most cases, this SAM is based on secure software design and is placed in a safe environment. Usually this SAM is able to address several personalization devices at the same time. The SAM for personalization must be capable of handling large data files and have enhanced computation capacities. It has a special set of commands to compute cryptograms as required for downloading keys and secret codes.

- SAM for terminal : A terminal SAM is built in a Merchant's terminal. It stores all sensitive data needed to establish a transaction (keys, secret codes) and performs mirror functions of the end user smart card commands involving the use of cryptography in an application upon request. By means of using a Terminal SAM, the execution of all sensitive operations is performed by the SAM rather than the terminal, which in many cases cannot be considered as a trusted device. The SAM supports a subset of standard security commands, Message Authentication Cryptogram (MAC) computation for batch signature plus some mirror payment functions for cards which are used in money based applications.

4.1 Hardware Security Module

A proven deterrent to information access attacks is the implementation of a hardware security module (HSM) (Cren, 2001). These devices are especially designed to withstand determined attempts to extract information from their memory. HSMs are generally designed in such a manner that they are highly resistant to physical, software or other kinds of attacks. They essentially take on the role of keykeeper in the sense that security sensitive key data is stored within an area that includes a physically confined boundary to the host system.

Security application modules provide tamper evidence or detection logic, which assures that if an attempt was made to gain access to secure storage areas, all data is either erased or lost due to physical damage inflicted to the module.

The primary benefit is obviously the enhanced security, but in most cases these devices will also perform a range of other services such as cryptographic processing acceleration and fast key generation. This can help to reduce the load on the host system CPU which would no longer be tied down with performing crypto operations.

4.2 Purchase Security Module

This security is based on the use of the smart card and the process of mutual authentication: the merchant's terminal authenticates the card, the card authenticates the terminal. The payment transaction is made through an exchange between the chip of the e-purse card and the chip of the merchant terminal PSAM. The use of cryptography during the process offers the high level of security of e-purse transactions.

Although it is not the merchant who is authenticated but his terminal, more precisely his PSAM, security is unquestionable. Without a PSAM the terminal cannot accept epurses. In order to obtain a PSAM, the merchant must have signed an e-purse acceptance contract with an acquiring bank which will then deliver the PSAM.

This level of security is valid for existing proprietary e-purses, it will be the same for epurses developped under the new standard Common Electronic Purse specifications (CEPS). This standard has been designed with the idea of using e-purse over the Internet.

4.3 Key Management:

Key management is an integral and significant part of a card management program. Anyone planning to implement a smart card program should have the resources available to ensure a complete and thorough understanding of card keys. It is important to understand how keys will be used, especially if the card system plans to work with more than one organization or entity. Keys hold the secret to the system. If not managed properly, the integrity of the entire system can become questionable and thereby useless.

Key management is an application that is used for generating and maintaining cryptographic keys. An interface between the card management system and the key management system makes it very easy to import keys into the card management system where they can be used to secure smart cards. Key management is the procedure to control key generation, key storage, key distribution, key usage, and key destruction.

During the pre-issuance phase of life cycle management, the card manufacturer should generate three key sets known as the transport key, master key, and the OP master key, which is injected into the smart card. The card manufacturer's OP master key set is wrapped with the transport key to send to the card issuer for the key ceremony. The key ceremony initializes the key sets into the card issuer's HSM and generates card issuer keys. During the card issuance phase, key pairs in the smart card are produced with the generation of the ID and email signature key, if required. Other activities, which may need to be considered in the card life cycle management. Following the issuance of the smart identification card, the agency must provide a method to update the smart card keys, replace PKI certificates, regenerate the PKI signature and encryption key pairs, and allow PIN reset. Following the issuance of the smart identification card, the agency must provide a method to update the smart card keys, replace certificates, regenerate the signature and encryption key pairs, and allow PIN reset.

4.4 Security Functions

4.4.1 External Authenticate

The external authenticate command is used by the card to authenticate the host and to determine the level of security required for all subsequent commands.

