INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
.800-521-0600

UMI[®]

NOTE TO USERS

Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.

66-86

This reproduction is the best copy available.

UMI

Using Predicate Transition Nets Approach to Verify the Security of the Secure Electronic Transaction Protocol

by

Zhenren Zhou, B.Eng.

A Thesis Submitted to
the Faculty of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

April 2000

© Copyright

2000, Zhenren Zhou

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52382-9

The undersigned hereby recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

**“Using Predicate Transition Nets Approach to Verify the
Security of the Secure Electronic Transaction Protocol”**

submitted by

Zhenren Zhou, B.Eng.

in partial fulfillment of the requirements for
the degree of Master of Computer Science

Thesis Co-Supervisor

Thesis Co-Supervisor

Director, School of Computer Science

Carleton University

April, 2000

Abstract

Providing customers with a secure way to buy goods over unsecured networks is one of the most important challenges in electronic commerce. Secure Electronic Transaction (SET) is one promising standard protocol that provides a way for customers to use their payment cards over the Internet securely and confidently.

This thesis presents the security mechanisms used by the SET protocol. It summarizes the five key sub-transaction processes of payment transaction system of the SET protocol and inherent security services provided by SET. It also presents the workflow of the payment transaction system of SET with Predicate Transition Nets approach. This thesis focuses on verification of security issues on the SET protocol. Simulation models are used to investigate the potential security threats at application level. Reachability Analyzer PROD is used throughout the implementation of the verification process. Logic of Authentication is used for analyzing the simulation results.

Acknowledgments

I would like to sincerely thank my supervisors Dr. George M. White and Dr. Evangelos Kranakis for all their help. A very special thank you goes to Dr. G. M. White who helped me consistently and constructively the entire way. He was always there when I needed to discuss with him. His time, support, advise, ideas, and comments are appreciated.

I would like to dedicate the thesis to my parents and sisters for their love and encouragement. A very special appreciation goes to my wife Chen Ma for her love, encouragement, support and understanding.

Table of Contents

1	Introduction	1
1.1	Security in Electronic Commerce Transaction	1
1.2	Motivation	6
1.3	Thesis Overview	10
2	Background	11
2.1	Security Mechanisms in Electronic Transaction	11
2.1.1	Cryptography	11
2.1.1.1	Secret Key Cryptography	12
2.1.1.2	Public Key Cryptography	15
2.1.2	Digital Signature	17
2.1.3	Integrity	19
2.1.4	Authentication	20
2.2	Overview of SET	21
2.3	Overview of the Security Services in SET	26
2.4	Threats in SET	29
3	SET Payment Process and Transaction	30
3.1	Cardholder Registration	30
3.2	Merchant Registration	34
3.3	Purchase Request	36

3.4	Payment Authorization	40
3.5	Payment Capture	42
4	Predicate Transition Net Approach and PROD	45
4.1	Predicate Transition Net	45
4.2	Pr/T-Net Reachability Analysis Tool PROD	51
5	Simulation and Verification	57
5.1	SET Payment Transaction	58
5.1.1	Cardholder Registration	58
5.1.2	Merchant Registration	66
5.1.3	Payment Request /Authorization / Capture	67
5.2	Security Issues on SET Payment Transaction	69
5.2.1	Masquerade	72
5.2.2	Modification of Information	87
5.2.3	Leakage of Information	94
5.2.4	Repudiation	99
5.3	Summary	100
6	Conclusions	103
	References	106

List of Tables

Table 5.1 Variables in the Complete Process	58
Table 6.1 Conclusions of the Verifications of the SET protocol	104

List of Figures

Figure 2.1	Secret Key Cryptography	12
Figure 2.2	One Round of DES	14
Figure 2.3	Public Key Cryptography	15
Figure 2.4	Secret Key Cryptography — Authentication	16
Figure 2.5	Digital Signature	18
Figure 2.6	Integrity Check-Value	19
Figure 2.7	Hierarchy of Trust	25
Figure 3.1	Cardholder Registration	30
Figure 3.2	Merchant Registration	34
Figure 3.3	Purchase Request	37
Figure 3.4	Dual Signature	39
Figure 3.5	Payment Authorization	40
Figure 3.6	Payment Capture	43
Figure 5.1	Workflow of Cardholder Registration	61
Figure 5.2	A diagram of the Pr/T Net for the Cardholder Registration process	62
Figure 5.3	Workflow of Merchant Registration	66
Figure 5.4	Processes of Purchase Request /Authorization / Capture	67
Figure 5.5	Workflow of Processes of Purchase Request /Authorization / Capture	68
Figure 5.6	The Pr/T Nets of the Whole (Key) Process of the SET Protocol	70
Figure 5.7	Masquerade in Cardholder Registration	73
Figure 5.8	Modification of Information in Cardholder Registration	89
Figure 5.9	Leakage of Information in Cardholder Registration	94

Chapter 1

Introduction

1.1 Security in Electronic Commerce (Transaction)

Electronic commerce, in the form of EDI (Electronic Data Interchange), has been in use since the early 1980s. It may be defined from an online perspective as the “capability of buying and selling products and information on the Internet and other online services” [25]. It describes automated business-related transactions and typically involves information communicated via electronic mail, EDI, or the World Wide Web.

Like any business transaction, electronic commerce transactions require a certain amount of trust between buyer and seller. In a traditional paper-based exchange, a merchant only sells to the trusted customer, and a customer only buys from the trusted vendor. In an electronic interchange, however, trust cannot be easily achieved in that the buyer and seller might never meet. The transaction, which might be exposed to unknown parties, must depend on untrusted communications and storage systems. Therefore, there is an inherent risk in this kind of transaction.

The key aims of electronic commerce security systems are to facilitate trust between merchants and clients, i.e., ensure the reliability of business transactions over open, insecure networks such as the Internet, and to secure business information between users who might never meet each other.

In terms of electronic data and funds, business information in electronic commerce should be absolutely secure during its transmission and transaction. This may simply be achieved by setting up closed, point-to-point networks (such as leased circuits between major trading partners). But, obviously, we cannot do this for the global business as global trade needs open, interconnected, unregulated, and unpoliceable networks such as the Internet.

Open networks are increasingly being used as platforms for communication in our society. Open and accessible, they allow rapid and efficient worldwide exchanges at low cost. This will lead to new forms of business configuration (e.g. "virtual" enterprises, work collaboration across the globe), of private communication (e.g. e-mail) and of organization of public services (e.g. electronic tax declaration). Open networks also have the capacity to offer substantial opportunities for global electronic commerce in goods and services, which can be ordered, supplied and paid for electronically.

However, the realization of such developments is hampered by certain observed insecurities typical to open networks: messages can be intercepted and manipulated, the validity of documents can be denied, personal data can be illicitly collected. Fraud is already increasing in several forms.

Therefore, important electronic documents are usually only exchanged in so-called "closed networks", which involve users between whom contractual relationships and mutual trust already exist. But this model cannot be transferred to open networks because of the absence of such relationships between users. As a result, the attractiveness and advantage of electronic commerce and communication cannot be fully exploited.

In order to make good use of the commercial opportunities offered by electronic communication via open networks, a secure and trustworthy environment is necessary.

Security in E-commerce can be categorized at different levels.

- **Network level security**

By using both internal and external networks, commercial messaging in E-commerce would drastically increase the vulnerability of the transactions to the security lapses of the networks.

As E-commerce applications inherently utilize the underlying network, it is imperative that a proper network level security model be established and enforced. The key issues at network level are the availability of network resources with proper backups in terms of both hardware and software, provision of suitable firewalls between the intranet and the Internet and securing the vital Web and application servers against hacking, virus attacks or spoofing. Application level encryption can effectively act against spoofing.

- **Application level security**

A typical E-commerce application involves several two-party transactions depending on the model used. As we try to imitate the manual paper-based transaction systems used with trust as an important element, we have several security issues:

- ☐ **Authenticity:** The ability to know the true identities of communicating parties.
- ☐ **Confidentiality(Privacy):** The ability to control who can see (or cannot see) information and under what terms. Transactions must be kept private in the sense that eavesdroppers cannot understand the message content.
- ☐ **Integrity:** The ability to assure that stored or transmitted information is unaltered.

Transactions must not be tampered or interfered with.

- ❑ Authorization(Access Control): The ability to provide protection against unauthorized access to information.
- ❑ Non-repudiation of origin and receipt: The ability to provide the proof of the origin of information in order to prevent any attempts by the sender (or the receiver) to intentionally deny having sent (or received) the information.
- Message level security

Commercial transactions have been using message and document standards. These have developed into international standards like UN/EDIFACT standard [16]. These standards have also been provided with security solutions that could be applied to individual messages, a collection of messages or an interchange.

- Communications and services infrastructure

The security of the system also depends on the efficiency and reliability of the system. Insufficient bandwidth, inefficient service and non availability of access affect wide spread use of the Internet for commercial messaging.

- Financial infrastructure

Because of the advantage of E-commerce, more and more institutions are being involved to develop and use E-commerce. As a result, E-commerce and financial networks with inter institutional coordination and standards need to be established. Links between E-commerce networks and financial networks should also be formed.

An important safeguard in financial activities is audit and control. With the use of E-commerce and the Internet, the need to provide for audit and control in a computerized environment becomes absolutely essential.

- Legal infrastructure

The success of the implementation of security measures depends on the available legal infrastructure. Some of the necessary legal instruments are:

- ❑ Admissibility of IT related evidence,
- ❑ Admissibility of digital signatures
- ❑ Commercial fraud and computer crime,
- ❑ Data protection
- ❑ Contracts dealing with E-commerce including IT products and IT based financial transactions
- ❑ Trans-border transactions

In these matters, each nation would have to depend on the international initiatives such as the UNCITRAL model law [23]. In addition to the provision of the above framework, a large quantity of existing legislation in the form of acts, ordinances, by laws etc. needs to be changed to accommodate E-commerce.

- Technology infrastructure

Secure E-commerce could only be ensured with continuing technology development. The establishment of Research & Development Centers devoted to E-commerce and special topics such as cryptography, awareness creation and the provision of training are all essential. Familiarity with global developments on security, National EDI/EC networks, National EDI/EC bodies, collaboration with United Nations/United States of America/EC-security organizations, shared use of software and secure systems need to be locally available.

1.2 Motivation

Electronic Commerce is rapidly becoming a faster and simpler way to conduct financial transactions, with revenues topping \$24.4 billion and expecting to be over \$638 billion by the turn of the century [18]. There is no doubt that E-commerce, as exemplified by the popularity of the Internet, is going to have an enormous impact on the global financial services industry.

People want the Internet to be as safe as real world transactions. However, lots of people are still fearful of using the Internet for conducting online transactions. They are afraid of the fraud and theft online that they hear about via media. Basically, there are three common security concerns of users:

- The privacy of personal information

People are worried who has access to their information and what their intentions are. It would be easy for an unscrupulous person to eavesdrop on the transaction conducted over the telephone. Users might believe that anytime they go online, their personal information can be stolen, or the Internet is filled with hackers who are just listening to the data being sent over the Internet, waiting for someone to transmit personal information.

- The authenticity and integrity of merchants

Just as in doing business in paper-based way, users need to verify the authenticity of a merchant when conducting business on the Internet. Users must be careful with whom they conduct business with on the Net.

In the online way, a user is not physically present where the transaction is being processed. He does not know who is handling his/her transaction and who has access to

his/her information. If someone was impersonating a merchant online, could he acquire the person's personal information that was sent to the merchant?

- The integrity of transactions

Users want to make sure that the information they are sending is the same information that the merchant receives, and that it hasn't been tampered with or changed en route to the merchant. While this may not seem that sinister on the surface, it definitely can have serious ramifications. What if a person was trying to transfer a lot of money from one bank account to another? While the transaction was being sent, someone hacked the transaction and changed the account number of the destination. The money would be sent to the wrong account and neither the bank nor the user would know until it was all over with. As far as the bank knows, the user legitimately executed the transaction and unless the user has made a paper record of the transaction, the user would have some hurdles to cross before the transaction could be corrected.

The online industry, including banks, brokers, software and hardware companies, does not dispute the fact that it is dangerous to do business on the Internet. But they do stress that, with proper safety precautions, the Internet can become a safe medium for doing business. Commercial businesses have had the means and the resources to establish security precautions for electronic transactions. The online industry would let users believe that everything is safe as long as they take standard precautions or they use its developed transaction protocols.

Most existing work on electronic commerce services concentrates on the development of electronic payment systems. Secure Electronic Transaction (SET), as one of the most promising protocols, is rapidly on the way to becoming a worldwide standard

for payment (credit card) transactions over public networks, although it is still in the stage of development and improvement.

SET is a multi-party protocol for conducting secure card payments over the Internet. The way of inter-operability is designed throughout protocol and message formats. Message integrity, authentication of all financial data and encryption of sensitive data are provided within the entire transaction process.

SET provides numerous security services for electronic transaction, described in detail in Section 2.3.

- Privacy of important data. The confidential order information is transmitted along with the payment data.
- Authentication of a cardholder for a branded bank card account. The cardholder account authentication is ensured by using a cardholder certificate and digital signatures. Authentication that a merchant can accept branded bank card payments is via its relationship with an acquiring financial institution. The merchant authentication is ensured by using merchant certificates and digital signatures.
- Preservation of the integrity of payment data. The payment information integrity is ensured by using a digital signature.
- Digital certificates (X509 v3) that associate cardholder and merchant with a financial institution and a payment system such as Visa and MasterCard payment system.

Although SET promises a safe future for this new era in commerce [20], nonetheless, potential malicious threats, including masquerade, modification of information, leakage of information, and repudiation, may occur throughout the payment transaction process.

Therefore, necessary security verification tests should be done before SET becomes the standard for the payment transaction protocol.

"The pilot will examine the SET standard's ability to ensure the privacy of cardholder information and reduce Internet fraud. First Union is fully committed to providing customers with a safe, efficient means of making purchases over the Internet." The pilot is the first Visa-sponsored SET test in the United States [19]. Widespread SET projects will soon be under way in Europe and Asia, but in the United States testing is limited to isolated financial institutions such as Wells Fargo, which is using VeriFone's vPos software, Microsoft's Merchant Server, and GTE's digital certificates to pilot SET transactions with its merchants [21].

Currently, we only know the general conclusion about security on SET protocol. No technical verification of security on SET has been openly reported. Therefore, it is necessary to verify the security services in SET in order to make SET acceptable worldwide.

This thesis will focus on security issues in SET, in terms of application level security. We will model and simulate the payment transaction process of the SET protocol, including all sub-transaction processes, investigate the security concerns in SET, and concentrate on verifying all the potential threat issues.

It is greatly beneficial for all users who are using or going to use the SET protocol if they know the verification result of security in SET. There may be various ways for the verification process. Here, a widely used model for concurrent and distributed systems called Predicate Transition Nets (more details in Section 4.1) and a Pr/T-Net Reachability

Analysis Tool called PROD (more details in Section 4.2) are used throughout the entire verification process.

1.3 Thesis Overview

The remainder of this thesis is organized as follows:

- Chapter 2 describes the security mechanisms in electronic commerce, the introduction of the SET protocol, and the inherent security services in SET. All kinds of potential threat over the SET protocol are also presented.
- Chapter 3 gives out the detailed description of the key payment transaction processes in the SET protocol.
- Chapter 4 introduces Predicate Transition Nets approach, describes its theory, definition, basic form and execution rules. As a reachability analysis tool for Predicate Transition Nets, PROD is also described.
- Chapter 5 presents the verifications of the security issues of SET protocol. It includes modeling the key payment transaction processes of SET protocol without and with security threats by using Predicate Transition Nets approach, simulating these transaction processes in PROD and analyzing the results of experiment.
- Chapter 6 contains the conclusion and the contribution.

Chapter 2

Background

Some background knowledge is given before the detailed SET Payment Process and Transaction is presented. First of all, the security mechanisms related to electronic commerce and SET are briefly introduced. More theory can be found in references [5][10][26][38]. Secondly, general descriptions of SET protocol and security services over SET are given; more details are found in references [39][40][41]. Finally, potential threats to payment transaction process of SET are described.

2.1 Security Mechanisms in Electronic Transaction

Security mechanisms are designed to detect, prevent, and recover from security attacks, i.e., any actions that compromise the security of information and system [37].

2.1.1 Cryptography

Cryptography, the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [15], has existed for centuries. Once the preserve of intelligence agency, it is increasingly been as an enabling technology for the information society,

especially electronic commerce. Cryptography allows users to safeguard and verify sensitive data such as credit card numbers, and other information stored in computers or transmitted over open networks such as the Internet.

In general, Cryptography does the following two things:

- Encrypts (encodes) the original message (the plaintext) by the use of some operations or transformations to create the encoded version of the message (the ciphertext).
- Decrypts (decodes) the encrypted message via the use of some operations or transformations to the original message.

The basic techniques used in cryptography are substitutions (each character is replaced by another so that the message is obscured) and transpositions (the characters remain the same but their order is changed to accomplish the obscuring). There are two types of cryptography, Secret Key Cryptography and Public Key Cryptography.

2.1.1.1 Secret Key (Symmetric) Cryptography

Secret Key Cryptography, uses the same key (which may be any kind of data structure) in the encryption and decryption transformations. It is illustrated in Figure 2.1.

Figure 2.1 Secret Key Cryptography

The sender (system I) and receiver (system II) share an identical key which is kept secret from everybody except these two systems. Only these systems can decrypt the

message by using the key. The key might be a string such as a hexadecimal string, a mathematical function, or a combination of several strings applied in a specified order.

A symmetric cryptosystem operates either as a block cipher or a stream cipher. A block cipher is an encryption scheme which breaks up the plaintext messages into strings of a fixed-size block of plaintext (n bits long, typical value is 64), and encrypts one fixed-size block of ciphertext (n bits long) at a time. A stream cipher is useful in the fact that the encryption transformation can change for each symbol of plaintext being encrypted. In situations where transmission errors are highly probable, a stream cipher is advantageous because it has no error propagation. It can also be used when the data must be processed one symbol at a time. Secret key encryption is most often used in private networks such as Automated Teller Machine (ATM) networks.

There are several implementations of secret key encryption, such as Data Encryption Standard (DES) and International Data Encryption Standard (IDEA).

DES, from IBM [1], is defined by the American Standard FIPS 46-2. It is the most well-known symmetric-key block cipher, and is used in SET protocol [Section 2.2]. DES is a block cipher which operates on 64-bit blocks of plaintext and employs a 56-bit key. Basically, this block is permuted according to a specific regimen, then broken into left and right halves. At this point, 16 rounds of identical operations (called round Function f) are performed, where data is combined with the key. Each round, the key information entered to the round function is called a subkey. The subkeys $K_1, K_2, \dots, K_{16}$ are computed by a key scheduling algorithm. Each round includes expansion E and permutation P . After the 16th time, data is joined back into a single 64-bit block, then

permuted a final time. This last permutation is the inverse of the first one. Figure 2.2 shows a general overview of a single round of DES.

Figure 2.2 One Round of DES

For secret key encryption, the two parties to an exchange must have the same key which must be protected from access by others. Therefore, the strength of this kind of cryptography depends on the key distribution techniques; see [37]. Secure key distribution, furthermore, is cumbersome in large networks and does not scale well to an electronic commerce environment where a company deals with thousands of online customers. Additionally, secret key encryption is impractical for exchanging messages between groups of previously unknown parties over a public network. For example, each customer would need a distinct secret key assigned by the merchant and transmitted via a separate secure channel such as a telephone, in order to conduct secure electronic commerce transactions over the Internet. Such effort would, undoubtedly, increase the

online overall cost. Because of the difficulty of providing secure key management, it is hard to justify that secret key encryption should be selected for commercial implementations of public electronic commerce.

2.1.1.2 Public Key (Asymmetric) Cryptography

Public Key (Asymmetric) Cryptography, first introduced in 1976 by Whitfield Diffie and Martin Hellman[8], uses a pair of related keys, one key is the private key which is known only to the user, and the other key is the public key which can be made publicly known. The message is encrypted with one key and decrypted only with the other corresponding key. Although the two keys are mathematically related, they are generated in such a way that it is computationally infeasible to deduce the secret key from the public key, and vice versa.

There are two basic modes for public key cryptography depending on whether the public key is used as the encryption key or as the decryption key. Figure 2.3 shows Encryption mode and Figure 2.4 depicts Authentication [Section 2.1.4].

Figure 2.3 Public Key Cryptography

Figure 2.4 Secret Key Cryptography — Authentication

There are several algorithmic implementations of a public key cryptosystem. RSA has proven to be an extremely reliable algorithm for performing both public-key encryption and digital signatures [Section 2.1.2] since its invention. Almost all of the major secure messaging protocols, as well as many types of secure communications protocols such as SSL and SET [Section 2.2], have depended heavily on the use of RSA.

The RSA algorithm [36] was first developed by Rivest, Shamir, and Adleman in 1977. The security of RSA is derived from the problem associated with the factoring of large numbers. In fact, finding large prime numbers is relatively easy, but factoring the product of two such numbers is difficult. If the numbers are large enough, factoring requires enormous processing resources so that the problem is considered to become computationally infeasible.

The keys are calculated from two distinct large random prime numbers p and q . Roughly the same length of these prime numbers may be more than 200 digits. The public key is comprised of a pair of numbers, n and e , where the modulus $n = p \times q$. The encryption key e is chosen randomly and should be relatively prime to $(p-1) \times (q-1)$, where the Euler totient function of n is $\phi(n) = (p-1)(q-1)$. A number e ($1 < e < \phi(n)$) is selected such that $\gcd(e, \phi(n)) = 1$. The decryption key d is calculated as $d = e^{-1} \bmod$

$\phi(n)$. The value of d and the prime numbers p and q are kept secret although p and q are no longer needed.

2.1.2 Digital Signature

A digital signature is analogous to a handwritten signature used for centuries to authenticate documents. Everyone's handwritten signature is unique and hard to forge and a document already signed would be hard to repudiate later. The same assurances are desirable for computer generated and transmitted documents. A digital signature is a cryptographic mechanism, which can prove that a document sent to us by somebody was indeed sent to us by that individual. This process is called authentication.

Generally speaking, there are two methods to implement digital signatures.

- By using encryption with pairs of public and private keys and a trusted arbitrator.

Each of the individuals who want to communicate shares a distinct secret key with the arbitrator. First of all, the message is encrypted by the sender and sent to the arbitrator. Secondly, the encrypted message is decrypted by the arbitrator with the secret key known only to the sender and the arbitrator, then the cleartext message is encrypted by the arbitrator with the secret key known only to the arbitrator and receiver, and then the encrypted message is sent to the receiver. Finally, the encrypted message is decrypted by the receiver with its secret key. Obviously, there are a number of drawbacks to this method: The trusted third party is needed, a considerable amount of overhead is required since each message have to be encrypted, decrypted, and transmitted twice.