USER CARD	TERMINAL APPLICATION	SAM
CSN =>	<= Read CSN	

IAC = 3DES(TRnd,Kauth)
IAC[3-0] =>
	GetChallange =>
	<= IntAut(TRnd)
	Verify IAC
	ExtAuth(CSN,IAC[3-0]) =>
		<= TRnd
		Kauth = 3DES(CSN,Mkauth)
		IAC' = 3DES(TRnd,Kauth)
		Compare IAC'[3-0] = IAC[3-0]
		<= if yes return ACK
		<= if no decrement counter + Reset SAM

Figure 4.1 - External authentication

4.4.2 Internal Authenticate

The internal authenticate command is used by the host to authenticate the card. The command computes a cryptogram based on the card random number and an authentication key. The cryptogram is sent to the user card for verification

USER CARD	TERMINAL APPLICATION	SAM

CRnd =>
EAC' = 3DES(CRnd,Kauth)
Compare EAC'[7-4] = EAC[7-4]
if yes return ACK =>
if no return ESC =>
	<= GetChallange
	IntAut(CRnd) =>
	Verify EAC
	<= ExtAuth(EAC[7-4])
		Kauth = 3DES(CSN,Mkauth)
		EAC = 3DES(CRnd,Kauth)
		<= EAC[7-4]

Figure 4.2 - Internal Authentication

4.4.3 Generating Session Key

This command is used to compute a temporary administration key using a specified key, and then a cryptogram using this temporary administration key.

The temporary key is used for the secure messaging process. Only one active temporary key is allowed at given time, therefore by issuing this command, any previous temporary keys will be lost.

USER CARD	TERMINAL APPLICATION	SAM
CSN =>	<= Read CSN	

SK = 3DES(CTC,Ksk)
CR = 3DES(TRnd,SK)
CR[3-0], CTC =>
	GetChallange =>
	
	<= SelSK(TRnd)
	
	Verify CR
	VeriSK(CSN,CTC,CR[3-0]) =>
		<= TRnd
		
		K'sk = 3DES(CSN,MKsk)
		SK' = 3DES(CTC,K'sk)
		CR' = 3DES(TRnd,SK')
		
		Compare CR'[3-0] = CR[3-0]
		<= if yes return ACK
		Keep temp SK for the session
		<= if no decrement counter +
		Reset SAM

Figure 4.3 - Session key generation

4.4.4 Session Key verification

This command is used to generate and verify the temporary key computed by a user card, this command is mirror function of the Select File Key command which is used to secure messaging.

The **Get Challenge** command must be issued prior to this command. If the command fails, the off-nominal use counter is decremented and the SAM is reset.

USER CARD	TERMINAL APPLICATION	SAM
CSN =>	<= Read CSN	
	GetChallange =>	<= TRnd
RN[7-0] = TRnd[7- 4] + CRnd[7- 4]		
SK = 3DES16(RN,Kadm)	<= SelFK(TRnd[7-0])	
CR = 3DES(TRnd,SK) CR[3-0]+RN[7-0] =>(12Bytes)	VeriFK(CSN, RN[3-0], CR[3-0]	K’adm = 3DES(CSN,Mkauth) SK’ = 3DES(TRnd[7-4] || RN[3-0],K’adm) CR’ = 3DES(TRnd,SK’) Compare CR’[3-0] = CR[3-0] <= if yes return ACK <= if no decrement counter + Reset SAM

Figure 4.4 Verification of session key

CHAPTER FIVE
JAVA CARD APPLICATION DEVELOPMENT

Java card technology provides, means to program Smart Cards using a subset of the java language. Nowadays there is a great variety of different java card devices/platforms available on the market. Although they all have to conform to the java card specification, they usually differ slightly from each other. It's very common that each java card device producer provides its own java card development environment. It can be a complete Integrated Development Environment or just a set of tools/scripts/compilers. None of those tools provide a uniform, high level, UML supported environment for creating java card applications. Some of those tools are not very user friendly either.

5.1 Java Card API

Smart card security is a complex multi-dimensional problem. There are different costs associated with different levels of security. Different cryptographic algorithms require chips at various costs, just as different security evaluations and certifications can vary costs in time and money. However, products at any level of security can benefit from Java Card technology.