- By using a public key system such as RSA, in this way, a message is encrypted by the sender with its private key and sent to the receiver, and encrypted message is decrypted

by the receiver with sender's public key. Figure 2.5 shows this method. Apparently, this way ensures the receiver that the sender did the process since only the sender knows its private key. From the digital signature point of view, a sender may not have to encrypt the entire message but rather provide just a signature authentication. For authentication purposes, digital signature algorithm simply encrypts the signature with an additional hash value and time stamp. Another fundamental process, a hash function, is used in both creating and verifying a digital signature. The basic idea of a hash function is that a hash value serves as a compact representative image, i.e., message digest of an input string, and can be used as if it were uniquely identifiable with that string. A hash function is used for data integrity in conjunction with digital signature schemes, where for several reasons a message is typically hashed first, and then the hash-value, as a representative of the message, is signed in place of the original message.

Figure 2.5 Digital Signature

2.1.3 Integrity

The contents of electronic commerce transactions must remain unmodified during transmission between sender and receiver in order to make the transactions successful. All transmissions must be absolutely tamperproof.

The necessary protection can be achieved by using an integrity check-value, an additional data item generated by the sender of a message, which accompanies the plaintext message during transmission. The process is shown in Figure 2.6.

Figure 2.6 Integrity Check-Value

The value of the integrity check-value depends on all bits of the input message — if any bit of the message is changed between the sender and receiver, a different integrity check-value would be generated and thus the receiver would know that integrity had been compromised. For example, after the receiver gets the message, he or she will regenerate the integrity check-value from the message content and check whether or not the two integrity check-values are the same. It is pretty similar to the common error-detection

procedures such as CRC (cyclic redundancy check) in communication systems except that here, an active attacker exists. To protect against an attacker, a message authentication code (MAC) has been widely used in industry [2].

Another approach to generate an integrity check-value is to apply a hash function to a bit string which contains both the message data bits and the data bits of a secret key. This is called a keyed hash function [29].

2.1.4 Authentication

Authentication is the means of gaining confidence that remote people or things are who or what they claim to be. Reliable authentication is needed to enforce access control, to determine who is authorized to receive or modify information, to enforce accountability, and to achieve nonrepudiation.

Authentication can be classified in two categories: user authentication and message authentication. User authentication often depends on password-based authentication which is used to establish a secure connection to the authentication server. Message authentication depends on cryptography authentication. The cryptographic techniques may be combined with hand-shaking protocols to ensure liveness. The authentication exchange mechanism may be used with time stamp, synchronized clocks, two-way or three-way handshakes, and nonrepudiation services.

2.2 Overview of SET

SECURE ELECTRONIC TRANSACTION (SET) is a specification designed to utilize technology for authenticating the parties involved in payment card purchases on any type of online network, including the Internet. SET was developed by Visa, MasterCard, and many other companies such as Microsoft, and IBM.

The SET specification is designed to meet three main objectives. First, it enables payment security for all involved parties, authenticates cardholders and merchants, provides confidentiality of payment data, and defines protocols for potential electronic security service providers. It also enables interoperability among applications developed by various vendors and among different operating systems and platforms. Finally, it strives to achieve acceptance on a global scale.

As the SET 1.0 protocol was finalized, an infrastructure based on this new technology began to emerge to support large-scale usage. For example, software vendors have begun to create the software package, which cardholders, merchants and financial institutions need, in order to take advantage of the security that SET offers. On the other hand, lots of Visa and MasterCard member financial institutions around the world conducted limited scale pilot tests during 1997 in order to better understand the requirements for implementing and supporting the use of SET technology.

The SET Mechanism

SET mainly makes use of cryptography concepts and defines some transaction protocols to securely conduct electronic commerce. The operation of SET depends on

software that implements a series of protocols being installed in either the workstations or servers of the related people and organizations.

There are several key issues in SET payment transaction mechanism:

The participants

- ❑ Issuer: a financial institution that establishes an account for a cardholder and issues the payment card (bank, credit, debit). The Issuer guarantees payment for authorized transactions using the payment card in accordance with payment card brand regulations and local legislation.
- ❑ Cardholder: an authorized payment card holder who is registered with the corresponding issuer to perform E-commerce transaction. SET ensures that during the cardholder's interactions with the merchant, the payment card account information remains confidential.
- ❑ Merchant: a seller of goods, services, or information, who accepts E-commerce payment. In SET, the merchant can offer its cardholders secure electronic interactions. A merchant that accepts payment cards must have a relationship with an Acquirer.
- ❑ Acquirer: a financial institution that establishes an account with a merchant and processes payment card authorizations and payments.
- ❑ Payment gateway: a system operated by an Acquirer or a designated third party that processes merchant payment messages, including payment instructions from cardholders.
- ❑ Brand: payment card brands can protect and advertise the brand, establish and enforce rules for acceptance and use of the payment cards, and provide networks to interconnect the financial institutions. In SET, These brands combine the roles of Issuer and Acquirer in interactions with cardholders and merchants.

❑ Third parties: Issuers and Acquirers sometimes choose to assign the processing of payment card transactions to third-party processors. For the purpose of simplicity, we do not distinguish between the financial institution and the processor of the transactions.

SET Certification

In SET, cardholders, merchants, and acquirers are issued digital certificates [40] by their sponsoring organizations for authentication purposes. There is a hierarchy of Certificate Authorities (CAs). The SET protocol specifies a method of entity authentication referred to as trust chaining. This method entails the exchange of digital certificates and verification of the public keys by validating the digital signatures of the issuing CA. This trust chain method continues all the way up to the CA at the top of the hierarchy, which is referred to as the SET root CA. The SET Root CA is owned and maintained by SET Secure Electronic Transaction LLC (SETCo).

The SET Root CA is highly secure, and off-line CA has four primary functions:

- ❑ Generate and securely store the SET Root CA public and private keys.
- ❑ Generate and “self-sign” the SET Root CA certificates.
- ❑ Process brand certificate requests and generate SET Brand CA certificates.
- ❑ Generate and distribute Certificate Revocation Lists (CRL).

The SET Root CA issues SET compliant digital certificates to payment card brands that wish to participate in SET. The Root CA issues Brand CA certificates in order to enable the Brand to issue certificates to their members, cardholders, and merchants. Figure 2.7 illustrates the hierarchy of trust [40].

SET Key Technology

SET uses RSA for digital signature and key exchange, DES for data encryption, SHA (Secure Hashing Algorithm) for message digest, and X.509 version 3 certificates for certificate authentication. In addition, SET introduces the concept of dual signatures which are used to link an order message with the payment instructions containing cardholder's account information. The order message is sent to the merchant and the payment instructions are sent to the acquirer. The merchant sends an authorization request to the acquirer together with the message digest of the order information. The Acquirer uses the message digest from the merchant and computes the message digest of the payment instructions to check the dual signature.

The Conduct of a Payment Transaction

The actual payment transaction processes are complex, because of the need for multiple security risks to be addressed, for multiple parties to authenticate each other, for the approaches of multiple payment processing services providers to be accommodated, and to reflect the needs and practices of multiple countries around the world.

A general brief description of the SET transaction process is as follows:

- ❑ the cardholder visits the Web pages of the specified merchant, selects items (goods and / or services), and states that he or she wishes to make a purchase;
- ❑ the merchant responds (the merchant's Web server can send a completed order form for the cardholder);

Figure 2.7 Hierarchy of Trust

- ❑ the cardholder fills in the order, selects means of payment, and sends the order and payment information;
- ❑ the merchant requests payment authorization from the cardholder's bank, confirms the transaction by sending a capture request; and sends the cardholder confirmation that the payment has been accepted.

2.3 Overview of the Security Services in SET

Security services are designed to enhance the security of data processing systems and the information transfers of an organization. The services are against security attacks and make use of one or more security mechanisms to provide the services [37].

SET uses a system combining public-key and secret-key cryptography to ensure payment data security. Digital certificates of authenticity provide additional security by enabling participants to identify themselves as legitimate business participants. These certificates are first issued to participants by the financial institution(s) responsible for payment processing systems. SET focuses on maintaining confidentiality of information, ensuring message integrity, and authenticating the parties involved in a transaction. It uses the existing algorithms and self-defined protocols necessary to provide payment transaction security services as follows:

Service for Confidentiality of information

The SET protocol assures cardholders that payment information is safe and can only be accessed by the intended recipients when it travels across the network, preventing interception of important information such as account numbers and expiration dates by unauthorized individuals.

SET provides confidentiality by employing both symmetric and asymmetric data encryption algorithms to protect financial information from eavesdroppers. The message data is initially encrypted by using a randomly generated symmetric encryption key. This key, in turn, is encrypted by using the message recipient's public key. This is referred to as the "digital envelope" of the message and is sent to the recipient along with the encrypted message itself. After receiving the digital envelope, the recipient decrypts it by

using his or her private key to obtain the randomly generated symmetric key and then uses the symmetric key to unlock the original message.

Service for Authentication

SET uses a distinct public/private key pair to create a "digital signature". Authentication is further strengthened by the use of certificates issued by a trusted third party "Certificate Authority (CA)". Within SET, dual signatures are used to link the order message sent to the merchant with the payment instructions containing purchaser's account information sent to the Acquirer.

Cardholders must register with a Certificate Authority before they can send SET messages to merchants. In order to send SET messages to the CA, the cardholder must have a copy of the CA public key-exchange key, which is provided in the CA key-exchange certificate.

❑ *Cardholder account authentication*

A merchant needs a way to verify that a cardholder is a legitimate user with a valid branded payment card account number. A mechanism that uses technology to link a cardholder to a specific payment card account number greatly reduces the incidence of fraud and the overall cost of payment processing. SET provides authentication of a message's origin from the cardholder by using digital signature verification algorithms and cardholder certificate.

❑ *Merchant authentication*

A cardholder needs a way to confirm that a merchant has a relationship with a financial institution allowing it to accept payment cards. Cardholders also need to be able to identify merchants with whom they can securely conduct electronic commerce. SET

provides authentication of a message's origin from the merchant by using digital signature verification algorithms and merchant certificate.

Service for Integrity of data

Important data such as payment information sent from the cardholder to the merchant include order information, personal data, and payment instructions. If any component is altered during transmission, the transaction will not be processed exactly. To eliminate this potential source of fraud and/or error, SET must provide the means to assure that the contents of each order and payment message received matches the contents of the message sent.

SET provides integrity by using one-way cryptographic hashing algorithms and digital signatures to assure that a message was not modified during the transmission between the originator and the recipient. Additionally, SET provides a linkage mechanism for verifying that a message contains a reference to another message via verifying an embedded link by using a one-way cryptographic hashing algorithm, and SET uses specific protocols and message formats to provide interoperability.

2.4 Threats in SET

Although potential security threats are not obviously described in SET standard, they might be there due to their common existence in electronic protocols. The threats can be classified into the following forms:

Masquerade occurs when a communicating entity successfully pretends to be an entity with a different identity.

Modification of Information means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. It may occur to any aspect of a message, such as its contents, attributes, recipient's or originator's identity.

Leakage of Information occurs when information is disclosed to intruders. It may lead to loss of confidentiality, loss of anonymity, misappropriation of messages, and traffic analysis.

Repudiation means a participant to transaction or communication activity later falsely denies that the activity occurred. It occurs when an actual participant denies having originated, submitted or received a message.

Other kinds of threat include Message De-sequencing, Denial of service, Interruption, Impairment of functionality, and Viruses, which are not considered in this thesis.

Chapter 3

SET Payment Process and Transaction

In this chapter, the major payment processes and transactions in SET, including Cardholder Registration, Merchant Registration, Purchase Request, Payment Authorization, and Payment Capture are described. More details in [27][39][40][41].

3.1 Cardholder Registration

Figure 3.1 Cardholder Registration

Any cardholder who wants to participate in SET transactions must register with a Cardholder Certificate Authority (CCA) unless the cardholder already has valid

certificates which can be traversed to the Root CA (see Figure 2.7). Figure 3.1 shows the flow of the cardholder registration process.

1. The registration process starts when the cardholder software sends an initiate request CInitReq to the CCA.
2. After receiving CInitReq, the CCA generates a response CInitRes and signs it by generating a digest of CInitRes and encrypting it with the CCA private signature key. The CInitRes, along with the attached CCA certificates (key-exchange and signature), is sent to the Cardholder software.
3. The certificates are verified by the receiving cardholder software by traversing the trust chain to the Root key according to the Hierarchy of Trust, and the CCA signature is verified by decrypting it with CCA public signature key and comparing the result with a newly generated digest of the CInitRes. If these are correct, one account number is entered by the cardholder. The cardholder software generates registration form request RegFormReq, which is encrypted with a randomly generated symmetric (secret) key N1, N1 is then encrypted along with the card account number with the CCA public key-exchange key. Finally all these are transmitted back to the CCA.
4. The CCA decrypts the symmetric key N1 and the cardholder's account number with CCA's private key of the key-exchange pair. The decoded symmetric key N1 is then used to decrypt the registration form request RegFormReq. The CCA may issue a referral response at this point if RegFormReq does not have the correct form; otherwise find an appropriate registration form RegForm and sign it by generating a

digest of the form and encrypting it with the CCA private signature key. At last, the response RegForm is sent to the cardholder software.

5. The cardholder software first receives the registration form and traverses the trust chain to the Root key to verify the CCA's certificate used to sign the response. The CCA signature is decrypted with the CCA public signature key and compared to a newly computed hash of the registration form. Then, the cardholder software creates a public- private signature key pair, if it has not already done so. The registration form RegForm sent by the CCA is filled in by the cardholder and used to generate a certificate request CertReq. The CertReq, the public signature key of cardholder, and a newly generated symmetric key N2 are combined, and the digest hash of the combined message is created. The hash value is signed by the cardholder software via encrypting it with the cardholder's private signature key. The combined message is encrypted with another newly generated symmetric key N3. N3, along with the cardholder's account information is then encrypted with the CCA's public key-exchange key, and sent back to the CCA.
6. The CCA decrypts with its private key-exchange key to obtain the symmetric encryption key N3 and cardholder's account information, decrypts the certificate request CertReq with N3, and then verifies cardholder signature by decrypting it with the cardholder public signature key and comparing the result with a newly generated digest of the certificate request. If the signature is not verified correctly, processing stops here and the appropriate error message is returned to the cardholder software. Otherwise, the CCA must verify certificate request by using the information in the registration request. This sort of information request / retrieval is not within the scope

of the SET protocol, so here we assume that the CCA and the card's Issuer have a mechanism to deal with this need, but it should be done within a reasonable time because of the online nature of the SET transaction.

Once the information is verified by the Issuer, the CCA may produce a cardholder certificate. The general way of issuing this certificate initiates with the CCA combining a random number with the one the cardholder supplied during the setup process. This produces a secret value only known to the CCA, but based on information from the cardholder. A hash value is derived from the cardholder's account number, expire date, and the secret value. This hash value is then placed in the certificate to show the link between the certificate and the cardholder's information. The CCA signs the cardholder certificate with CCA private signature key and assigns a valid period to that certificate based on the CCA's policy. This period could be the same as the expiration date of the card.

Later, the CCA generates certificate response CertRes and signs it by generating a digest of the response and encrypting it with the CCA private signature key, and then encrypts the CertRes with symmetric key N2 from cardholder request. Finally, this encrypted CertRes is transmitted to the cardholder software.

7. The cardholder software verifies the certificate by traversing the trust chain to the Root key, and then decrypts CertRes with the symmetric key N2 saved during prior processing. The CCA signature is verified by decrypting it with the CCA's public signature key and comparing it with a newly generated hash of the response. At this point, the cardholder software possesses a valid cardholder certificate that has been signed by a CCA and information from the CertRes, and can show them to a similarly

registered and certificated merchant. Thereafter, the certificated cardholder and the certificated merchant can conduct SET transactions between them.

3.2 Merchant Registration

Like a cardholder, any merchant who wants to participate in SET transactions must register with a Merchant Certificate Authority (MCA) unless the merchant already has valid certificates which can be traversed to the Root CA.

The procedure that it goes through closely resembles the one that the cardholder follows, but there are enough differences to warrant a separate description. Figure 3.2 is a diagram of the cardholder registration process.

Figure 3.2 Merchant Registration

1. The merchant software sends an initiation request MInitReq to the Merchant Certificate Authority (MCA).

2. After receiving MInitReq, The MCA issues the appropriate registration form RegForm from the merchant's financial institution and then signs it by generating a digest of the form and encrypting it with the MCA private key. The registration form and the MCA certificates are sent back to the merchant software.
3. The merchant software first verifies the MCA certificates through a traversal of the trust chain to the Root key, then verifies the MCA signature by decrypting it with the MCA public signature key and comparing the result with a newly generated digest of the registration form. Secondly, the merchant software creates two pairs of asymmetric keys: one for key encryption and one for signatures. The merchant completes the enclosed registration form RegForm. Thirdly, the merchant software generates a certificate request CertReq, and creates a combining message with the CertReq, both merchant public keys and signs it by generating a digest of the certificate request and encrypting it with the merchant private signature key. The combining message is encrypted with a randomly generated symmetric key N1. N1, along with the merchant's account information is then encrypted with the MCA public key-exchange key. Finally, all the encrypted messages are transmitted to the MCA.
4. The MCA first decrypts the symmetric key N1 and merchant account information with its private key-exchange key. The symmetric key, N1, is then used to decrypt the merchant's registration request. Secondly, the MCA verifies the merchant signature by decrypting it with the merchant public signature key and then comparing the result with a newly generated digest of the certificate request. If the merchant signature

cannot be verified, all processing stops and an error message is returned to the merchant software.

Now, the MCA must confirm the information in the registration form provided by the merchant software before. Similar to that in cardholder registration, this sort of information request / retrieval is not within the scope of the SET protocol, so we assume here that the MCA can contact the acquirer in some manner and validate the information in the registration form. Anyway, if the verification is successful, the MCA creates the merchant certificates and signs them with the MCA private signature key. A validity period established as a component of these certificates must be provided. It is usually equal to the length of the contract between the merchant and acquirer, but can vary according to MCA policy. These certificates are encrypted with a new symmetric key N_2 , which is itself encrypted with the merchant's public key-exchange key. At last, the entire message CertRes is sent to the merchant software.

5. The merchant software verifies certificates by traversing the trust chain to the Root key, then decrypts the symmetric key with its private key-exchange key and verifies MCA signature by decrypting the certificates with the symmetric key and comparing the result with a newly generated digest of the response. The certificates are validated by the usual traversal, and stored for future use in transactions.

3.3 Purchase Request

The SET protocol is designed to process card payment over the Internet. It doesn't come into play in the shopping experience until after customers (i.e., the cardholders) have completed all the browsing, selecting of goods, and ordering that they are going to

do for the session. Before the actual payment transaction starts, lots of things should be done first. For instance, installment payment should be made for merchandise and the arrangement between merchant and cardholder should be set up before invoking SET. All other ordering functions, including adding on any applicable shipping charges and taxes, must also be completed before the Purchase Request phase begins.

From the viewpoint of cardholder, the Purchase Request is the central phase among the SET transactions. In this phase, the cardholder, merchant, and Payment Gateway are linked either directly or indirectly. Although a seamless flow is presented, it is quite possible that discontinuities could occur. Delays certainly happen since the cardholder needs to fill out information and the merchant has to check his stock in order to fill in the order. The flow of Purchase Request is depicted as Figure 3.3.

Figure 3.3 Purchase Request

1. The cardholder software sends a purchase initiation request PInitReq to the merchant software.
2. After receiving the PInitReq, merchant software generates response PInitRes including a unique transaction identifier TI, and signs PInitRes by generating a

digest of the response and encrypting it with the merchant private signature key. PInitRes, along with the merchant and payment gateway certificates is then sent to the cardholder software.

3. The cardholder software first traverses the trust chain to the Root key to verify the certificates and then saves them for later use. The merchant signature is verified by decrypting it with the merchant public signature key and comparing it with a newly generated digest (hash) of the response. Secondly, the cardholder software creates the Order Information OI, and the Cardholder completes Payment Information PI data structures. After that, the cardholder software generates a dual signature by hashing the concatenation of the digests of OI and the PI and encrypting the dual hash with the cardholder private signature key shown in Fig.3.4. The idea of using the dual signature is to prevent the merchant from reading the account information in the payment instruction and to prevent the acquirer from knowing the detail of the cardholder's order. The PI has been hashed into digest before it is sent to the merchant software who will transfer the digest to the acquirer after the purchase has been approved. The merchant cannot reverse the digest to read the PI. On the other hand, the merchant does not send the OI to the acquirer so that the acquirer cannot read the OI. The PI is then encrypted by a randomly generated symmetric key N1. N1, along with the cardholder's account information, is encrypted with the Payment Gateway's public key-exchange key. At last, cardholder software sends the OI and the encrypted PI along with the dual signature to the merchant software.
4. The merchant software first verifies the cardholder certificate (the cardholder got

Figure 3.4 Dual Signature

it in cardholder registration) by traversing the trust chain to the Root key, verifies cardholder dual signature on the Order Information OI by decrypting it with the cardholder public signature key and comparing the result with a newly generated digest of the concatenation of the digests of the OI and the PI, and then processes the request (including forwarding PI to the payment gateway for authorization, which is described in the next section). Secondly, the merchant software creates the purchase response PurRes including the merchant signature certificate and signs it by generating a digest of the PurRes and encrypting it with the merchant private signature key. Finally, the PurRes is sent to the cardholder software. If the transaction has been authorized, the merchant fills in the cardholder's order.

5. After receiving the PurRes from merchant software, the cardholder software verifies the merchant signature certificate by traversing the trust chain to the Root key, and then verifies the merchant digital signature by decrypting it with the merchant public signature key and comparing the result with a newly generated digest of the purchase response. At last, the cardholder software stores the purchase response as receipt.

3.4 Payment Authorization

During the processing of an order from a cardholder (Section 3.4), the merchant needs to authorize the transaction. The signed authorization request, including the amount to be authorized, the transaction identifier and other information about the transaction, is encrypted and transmitted to the Payment Gateway. At the same time, the signed cardholder payment instructions for this same transaction are encrypted and sent to the Payment Gateway. After successfully verifying the authorization request with the cardholder's Issuer, the Payment Gateway creates, signs, encrypts the authorization response and sends it to the merchant. After processing the verification, the merchant completes processing of purchase request. At this point, the Payment Authorization process is done. Figure 3.5 depicts the general process of the Payment Authorization.

Figure 3.5 Payment Authorization

1. The merchant software creates an authorization request **AuthReq** and signs it by generating a digest of the authorization request and encrypting it with the merchant private signature key. The **AuthReq** is then encrypted with a randomly generated symmetric key **N2**, which is in turn encrypted by the payment gateway's public key-

exchange key. The encrypted AuthReq including the encrypted PI from the cardholder is then sent to the Payment Gateway.