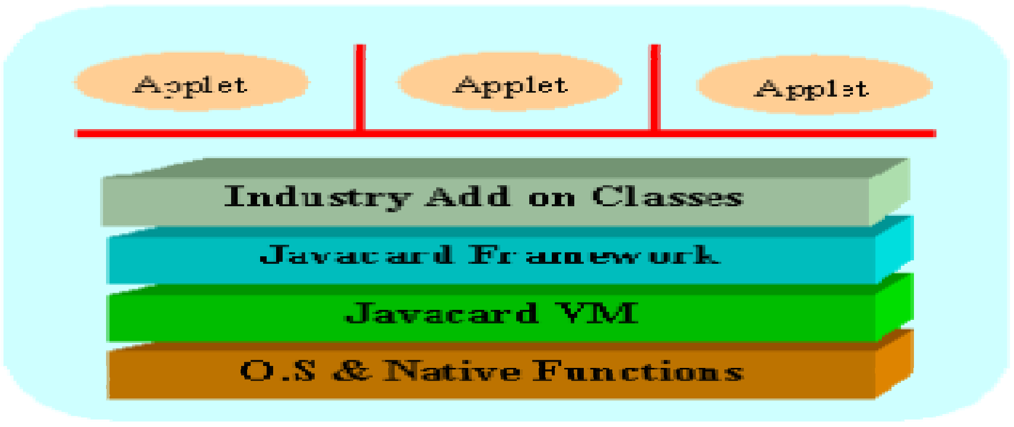

Figure 5.1 - Java card structure

The Java Card platform provides a secure execution environment with a firewall between different applications in the same card. This allows different applications on the same card to function separately and independently from each other as if they were on separate cards (Chen & Giorgio, 1998). The Java programming language and the Java Card API allow development using modern object-oriented programming to create secure applications quickly and easily. By contrast, traditional smart card application programming has used assembly language or the C programming language, which forces the security evaluation to look at the entire application as a unit to verify behavior. Java Card applications can encapsulate sensitive data and algorithms within objects, which have provable behavior and increased security. The result is code that is simpler and easier to develop and maintain. Thus, benefits including lower cost, faster time to market and higher security are realized.

One of the most important security benefits of the Java Card platform is the link it provides between smart cards and the larger world of Java technology. The Java platform has taken a leading role in the academic and developer community, creating a strong base of technical knowledge and shared understanding. Many of the issues surrounding security and "correctness" of operation using the Java programming language and Java Card technology have been widely researched and implemented in multiple contexts. Experts familiar with these issues can be found around the world. Smart card application developers and issuers benefit from all of this research on the Java Card platform, other Java platforms and the general principles underlying all Java technology.

5.2 Global Platform

Global Platform is an international, non-profit smart card association. Its goal is to create and promote global smart card technology specifications, including specifications for smart cards, smart card devices, and smart card systems (Global Platform, 2003). Throughout the world there are currently approximately 20 million individuals use smart cards that are implemented using Global Platform specifications. Global Platform serves the following industries: retail, health care, government, transit, financial, and mobile telecom. Global Platform’s strategy is to create systems that are interoperable, backwards-compatible, and standards-based.

5.3 Java card applet

Java card is a special type of Smart Card. It provides a Java Virtual Machine, as Runtime Environment for applications and applets, and a system memory, as the storage for data. Using the java API set provided by the java card, the developing time is reduced. Furthermore, the same application can be executed on java cards produced by different vendors (Sun Microsystem, 2002). It has the same form-factor of a Smart Card, so it is comfortable to carry too.

Java Card technology allows applets written in the Java language to be executed on a smart card. It defines a Java Card Runtime Environment (JCRE) and provides classes and methods to help developers create applets. Applets run within the JCRE.

A Java Card communicates with a card acceptance device (CAD). The CAD selects an applet on the card and sends it a series of commands to execute. Each applet is identified and selected by its application identifier (AID). Commands such as the selection command are formatted and transmitted in the form of application protocol data units (APDUs). Applets reply to each APDU command with a status word that indicates the result of the operation. An applet can optionally reply to an APDU command with other data.

The Java Card is being implemented on many form factors. The Java iButton from Dallas Semiconductor is one of the first devices claiming support for the Java Card 2.0 API. Java Card isn't just for cards, though.

5.4 Java card security

Java Card technology was developed specifically to enhance the security of smart cards. Important security safeguards are built into the Java programming language itself. The Java Card platform provides further security enhancements, such as transaction atomicity, cryptographic classes and the applet firewall. Additional security is provided by the "split virtual machine" architecture of the Java Card platform on both the workstation side and the card side. The Java Card technology also embraces techniques using compressed archive files with cryptographic signatures to provide tamper-proof distribution and installation procedures for Java class files and Java Card applets.

Complementing features inherited from the Java language, the Java Card platform provides several specific enhancements to security. The Java Card framework and run-time environment provide these essential security features: transaction atomicity, the applet firewall, and classes to support cryptographic signing and authentication of CAP files.

5.5 Java Card Development Tools

Sun java card Development Kit (JCDK) is a reference implementation of the java card development kit and tools. On its own the kit does not operate on any real java card devices, but such a device can be simulated or emulated by the kit tools. The package includes:

- Class file verifier and converter, which makes sure that a given java card applet complies to java card language restrictions and creates a .cap file suitable for downloading to a java card device.
- jcwde tool (Java card Workstation Development Environment), which is a simple java card simulator (e.g. it does not allow saving the state of a smart card between subsequent runs).

- cref tool (C reference implementation of a java card environment). This tool serves as a fully operational java card emulator. It operates on java card EEPROM images and allows saving a smart card's state after each session in form of such an EEPROM image.
- apdutool, which is used to send Application Protocol Data Units to a simulated/emulated java card. APDUs are the only means of communication between a smart card and the host application/system.