2. The Payment Gateway first verifies the merchant certificates by traversing the trust chain to the Root key, and decrypts the authorization request's envelope with its private key-exchange key to get the symmetric key N2. N2 is used to decrypt the actual authorization request. Secondly, the Payment Gateway verifies the merchant digital signature by decrypting it with the merchant public signature key and comparing the result with a newly generated digest of the authorization request. Thirdly, the Payment Gateway verifies the enclosed cardholder certificates by traversing the trust chain to the Root key, decrypts the first symmetric key N1 and the cardholder's account information with the Payment Gateway's private key-exchange key. The PI block is decrypted with N1. The dual signature on the PI is then verified by decrypting it with the cardholder's public signature key and comparing the result with a newly generated digest (hash) of the concatenation of the OI and the PI hashes. To ensure consistency between the merchant's authorization request and the cardholder's PI, the Payment Gateway verifies that the transaction identifier from the merchant software matches the one in the PI from the cardholder software. If it does, an authorization request is generated and sent to the Issuer via a financial network. How this authorization request works is beyond the scope of SET. We assume there is a mechanism between the Payment Gateway and the Issuer that simply returns a "go / no go" to the transaction flow. There may be a time discontinuity here if the Issuer does not immediately respond to the authorization request.

Later, the Payment Gateway creates authorization response AuthRes and signs it by generating a digest of the AuthRes and encrypting it with the Payment Gateway private signature key. The AuthRes is then encrypted with a newly randomly generated symmetric key N3. N3 is encrypted with the merchant's public key-exchange key. The Payment Gateway creates a capture token and signs it by generating a digest of the capture token and encrypting it with the Payment Gateway private signature key, and then encrypts capture token with a new randomly generated symmetric key N4. N4 and the cardholder account information is then encrypted with the Payment Gateway public key-exchange key. At last, the Payment Gateway sends the encrypted response to the merchant software.

- 3 Upon receipt, the merchant software verifies the Payment Gateway's certificates by traversing the trust chain to the Root key, decrypts symmetric key N3 with its private key-exchange key, and then decrypts the authorization response with N3. The Payment Gateway's signature is verified by decrypting it with the Payment Gateway's public signature key and comparing it to a newly generated digest (hash) of the authorization response. The merchant saves the encrypted capture token (if present) for later capture processing.

3.5 Payment Capture

After completing the processing of an order from a cardholder (Section 3.4), the merchant wants to be paid. The signed capture request from the merchant, including the final amount of the transaction, the transaction identifier, and other information about the

transaction, is encrypted and sent to the Payment Gateway. After successfully verifying the capture request, the Payment Gateway sends the related information to the Issuer, then generates, signs, encrypts capture response and sends it to the merchant. After processing the verification, the merchant completes processing of capture request. At this point, the Payment Capture process is done.

Note: The SET is flexible to allow both single capture request and multiple capture requests in Payment Capture. Figure 3.6 depicts the general process of the Payment Capture.

Figure 3.6 Payment Capture

1. The merchant software creates the Capture Request CapReq, including the final transaction amount, the transaction identifier from the OI, and other information, and imbeds the merchant certificates in capture request and signs it by generating a digest of the capture request and encrypting it with the merchant private signature key. The CapReq is encrypted with a randomly generated symmetric key N5. N5 is then further encrypted using the Payment Gateway's public key-exchange key. The CapReq along with the encrypted capture token from the authorization response is sent to the Payment Gateway.

2. The Payment Gateway first verifies the merchant's certificates by traversing the trust chain to the Root key, and decrypts symmetric key N5 with the Payment Gateway's private key-exchange key. The capture request can be decrypted with N5. Secondly, the Payment Gateway verifies the merchant digital signature by decrypting it with the merchant public signature key and comparing the result with a newly generated digest of the capture request, decrypts symmetric key N4 with the Payment Gateway private key-exchange key, and then decrypts the capture token with N4.

The Payment Gateway must ensure consistency between merchant's capture request and the capture token. The Payment Gateway sends capture request via a financial network to the Issuer, which is not within the scope of SET.

Later, the Payment Gateway creates a Capture Response CapRes, including a Payment Gateway signature certificate, and signs it by generating a digest of the capture response and encrypting it with the Payment gateway private signature key. The CapRes is encrypted by the Payment Gateway with a new randomly generated symmetric key N6. N6 is then encrypted with the merchant's public key-exchange key. At last the encrypted CapRes is sent to the merchant software.

3. The merchant software first verifies the Payment Gateway's certificate by traversing the trust chain to the Root key, and decrypts N6 with its private key-exchange key. N6 is then used to decrypt the capture response CapRes. Secondly, the merchant software verifies the Payment Gateway's signature by decrypting it with the Payment Gateway public signature key and comparing the result with a newly generated digest of the capture response. Finally, CapRes is stored by the merchant software for any further reconciliation steps with the acquirer.

Chapter 4

Predicate Transition Net Approach and PROD

Realistic protocols, and the programs that implement them, are often quite complicated. Consequently, much research has been done trying to find formal, mathematical techniques for specifying and verifying protocols [42]. There are two key concepts used in many protocol models: one is the finite state machine, the other is the Petri Net. High-level Petri Nets, such as Predicate Transition Net [Section 4.1], are a powerful net type by which it is possible to handle rather complex systems in a succinct and manageable way. They are widely used in system modeling and validation due to the expressiveness of powerful description languages and specific analysis tools based on these. PROD [Section 4.2] is a reachability analysis tool for Predicate Transition Net approach.

4.1 Predicate Transition Net Approach

Net theory is a system theory that aims at an understanding of systems whose structure and behavior are determined by the distributedness and combinatorial nature of their states and changes. It studies such systems at different conceptual levels, in various degrees of detail, and in many areas (both computer science and other fields) of

application. One important research area in net theory is concerned with the conceptual and mathematical foundation of an adequate notion of dynamic system along with different modeling ways of dynamic system.

Petri net theory is a uniform mathematical formalization of discrete-event systems that have the nature of concurrency and nondeterminism. It abstracts both the structural and the behavioral features of the systems, and thus helps derive their static and dynamic properties, evaluate the system, and provide improvement suggestions. The basic net theoretical system model is called “condition/event systems”, proposed by C. A. Petri, as the common reference model of net theory. Petri nets as a net-like mathematical tool for the study of communication with automata, define a dynamic system which has an adequate representation in a net theoretical model.

Models with Petri nets allow analysis of properties such as reachability (a state can be reached from a given initial state), boundedness (the maximum number of tokens in each place is bounded), and liveness (a potential firing is always available to move a token from a given place). In general, such analysis would suffer from the state-explosion problem (the state space of the system can be far too large with respect to the time and other resources needed to inspect all states in the space), but extensions such as Predicate Transition Nets can reduce the space size.

Recently, many of the theoretical and practical work in the area of high-level Petri nets use Predicate Transition Nets. The main reason for the success of this kind of nets is that they make it possible to obtain much more succinct and manageable descriptions than can be obtained from low-level Petri nets. In addition, they also provide a wide range of analysis methods and tools. The step from low-level nets to high-level nets can

be compared to that from assembly languages to advanced programming languages with an elaborated type concept. In low-level nets, there is only one kind of token, which means that the state of a place can be simply described by an integer and even by a boolean in many cases. In high-level nets, each token can carry more complex information / data which may describe the entire state of a process or a database [24].

Predicate Transition Nets are defined to use the notation and concepts of many-sorted algebras — as known from abstract data types. The key aim of developing Predicate Transition Nets is to introduce the concept of individuals with changing properties and relations into net theory.

- **Predicate/Transition Nets**

Predicate/Transition Nets (Pr/T Nets) as High-Level Petri Nets in general that describe distributed systems as first-order dynamic structures were first introduced by H. J. Genrich and K. Lautenbach.

As an example of a major extension of Petri nets, a Predicate Transition Net permits specification of different types representing a module definition's various behaviors. It enhances the modeling and analyzing power of Petri net and offers a formal treatment of individuals and their relations for dynamic systems [14]. Predicate Transition Nets are schemes of system models with n-tuples of expressions (including variables) as arc inscriptions and as invariant entries. Due to the use of variables, formal expressions, products and sums, they are essentially a syntax-based formalism.

A Pr/T Net expresses a distributed system in terms of a first-order language consisting of function symbols and relation symbols in which some of the relation symbols have a dynamic interpretation (relative to a fixed domain of individuals). The

places of the net represent the relations whose extensions change dynamically. Consequently, they are marked by tokens consisting of tuples of individuals to denote the “current” interpretations of the associated relation symbols. The interpretations capture the distributed states of the system in a natural way. The transitions of the net correspond to the schemes of elementary changes (events). One occurrence of an event changes the “old” interpretations of the relation symbols that are in its range into the “new” interpretations. This effect is presented via token patterns annotating the flow arcs impinging upon a transition. An event proper is created by binding the individual variables in the neighboring token patterns so that a “guard” associated with the transition is satisfied. Such guards consist of function symbols and relation symbols that have a static (fixed) interpretation over the domain of individuals [11].

In the following, Pr/T Nets can be defined as formal objects that can be interpreted and manipulated in a mathematical way that is comparable to working with logical formulae of algebraic expressions. The approach is to model a dynamic system in terms of a set of individuals that is structured by functions and relations. The structure is partially static and partially dynamic.

First of all, the model of the Pr/T Nets uses first-order structures as its formal basis for the representation of discrete dynamic system. A first-order structure is a tuple of objects, $R = (D; f_1, \dots, f_k; R_1, \dots, R_n)$, where D is a non-empty set of individuals called the domain (or universe) of R , the f_i s are basic functions (operations) in D and the R_j s are various relations in D . A dynamic structure is characterized by the fact that some relations are variable in the sense that their extensions may vary from state to state due to the occurrence of processes in the modeled system.

Secondly, the static and the variable parts are separated from each other. The static part remains an ordinary relational structure. It is the support of the dynamic system. The dynamics is represented as an annotated net, called Predicate/Transition Net. The variable relations appear as the places of the net. The function symbols and relation symbols of the static part, or more precisely, i.e., operators and predicates respectively, appear in the annotations. In other words, the annotations of the net are interpreted in terms of a given static relational structure, the support. This allows to derive from the annotated net the condition/event system that is the net theoretical expression for the dynamic structure. The presentation of dynamic structures requires that predicates denoting static relations and predicates denoting variable relations be distinguished.

- The Basic Form of Pr/T nets

For a given first-order language L , there is the class of strict predicate/transition nets with annotations in L . Strict means that multiple occurrences of tuples are not allowed on the places. The places represent variable relations, not multi-relations.

Let L be a first-order language and let L_s designated the sublanguage using only Π_s , the predicates denoting static relations. The class Pr/T_L is composed of marked annotated nets, in other words, a Predicate Transition Net is a marked annotated net $\text{Pr/TN} = (N, A, M)$, where N is the underlying net, A is its annotation in L , and M is the marking.

◆ a directed net $N = (P, T; F)$ where

P is the set of nodes — predicates (‘first-order’ places) O

T is the set of nodes — transitions (‘first-order’ transitions) $\square$

$F \subseteq P \times T \cup T \times P$ is the set of input / output arcs (flow relations) $\rightarrow$

- ◆ four types of annotation of N , $A = (A_N, A_P, A_T, A_F)$
 - A_N is a first-order structure for L_s called the support of Pr/TN. It contains a finite set of constants U , a set of variables V (ranging over U), functions F_i in U , and relations R_i in U . Describing static aspects, it annotates the entire net rather than specific elements.
 - A_P is a bijection between the set of places P and a set of variable predicates Π_v .
 - A_T is a mapping of the set of transitions T into the set of formulae (called transition selectors) that use only static predicates (in L_s) and operators.
 - A_F is an arc-labeling function that associates formal tuple sums of variables and constants to the arcs. Hence the mapping is defined as $A_F \rightarrow W \cup W^2 \dots \cup W^n$, where $W = U \cup V$ and W^k denotes the set of all k -tuples in W .
- ◆ the marking M of predicates P represents the displacement of tokens in the net. Pr/TN tokens consist of separate individuals (items) which differ from ordinary Petri nets, where tokens are indistinguishable. In ordinary nets, the numbers of token within places are of interest, whereas in Pr/TNs, the types of tokens within places are important. In this way, token sets are formed in Pr/TNs, which are attached to places and transitions of the net. If $x \in P \cup T$, and the arity (number of arguments) of x is k , a token set of x can be defined as $C(x) = U^k$. A given token $c = (u_1, \dots, u_k)$ in a place $p \in P$ says a predicate of arity k , corresponding to the place, is true for a given instantiation of the argument tuple contained in the token. Similarly, for a token c in a transition $t \in T$, the variables in the annotations specified for the transition and its incident arcs are replaced by the corresponding constants appearing in the token [43].

- Modeling a system by using Pr/T nets has many advantages:
 - ♦ the overall behavior is easier to understand due to the graphical and precise nature of the representation scheme;
 - ♦ it can be interactively executed and analyzed by the use of the Petri net and Pr/T theories;
 - ♦ it can be synthesized using bottom-up and top-down approaches;
 - ♦ it can be further investigated by Pr/T tools such as PROD.

More detailed theory and modeling about Petri Nets and Predicate Transition Nets can be found in [12][13][24][30][31][35].

4.2 Pr/T-Net Reachability Analysis Tool PROD

Reachability analysis is a powerful formal way to analyze (including to detect errors) concurrent and distributed systems with a finite state space such as telecommunication protocols.

Reachability analysis is based on the generation of the complete state space, thus it greatly suffers from the state-explosion problem as the state space of the system can be far too large with respect to the time and other resources needed to inspect all states in the space.

Fortunately, errors can be detected in various cases without inspecting all reachable states of the system. The stubborn set method [44][45] is one of the efficient methods that try to relieve the state space explosion. On the other hand in on-the-fly verification of properties, the properties are verified during state space generation, in

contrast to the traditional approach where properties are verified after state space generation.

The Pr/T-Net reachability analyzer PROD [17], developed at the Digital Systems Laboratory, Helsinki University of Technology, allows to automatically generate the whole finite reachability graph or to reduce it by supporting on-the-fly verification of linear time temporal properties with the aid of the stubborn set method and verification of branching time temporal properties. The basic idea of reachability graph is to construct a graph which contains a node for each reachable state and an arc for each possible change of state. The graph can be used to prove properties about the modeled system. For bounded systems a number of questions can be answered. Deadlocks, mutual exclusion, reachability and marking bounds can be decided by a simple search through the nodes of the reachability graph, while liveness and home markings can be decided by constructing and inspecting the strongly connected components. PROD also has a rich query language to inspect the generated state space.

The PROD consists of five subprograms:

- prod: a batch program, hides some environment-specific commands that are needed in using the tool;
- prpp: a Net Description Language preprocessor, reads a net description in the Net Description Language, and generates a C program (called the reachability graph generator) which, when compiled and linked, produces a reachability generator. Then the reachability generator is executed to create the reachability graph;
- strong: a tool for calculating strongly connected components of the reachability

graph;

- araproduct: an interface to ARA (a LOTOS reachability analysis tool developed in Technical Research Center of Finland);
- probe: a graph query program for navigating / inspecting the reachability graph.

In general, the query program probe is an interactive program though it is possible to run it in a batch file. There are many commands within probe, here we briefly describe some important ones.

In the reachability graph, a node corresponds to a reachable marking of the net, an arrow corresponds to a fired transition instance. The set of immediate successor arrows of a node depends upon the graph generation method, by default, it corresponds to all enabled transition instances.

After probe is executed, the number of the current node of the graph is displayed at a command prompt. For example, the command prompt 0# means the current node of the graph is node number 0, which always represents the initial marking of the net. The command “statistics” is used to show the general overview of the graph, including how many nodes, arrows, connected components exist in the graph. The command “showends all” is used to display more detailed information about all nodes in the graph. The command “sets” is used to display a list of the reviewable sets. The command “quit” is used to quit probe program and then return control to the calling program.

How PROD works (on UNIX)

In general, the following steps are needed.

- write a program called `sample.net` in PROD Net Description Language.
- `prod sample.init` — executes `prpp` on the net file `sample.net`, `prpp` generates a C-file `sample.c` along with some other data files, which are then compiled and linked by C-compiler to an executable `sample` (the proper reachability graph generator).
- `sample` — generates the full reachability graph.
- `strong sample` — computes the strongly connected components of the graph.
- `probe sample` — inspects the graph

The Net Description Language

PROD's Net Description Language is the C preprocessor language extended with net description directives. Here are two key issues, others in [47].

- Places **#place** place-name [**lo**(lower-limit)] [**hi** (upper-limit)] [**mk** (initial-marking)]
- Transitions **#trans** transition-name [**fd** (list)]
 [**in** {place-name : input-expression; ... }]
 [**out** {place-name : output-expression; ... }]
 [**gate** gate-expression;]
 [**comp** C-block]
 #endtr

PROD supports the reduced reachability graph generation methods as follows:

- The stubborn set method is based on the idea that it is not necessary to generate all possible interleavings of independent transitions for the verification of certain properties in the net. A stubborn set is comprised of a set of transitions relative to the marking. Reachability graph generation using the stubborn set methods proceeds by computing a stubborn set at each marking and firing only enabled transitions within

this set. In PROD, the calculation of the stubborn set can be done in the following two algorithms:

- ◆ The incremental algorithm (option `-s`) first selects one enabled transition and then looks for other transitions to be included so that the stubbornness property is retained. The selected transition can be the first one found (default) or any randomly selected one (option `-x`). Normally the incremental algorithm computes the smallest stubborn set at the marking. With option `-y` the first stubborn set found is selected.
- ◆ The deletion algorithm (option `-d`) starts with the set of all transitions and then looks for other transitions to be deleted so that the stubbornness property is retained. The selected transition can be the first one found (default) or any randomly selected one (option `-z`).
- The CFFD preserving stubborn set method (option `-c`) is only for the stubborn set that preserves the chaos-free, failures divergences properties [44].
- Sleep sets (option `-s`) are an alternative method by using the independence of transitions to reduce the reachability graph. Sometimes sleep sets used in conjunction with stubborn set can increase the reduction of reachability graph.
- Symmetries (option `-e`) are equivalent classes of markings. The computation of symmetries can be very time-consuming, but it is well known that symmetries in conjunction with stubborn set can radically reduce the reachability graph.

PROD also has a branching time temporal logic CTL model checker, which is implemented as a part of the probe reachability graph navigator tool [14].

On-the-fly verification

On-the-fly verification of a property means that the properties are verified during state space generation, in contrary to the traditional approach where the properties are verified after state space generation.

CFFD-equivalence [44] is a behavioral equivalence with a close relationship to linear time temporal logic. It can be defined in full or reduced reachability graphs of a given net with a given set of visible transition instances. A CFFD-preserving reduced reachability graph, i.e., a graph that is CFFD-equivalent to the full reachability graph, can be created by using the stubborn set method [44][46].

PROD Applications

PROD has been freely downloaded to at least hundreds of sites worldwide since its invention. At the Digital System laboratory, Helsinki University of Technology, PROD has been used in several academic and industrial projects, which include being a teaching aid in a course on Parallel and Distributed systems, Synchronized Ring (FSR) — a high speed parallel data bus designed for high-throughput applications like ATM-switches. Other substantial models have been built and analyzed with PROD including the TCAP/TSL-protocol, a video on demand system [33][34][42], an authentication protocol [3] and a specification of a simple telephone exchange (YXA).

Chapter 5

Simulation and Verification

The SET protocol could be described as an intricate dance between the following parties: the cardholder, the merchant, CCA(Cardholder Certificate Authority), MCA (Merchant Certificate Authority), Payment Gateway. Petri Nets and Predicate Transition Nets have been widely used in modeling and validation of communication protocols [4][7][24][28][32]. As we said in Chapter 4, Predicate Transition Nets can handle complex systems in a succinct and manageable way. The characteristic feature of Pr/T Nets is that tokens represent individual data items. The use of structured tokens permits concise net representations of the complex SET system. In addition, the expressiveness of powerful description language (such as net description language) and specific analysis tools (such as PROD) based on Pr/T Nets can be used to simulate and analyze the process of the SET protocol. In SET, once one cardholder or one merchant finishes his registration, he can use the SET payment protocol until his registration certificate is expired. Therefore, we must consider the entire payment transaction (five parts) as three parts: Cardholder Registration, Merchant Registration and payment process. In the following simulations, the workflow of the Cardholder Registration and Merchant Registration processes and the workflow of the payment protocol process [Chapter 3] are modeled by Predicate Transition Nets, and all these models are simulated and analyzed

with PROD. At a later stage, four key threats at the application level are considered by modeling with Pr/T Nets and simulating with PROD respectively for each key transaction in order to verify the security of the SET protocol. Finally, the results of the verifications are summed up.

5.1 SET Payment Transaction

5.1.1 Cardholder Registration

Cardholder Registration Process is shown in Figure 3.1 according to the specification in Section 3.1. There are two parties in this process: the Cardholder (Software) and the Cardholder Certificate Authority. The order of the workflow is from the top to the bottom. To make the simulation simpler, some detailed variables may be substituted by only one variable. For example, account information may include account number, cardholder's identification and valid period. In this case, only CAcctInfo is used instead of these three variables. In fact, if any one or all of these is changed, CAcctInfo is changed, and thus the results are the same. In other words, the data items in our simulation are enough to represent the transaction process. All variables in the complete process are listed in Table 5.1.

Table 5.1 Variables in the Complete Process

Symbol	Variable	Meaning
x_1	CInitReq	Cardholder Software's Initial Request for registration
x_2	CInitRes	response (from CA) to the CInitReq
y_2	h_CInitRes	hash digest of CInitRes

z ₂	CAC	Cardholder CA's Certificate
x ₃	CACctNo	Cardholder's Account Number
y ₃	RegFormReq	Registration Form Request from Cardholder
z ₃	N1	random number one for Cardholder Registration
x ₄	RegForm	response (from CA) to the RegFormReq
y ₄	h_RegForm	hash digest for RegForm
u ₅	CPuK	Cardholder's Public Key
v ₅	CCertReq	Certificate Request from Cardholder
w ₅	N2	random number two for Cardholder Registration
x ₅	h_CCertReq_CPuK_N2	hash digest for CCertReq, CPuK and N2
y ₅	CACctInfo	Cardholder's Account Information
z ₅	N3	random number three for Cardholder Registration
x ₆	CCertRes	Certificate Response to CcertReq
y ₆	h_CCertRes	hash digest of CCertRes
x ₇	MInitReq	Merchant Software's Initial Request for registration
x ₈	RegForm	Registration Form from MCA
y ₈	h_RegForm	hash digest of RegForm
z ₈	CAC	Merchant CA's Certificate
u ₉	MCertReq	Merchant's Certificate Request to MCA
v ₉	MPuK	Merchant's Public Key
w ₉	MPuSK	Merchant's Public Signature Key
x ₉	h_MCertReq	hash digest of MCertReq
y ₉	N1	random number one for Merchant Registration
z ₉	MAcctInfo	Merchant's Account Information
x ₁₀	h_MCert	hash digest of MCert
y ₁₀	h_MCertRes	hash digest of MCertRes
z ₁₀	N2	random number two for Merchant Registration
x ₁₁	PInitReq	Purchase Initiate Request
w ₁₂	PInitRes	Response to Purchase Initiate Request
x ₁₂	h_PInitRes	hash digest of PInitRes