5.5.1 Eclipse Platform

The Eclipse Platform is designed for building integrated development environments (IDEs) that can be used to create applications as diverse as web sites, embedded Java™ programs, C++ programs, and Enterprise JavaBeans™.

5.5.2 JCOP Tools 3.0 (Eclipse Plugin)

IBM JCOP Tools 3.0 ("JCOP Tools") provide a set of development tools for the successful development, testing, and deployment of applications for any generic Open Platform Java Card, with specific support for the IBM JCOP platform. JCOP itself is the IBM BlueZ implementation of the basic specifications JavaCard 2.1.1 and OpenPlatform 2.0.1 including refinements from Visa International set in the Visa OpenPlatform Card Implementation Requirements. Applications for a generic Open Platform compliant JavaCard can be fully and conveniently developed using the JCOP Tools.

JCOP Tools provide both a set of command line tools as well as a fully integrated, graphical development environment, the IDE, allowing for all the individual development steps to be performed in a single application. The IDE allows for the creation, modification and management of project data and application source code, simplifies error detection and trouble shooting during the compilation and applet conversion process, and offers a powerful testing and debugging environment for JCOP applications. The JCOP Tools feature:

- JCOP simulation programs which are executed on the development host, but behave very similar to physical JCOPs and cards,
- Debugger, allowing for the debugging of JavaCard applications at the source code level,
- 3. Powerful shell for either issuing interactive commands or executing long-running batch scripts to auto-test the application.

Thus, the JCOP Tools enable a flexible and time-efficient development process on the fast and convenient development host, while still allowing for the final deployment and testing of JCOP applications on real smart cards compliant with JavaCard 2.1.1 and OpenPlatform 2.0.1. With the support of PC/SC and contact-less readers, the JCOP Tools can communicate with a wide range of available card readers and terminals. Additionally, JCOP Tools do not only support the Microsoft Windows operating system family, but can also be deployed on Linux systems or even Mac OS X computers.

5.5.2.1 Requirements

- Supported Host Operating Systems
- Java Runtime Environment (JRE) Version 1.3.x or later
- Eclipse SDK 2.1.x
- Drivers - PC/SC for Windows, the optional M.U.S.C.L.E. for Linux

CHAPTER SIX
PC APPLICATION DEVELOPMENT

6.1 Personal Computer/Smart Card (PC/SC)

The PC/SC Workgroup was formed in 1996 and included Schlumberger Electronic Transactions; Bull CP8, Hewlett-Packard, Microsoft, and other leading vendors. This group has developed open specifications for integrating smart cards with personal computers. The specifications are platform-independent and based on existing industry standards. They are designed to enable application developers to create smart card-based secure network applications for banking, health care, corporate security, and electronic commerce. The specifications include cryptographic functionality and secure storage, programming interfaces for smart card readers and PCs, and a high-level application interface for application development. The specifications are based on the ISO/IEC 7816 standard and support EMV and GSM application standards.

6.2 Open Card Framework

The Open Card Framework is a set of guidelines announced by IBM, Netscape, NCI, and Sun Microsystems, Inc., for integrating smart cards with network computers. The guidelines are based on open standards and provide architecture and a set of APIs that enable application developers and service providers to build and deploy smart card solutions on any Open Card-compliant network computer. Through the use of a smart card, an Open Card-compliant system will enable access to personalized data and services from any network computer and dynamically download from the Internet all device drivers that are necessary to communicate with the smart card. By providing a high-level interface, which can support multiple smart card types, the Open Card Framework is intended to enable vendor-independent card interoperability. The system incorporates Public Key Cryptography Standard (PKCS) - 11 and is expandable to include other public key mechanisms.

6.3 Borland C++ Builder

Borland C++Builder is an object-oriented, visual programming environment to develop 32-bit applications for deployment on Windows and Linux. Using C++Builder. It is possible to create highly efficient applications with a minimum of manual coding. C++Builder provides a suite of Rapid Application Development (RAD) design tools, including programming wizards and application and form templates, and supports object-oriented programming with two comprehensive class libraries:

- The *Visual Component Library* (VCL)
- The *Borland Component Library for Cross-Platform* (CLX),

When C++Builder is started to use, you are immediately placed within the integrated development environment(IDE). This IDE provides all the tools you need to design, develop, test, debug, and deploy applications, allowing rapid prototyping and a shorter development time. The IDE includes all the tools necessary to start designing applications, such as the:

- Form Designer, or *form*, a blank window to design the UI for application.
- Component palette for displaying visual and nonvisual components
- Object Inspector for examining and changing an object's properties
- Object TreeView for displaying and changing a components' logical relationships.
- Code editor for writing and editing the underlying program logic.
- Project Manager for managing the files that make up one or more projects.
- Integrated debugger for finding and fixing errors in your code.
- Many other tools such as property editors to change the values for an object's property.
- Command-line tools including compilers, linkers, and other utilities.
- Extensive class libraries with many reusable objects.