$y_{12} z_{14} v_{16} z_{20}$	MCert	Merchant's Certificate
$z_{12} z_{17} z_{21}$	PGCert	Payment Gateway Certificate
u_{13}	$h_h_OI_h_PI$	dual hash, digest of h_OI and h_PI
v_{13}	OI	Order Information
$w_{13} w_{16}$	h_PI	hash digest of Payment Information (PI)
$x_{13} x_{16}$	N1	random number one for PR-PA-PC
$y_{13} y_{16} y_{17}$	CACctInfo	Cardholder's Account Information
$z_{13} z_{16}$	CCert	Cardholder Certificate
y_{14}	h_PurRes	hash digest of PurRes
z_{15}	AuthReq	Authorization Request
t_{16}	$h_AuthReq$	hash digest of AuthReq
u_{16}	N2	random number two for PR-PA-PC
s_{17}	$h_AuthRes$	hash digest of AuthRes
t_{17}	AuthRes	Authorization Response
u_{17}	N3	random number three for PR-PA-PC
v_{17}	h_CapTok	hash digest of CapTok
w_{17}	CapTok	Capture Token
$x_{17} y_{20}$	N4	random number four for PR-PA-PC
y_{19}	CapReq	Capture Request
w_{20}	h_CapReq_MCert	hash digest of CapReq and MCert
x_{20}	N5	random number five for PR-PA-PC
w_{21}	h_CapRes_MCert	hash digest of CapRes and MCert
x_{21}	CapRes	Capture Response
y_{21}	N6	random number six for PR-PA-PC

Figure 5.1 Workflow of Cardholder Registration

Figure 5.1 depicts the workflow of Cardholder Registration. In our simulation, the Cardholder Registration process includes seven places, six transitions and six transmissions. During any one transition, security mechanisms are applied as described in the specification of the SET protocol. Figure 5.2 shows a diagram of the Pr/T Net for the Cardholder Registration process. The variables are in Table 5.1. From the Pr/T net's point of view, in terms of hiding of details in a consistent way, the token in one place has its data structure which is different from that in any one of other places and it represents the actual information at the moment. The token may include plaintext items, ciphertext items (important data), signatures (by RSA and hash value). For example, in the following PROD program, place p4 has two plaintext items (RegForm, CAC), three ciphertext items (CAcctNo, RegFormReq, N1), and one signature (h_RegForm).

Figure 5.2 a diagram of the Pr/T Net for the Cardholder Registration process

In the following PROD program that is the key part for Cardholder Registration process, some macros from the C library are used to simulate different encryption / decryption processes. The macro `toascii('X')` defined in preprocessor is supposed to represent an encryption process by a certain key, for example, `toascii('a')` is specifically defined to represent `encCAPrK(h_InitRes)`.

```
#place p1 hi(<.CInitReq.>) lo(<.0.>) mk(<.CInitReq.>)
#place p2 hi(<.CInitRes, h_CInitRes, CAC.>) lo(<. 0 , 0 , 0 .>)
#place p3 hi(<.h_CInitRes, CAcctNo, RegFormReq, N1.>) lo(<. 0 , 0 , 0 , 0 .>)
#place p4 hi(<.CAcctNo, RegFormReq, N1, RegForm, h_RegForm, CAC.>) \
    lo(<. 0 , 0 , 0 , 0 , 0 , 0 .>)
#place p5 hi(<.RegForm, h_RegForm, CPuK, CCertReq, N2, h_CCertReq_CPuK_N2, \
    CAccInfo, N3.>) lo(<. 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 .>)
#place p6 hi(<.CPuK, RegForm, N2, h_CCertReq_bPuK_N2, CAcctInfo, N3, CCertRes, \
    h_CCertRes, CAC.>) lo(<. 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 .>)
```

```

#place p7 hi(<.CCertRes, h_CCertRes.>) lo(<. 0 , 0 .>)

/* CA gets CInitReq, produces CInitRes, signs it and encrypts, concatenates CAC, and sends to
Cardholder Software ( CS ) */
#trans t1
  in {p1: <.CInitReq.>;}
  out{p2: <.CInitRes, encCAPrK(h_CInitRes), CAC.>;}
#endtr

/* CS checks CInitRes & CAC, decrypts & checks the digest. If they are all right, Cardholder fills
in CAcctNo; CS creates RegFormReq and encrypts it with N1, then encrypts CAcctNo and N1
with CAPuK, and sends to CA */
#trans t2
  in {p2: <.a, b, c.>;}
  out{p3: <.i, encCAPuK(CAcctNo), encN1(RegFormReq), encCAPuK(N1).>;}
  gate b == toascii('a');
  comp { i = decCAPuK(b); Accept();}
#endtr

/* CA decrypts CAcctNo and N1 with CAPrK, decrypts RegFormReq with N1 and checks it, then
creates RegForm and encrypts it with CAPrK .and sends it along with CAC to CS */
#trans t3
  in {p3: <.a, b, c, d.>;}
  out{p4: <.i, j, k, RegForm, encCAPrK(h_RegForm), CAC.>;}
  gate (b == toascii('A')) && (c == toascii('1')) && (d == toascii('B'));
  comp {(i=decCAPrK(b)) && (j=decN1(c)) && (k=decCAPrK(d)); Accept();}
#endtr

/* CS checks RegForm, decrypts & checks the digest, and creates one (pair) keys; takes off
a,b,c,f, creates CCertReq, Cardholder fills in CAcctInfo, CS generates N2 and signs CCertReq,
CPuK and N2 and encrypts it with CPrK, encrypts CAcctInfo with CAPuK, encrypts N2 with N3,
encrypts CPuK, CCertReq and N2 with N3, and encrypts N3 with CAPuK. at last send all these
stuff to CA */
#trans t4
  in {p4: <.a, b, c, d, e, f.>;}
  out{p5: <.d, i, encN3(CPuK), encN3(CCertReq), encN3(N2), \

```

```

        encCPrK(h_CCertReq_CPuK_N2), encCAPuK(CAcctInfo),encCAPuK(N3).>};
gate e == toascii('b');
comp { i = decCAPuK(e); Accept();}
#endtr
/* CA decrypts CAcctInfo and N3 with CAPrK, decrypts CPuK, CCertReq and N2 with N3,
decrypts h_CCertReq_CpuK_N2 with CAPuK and checks this digest, verifies CCertReq; and
produces CCertRes, signs it and ecrypts with CAPrK, encrypts CCertRes and attaches CAC , then
sends to CS */
#trans t5
in {p5: <.a, b, c, d, e, f, g, h.>;}
out{p6: <.i, j, k, l, m, n,encN2(CCertRes), encCAPrK(h_CCertRes), CAC.>;}
gate (c == toascii('3')) && (d == toascii('4')) && (e == toascii('5')) \
    && (f==toascii('E')) && (g==toascii('C')) && (h==toascii('D'));
comp { (i=decN3(c)) && (j=decN3(d)) && (k=decN3(e)) && (l=decbPuK(f)) \
    && (m = decCAPrK(g)) && (n = decCAPrK(h)); Accept();}
#endtr
/* CS checks & takes off a,b,c,d,e,f,CAC; decrypts CCertRes with N2, and decrypts with CAPrK
and checks the digest, and holds the CCertRes which includes CCert */
#trans t6
in {p6: <.a, b, c, d, e, f, g, h, i.>;}
out{p7: <.m, n.>;}
gate (g == toascii('2')) && (h == toascii('c'));
comp { (m = decN2(g)) && (n = decCAPuK(h)); Accept();}
#endtr

```

The above program and its related preprocessors are executed as explained in Section 4.2 and the results are obtained as follows. The numbers are used to signify the encryption values of different macros, for example, 97 is the value of toascii('a').

Node 0
p1: <.CInitReq.>

```

Node 1
p2: <.CInitRes,97,CAC.>
-----
Node 2
p3: <.h_CInitRes,65,49,66.>
-----
Node 3
p4: <.CAcctNo,RegFormReq,N1,RegForm,98,CAC.>
-----
Node 4
p5: <.RegForm,h_RegForm,51,52,53,69,67,68.>
-----
Node 5
p6: <.CPuK,CCertReq,N2,h_CCertReq_CPuK_N2,CAcctInfo,N3,50,99,CAC.>
-----
Node 6
p7: <.CCertRes,h_CCertRes.>
-----
7 nodes (all)

```

The result gives out a general description of the entire procedure of Cardholder Registration. From the beginning (p1) where cardholder software initiates the registration request (CInitReq) to CCA, to the end (p7) where the cardholder software gets the CCert from the CCertRes, there are six transmissions (i.e., 3-way handshake transactions) in Cardholder Registration process. The result shows that all the Pr/T Net model transactions of the process can successfully fire. This is the normal situation without any threats. After successful registration, the Cardholder can process the SET payment transactions later.

NOTE TO USERS

Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.

66-86

This reproduction is the best copy available.

UMI

occur. This means that the intruder can use the participant's important data such as account number. Fortunately, this case would not happen because the masquerade for any one of CAcctNo/MAcctInfo/CAcctInfo/CapTok means anything within the item has to be changed. In other words, if there is masquerade for any one of CAcctNo/MacctInfo/CAcctInfo/CapTok, the whole data structure for that item should be "masqueraded". This case can be categorized to the Modification of Information as follows.

5.2.2 Modification of Information

Modification of Information means that original information can be interrupted and modified by the intruder. In our simulation, the solid-line ellipse represents the normal situation, while the dot-line ellipse represents the MOI situation. Here, in Figure 5.8, we only present part of the complete process, i.e., MOI in Cardholder Registration. The similar remaining parts of the complete process are omitted but the whole results of the complete process are given. To make the modeling simpler, we assume that in each case, each time the MOI may occur in only one of the transmissions although all six cases are shown together in Figure 5.8. We simulate the MOI by appending X to any "perilous" items and then analyzing the resulting net as before.

We know the PROD result has six transactions, six transmissions and 7 nodes between A (Cardholder Software) and B (CCA) for the normal situation of Cardholder Registration. If the PROD result shows more transactions / transmissions / nodes than those of the normal situation, we can determine one or more attacks occur. This kind of deduction is also applied to the complete process. Since SET provides a hierarchy of

Certificate Authorities (CAs) (see Chapter two), and the certificate verifications can be easily done using only PROD, not all CA certificate verifications in the following are analyzed by Logic.

- **Transmission one($A \Rightarrow B$):** the MOI can successfully occur.

Analysis: CInitReq is a plaintext of data which can be changed and cannot be discovered.

- **Transmission two($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : (CInitResX, [h_CInitRes]K_B^{-1}, CACX)$, A sees $[h_CInitRes]K_B^{-1}$, then A believes B said $h_CInitRes$; and A sees CInitResX and thus knows the data is changed.

- **Transmission three($A \Rightarrow B$):** the MOI cannot occur.

Analysis: if $A \rightarrow B : ([CacctNo]K_B, [RegFormReq]N1, [N1]K_B)$, only B sees CAcctNo, $[RegFormReq]N1$ and $N1$, and thus RegFormReq, and only B can change CAcctNo.

- **Transmission four($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : (RegFormX, [h_RegForm]K_B^{-1}, CACX)$, A sees $[h_RegForm]K_B^{-1}$, then A believes B said $h_CInitRes$; and A sees RegFormX and thus knows the data is changed.

- **Transmission five($A \Rightarrow B$):** the MOI cannot occur.

Analysis: if $A \rightarrow B : ([CPuK]N3, [CCertReq]N3, [N2]N3, [h_CCertReq_CPuK_N2]K_A^{-1}, [CacctInfo]K_B, [N3]K_B)$, only B sees $([CPuK]N3, [CCertReq]N3, [N2]N3, [h_CCertReq_CPuK_N2]K_A^{-1}, CacctInfo, N3)$, and thus only B can change CAcctInfo.

- **Transmission six($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : ([CCertRes]N2, [h_CCertRes]K_B^{-1}, CAC)$, only A and B can change CCertRes and only B can change $h_CCertRes$.