C++Builder can be used to design any kind of 32-bit application, from general purpose utilities to sophisticated data access programs or distributed applications.

As the user interface is visually designed for application, C++Builder generates the underlying C++ code to support the application. As the properties of components and forms are selected and modified, the results of those changes appear automatically in the source code, and vice versa.

The source files can be modified directly with any text editor, including the built-in Code editor. The changes are immediately reflected in the visual environment.

6.3.1 Requirements

- Intel Pentium II/400 MHz or compatible
- Microsoft Windows 98, 2000 (SP2), and XP
- 128MB RAM (256MB recommended)
- 650MB hard disk space (full install)
- CD-ROM Drive
- SVGA or higher resolution monitor (800x600, 256 colour)
- Mouse or other pointing device

CHAPTER SEVEN
CONCLUSION

The computer is usually known as combination of a CPU, memory unit and some I/O devices. When discussing about computers, there are lots of different point of views. Some people would say they are too complex, some would say very easy to use but too difficult to develop application and some would say everything is possible with a computer. It is easily seen that smart card is also a computer. It also have a CPU, memory unit and I/O. When looked at in this manner, it is possible to use smart card in many applications. The java card increases the number of this applications, makes the aplication development easier and allows the development time shorter.

Smart card applications in GSM or banking are usually nation or world wide. That makes the number of cards used in those areas very very high. If a developer wants to use one of those type of cards, it is not possible to get engineering sample. Even it is possible, it cost more than a home developer could pay. But the other side, it is possible to get java card in small amounts. All java card manufacturer supply Integrated Development Environment(IDE) for their products. Developer may buy this IDE from developer or use standart java development environments supplied by SUN free of charge. The number of java card manufacturer is not high enough. It is expected that the java cards and IDEs should be very expensive, but is it not. A java card IDE may cost less then 200$ and a card may cost about 5-10$, depending on the manufacturer.

This study meets home developer with java cards and java card development environments. The java card is not as complex as other smart cards. If the developer is familiar with java by previous experiences, he can easily adapt to java card and its environment. Java card has a Java Runtime Environment(JRE) installed in it like all other java enabled product. This JRE gives platform independency to user and makes the adaptation time shorter. Application developer does not need to learn a new language. He can use java language with java card specified API, which is reduced form of standart java.

With the growing disk space in computers, we have lots of data stored. Some of data are needed to be hidden for security. There are wide variety of programs and plug-ins available in software market. Like the other software all of them can be cracked. The best way to make the data secure is to use a hardware which is unique in the world. The java cards and IDE allow developer to develop application on this unique hardware. Therefore, it is possible to have home made, uncrackable, trustable and tamper proof security application module(SAM).

This study gives information about infrastructure of a personal security application module(SAM) and java card. A sample java card application has also been developed in order to help reader to understand Java Card and SAM.

REFERENCES

Chen & Giorgio, 1998, Chen Z., Giorgio R. D. (1998) *Understanding Java Card 2.0* Retrieved May 3, 2005, from http://www.javaworld.com/jw-01-1998/jw-01-javadev.html

COMMISSION OF THE EUROPEAN COMMUNITIES (1992, 1993). *Information Technology Security Evaluation Manual (ITSEM) Version 1.0*

Cren (2001) Retrieved May 20, 2005, from http://www.cren.net/crenca/onepagers/hsm.html

Eracom. (2001). *The Role of Hardware Security Modules Within Windows 2000 PKI.*

Gemplus. (1998). *SAM for MP-COS EMV Refeernce Manuel*

Global Platform. (2003). *Card Specification Version 2.1.1*

Glossary (n.d) Retrieved May 1, 2005, from http://www.vpsource.com/glossary.html

Rankl, W. & Effing, W. (1997). *Smart Card Handbook.* John Wiley & Sons

Rankl, W. & Effing, W. (2003). *Smart Card Handbook.* (3rd ed). John Wiley & Sons

Sun Microsystem. (2002). *Java Card™ 2.2 Application Programming Interface*

Sun Microsystem. (2001). *Java Card™ Platform Security. Technical white paper*

VISA, (2001). *Welcome to Visa Integrated Circuit Card Specification*