Figure 5.8 Modification of Information in Cardholder Registration

Similar situation is for the remaining part of the complete process, so we do not present the workflow but only give out the result and analysis for the remaining part of the complete process. Transmissions seven to ten are between A (Merchant Software) and B (Merchant CA).

- **Transmission seven($A \Rightarrow B$):** the MOI can successfully occur.

Analysis: MInitReq is a plaintext of data which can be changed and cannot be discovered.

- **Transmission eight($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A$: (RegFormX, [h_RegForm] K_B^{-1} , CACX), A sees [h_RegForm] K_B^{-1} , then A believes B said h_CInitRes; and A sees RegFormX and thus knows the data is changed.

- **Transmission nine($A \Rightarrow B$):** the MOI cannot occur.

Analysis: if $A \rightarrow B$: ([MCertReq]N1, [MPuK]N1, [MPuSK]N1, [h_MCertReq] K_A^{-1} , [N1] K_B , [MAcctInfo] K_B), only B can change MAcctInfo.

- **Transmission ten($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A$: ([MCertRes]N2, [h_MCertRes] K_B^{-1} , [N2] K_A), only B can change h_MCertRes.

Transmissions eleven to fourteen are between A (Cardholder Software) and B (Merchant Software).

- **Transmission eleven($A \Rightarrow B$):** the MOI can successfully occur.

Analysis: PInitReq is a plaintext of data which can be changed and cannot be discovered.

- **Transmission twelve($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : (PInitResX, [h_PInitRes]K_B^{-1}, MCertX, PGCertX)$, A sees $[h_PInitRes]K_B^{-1}$, then A believes B said $h_PInitRes$; and A sees $PInitResX$ and thus knows $PInitRes$ is changed.

- **Transmission thirteen ($A \Rightarrow B$):** the MOI cannot occur.

Analysis: if $A \rightarrow B : ([h_h_OI_h_PI]K_A^{-1}, OI, [h_PI]N1, [N1]K_B, [CAcctInfo]K_B, CCertX)$, only A can change $h_h_OI_h_PI$ and only B can change $CAcctInfo$.

- **Transmission fourteen ($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : ([h_PurRes]K_B^{-1}, MCertX)$, only B can change h_PurRes .

Transmissions fifteen to twenty one are between A (Merchant Software) and B (Payment Gateway Software).

- **Transmission fifteen:** the MOI cannot occur.

Analysis: $AuthReq$ is produced by the Merchant Software. It is actually not a transmission here but an interaction.

- **Transmission sixteen ($A \Rightarrow B$):** the MOI cannot occur.

Analysis: if $A \rightarrow B : ([AuthReq]N2, [h_AuthReq]K_A^{-1}, MCertX, [h_PI]N1, [N1]K_B, [CAcctInfo]K_B, CCertX)$, only A can change $h_AuthReq$, and only B can change $CAcctInfo$.

- **Transmission seventeen ($B \Rightarrow A$):** the MOI cannot occur.

Analysis: if $B \rightarrow A : ([h_AuthRes]K_B^{-1}, [AuthResX]N3X, [N3X]K_A, [h_CapTok]K_B^{-1}, [CapTok]N4X, [N4]K_B, [CAcctInfo]K_B, PGCertX)$, only B can change $h_AuthRes$, h_CapTok and $CAcctInfo$.

- **Transmission eighteen:** the masquerade cannot occur.

Analysis: the encrypted CapTok is saved within the Merchant site. It is actually not a transmission here but an interaction.

- **Transmission nineteen:** the MOI cannot occur.

Analysis: CapReq is produced by the Merchant Software. It is actually not a transmission here but an interaction.

- **Transmission twenty(A $\Rightarrow$ B):** the MOI cannot occur.

Analysis: if $A \rightarrow B : ([\text{CapReq}]N5, [\text{h_CapReq_MCert}]K_A^{-1}, [N5]K_B, [N4]K_B, [\text{CapTok}]K_B, \text{MCertX})$, only A can change h_CapReq_MCert and only B can change CapTok.

- **Transmission twenty one(B $\Rightarrow$ A):** the MOI cannot occur.

Analysis: if $B \rightarrow A : ([\text{h_CapRes_MCert}]K_B^{-1}, [\text{CapRes}]N6, [N6X]K_A, \text{PGCertX})$, only B can change h_CapRes_MCert.

Analyses of MOI and its impersonation in the whole transaction process:

- a) the MOI can occur in transmission one. Thus, the cardholder may not get his certificate (at most the intruder can get his but this does not bring the cardholder in danger at all) but could do the registration again.
- b) the MOI can occur in transmission seven. Thus, the merchant may not get his certificate (at most the intruder can get his but this does not bring the merchant in danger at all) but could do the registration again.
- c) the MOI can occur in transmission eleven. Thus, the cardholder may not purchase anything (at most the intruder can purchase something but this does not bring the cardholder in danger at all) but could do the PR-PA-PC again.

Therefore, if there is any MOI in Cardholder / Merchant Registration, the cardholder / merchant cannot get his certificate so that the cardholder / merchant cannot

do the payment transaction further. In other words, there is not any possibility that the MOI might occur in the Cardholder / Merchant Registration process that could be exploited in the payment transaction process.

In PR-PA-PC process, only PInitReq can be modified, which may or may not lead to another PR-PA-PC process. Obviously, the MOI is useless for Masquerade and the LOI, because something modified before is meaningless for Masquerade, the LOI and whatever later on.

5.2.2 Leakage of Information

Figure 5.9 Leakage of Information in Cardholder Registration

Leakage of Information means that the information can be intercepted and revealed to the intruder. In our simulation, the real-line ellipse represents the normal situation, while the dot-line ellipse represents the LOI situation. Here, in Figure 5.9, we only present part of the complete process, i.e., the LOI in Cardholder Registration. The similar remaining parts of the complete process are omitted but the whole results of the complete process are given. To make the modeling simpler, we assume that in each case, each time the LOI may occur in only one of the transmissions although all six cases are shown together in Figure 5.9. Transmissions one to six are between A (Cardholder Software) and B (CCA):

- **Transmission one(A $\Rightarrow$ B):** the LOI can occur.

Analysis: CInitReq is a plaintext of data which can be leaked and cannot be discovered.

- **Transmission two(B $\Rightarrow$ A):** the LOI can occur.

Analysis: if $B \rightarrow A : (CInitRes, [h_CInitRes]K_B^{-1}, CAC)$, the data can be leaked to anyone who knows K_B .

- **Transmission three(A $\Rightarrow$ B):** the LOI cannot occur.

Analysis: if $A \rightarrow B : ([CAcctNo]K_B, [RegFormReq]N1, [N1]K_B)$, only B can know the data.

- **Transmission four(B $\Rightarrow$ A):** the LOI can occur.

Analysis: if $B \rightarrow A : (RegForm, [h_RegForm]K_B^{-1}, CAC)$, the data can be leaked to anyone who knows K_B .

- **Transmission five(A $\Rightarrow$ B):** the LOI can partially occur.

Analysis: if $A \rightarrow B : ([CPuK]N3, [CCertReq]N3, [N2]N3, [h_CCertReq_CPuK_N2]K_A^{-1}, [CAcctInfo]K_B, [N3]K_B)$, only $h_CCertReq_CPuK_N2$ can be leaked to anyone with K_A .

- **Transmission six($B \Rightarrow A$):** the LOI can partially occur.

Analysis: if $B \rightarrow A : ([CCertRes]N2, [h_CCertRes]K_B^{-1}, CAC)$, only $h_CCertRes$ can be leaked to anyone who knows K_B .

Similar situation is for the remaining part of the complete process, so we do not present the workflow but only give out the result and analysis for the remaining part of the complete process. Transmissions seven to ten are between A (Merchant Software) and B (Merchant CA).

- **Transmission seven($A \Rightarrow B$):** the LOI can successfully occur.

Analysis: MInitReq is a plaintext of data which can be leaked and cannot be discovered.

- **Transmission eight($B \Rightarrow A$):** the LOI can occur.

Analysis: if $B \rightarrow A : (RegForm, [h_RegForm]K_B^{-1}, CAC)$, the data can be leaked to anyone who knows K_B .

- **Transmission nine($A \Rightarrow B$):** the LOI can partially occur.

Analysis: if $A \rightarrow B : ([MCertReq]N1, [MPuK]N1, [MPuSK]N1, [h_MCertReq]K_A^{-1}, [N1]K_B, [MAcctInfo]K_B)$, only $h_MCertReq$ can be leaked to anyone who knows K_A .

- **Transmission ten($B \Rightarrow A$):** the LOI can partially occur.

Analysis: if $B \rightarrow A : ([MCertRes]N2, [h_MCertRes]K_B^{-1}, [N2]K_A)$, only $h_MCertRes$ can be leaked to anyone who knows K_B .

Transmissions eleven to fourteen are between A (Cardholder Software) and B (Merchant Software).

- **Transmission eleven($A \Rightarrow B$):** the LOI can successfully occur.

Analysis: PInitReq is a plaintext of data which can be leaked and cannot be discovered.

- **Transmission twelve($B \Rightarrow A$):** the LOI can successfully occur.

Analysis: if $B \rightarrow A : (P_{InitRes}, [h_P_{InitRes}]K_B^{-1}, MCert, PGCert)$, the data can be leaked to anyone who knows K_B .

- **Transmission thirteen**($A \Rightarrow B$): the LOI can partially occur.

Analysis: if $A \rightarrow B : ([h_h_OI_h_PI]K_A^{-1}, OI, [h_PI]N1, [N1]K_B, [CAcctInfo]K_B, CCert)$, $h_h_OI_h_PI$ can be leaked to anyone with K_A . OI can be leaked to anyone.

- **Transmission fourteen**($B \Rightarrow A$): the LOI can occur.

Analysis: if $B \rightarrow A : ([h_PurRes]K_B^{-1}, MCert)$, h_PurRes can be leaked to anyone with K_B .

- **Transmission fifteen**: the LOI cannot occur.

Analysis: AuthReq is produced by the Merchant Software. It is actually not a transmission here but an interaction.

- **Transmission sixteen**($A \Rightarrow B$): the LOI can partially occur.

Analysis: if $A \rightarrow B : ([AuthReq]N2, [h_AuthReq]K_A^{-1}, MCert, [h_PI]N1, [N1]K_B, [CAcctInfo]K_B, CCert)$, only $h_AuthReq$ can be leaked to anyone with K_A .

- **Transmission seventeen**($B \Rightarrow A$): the LOI can partially occur.

Analysis: if $B \rightarrow A : ([h_AuthRes]K_B^{-1}, [AuthRes]N3, [N3]K_A, [h_CapTok]K_B^{-1}, [CapTok]N4, [N4]K_B, [CAcctInfo]K_B, PGCert)$, only $h_AuthRes$ can be leaked to anyone with K_B .

- **Transmission eighteen**: the LOI cannot occur.

Analysis: the encrypted CapTok is saved within the Merchant site. It is actually not a transmission here but an interaction.

- **Transmission nineteen**: the LOI cannot occur.

Analysis: CapReq is produced by the Merchant Software. It is actually not a transmission here but an interaction.

- **Transmission twenty**($A \Rightarrow B$): the LOI can partially occur.

Analysis: if $A \rightarrow B$: $([CapReq]N5, [h_CapReq_MCert]K_A^{-1}, [N5]K_B, [N4]K_B, [CapTok]K_B, MCert)$, only h_CapReq_MCert can be leaked to anyone with K_A .

- **Transmission twenty one ($B \Rightarrow A$):** the LOI can partially occur.

Analysis: if $B \rightarrow A$: $([h_CapRes_MCert]K_B^{-1}, [CapRes]N6, [N6]K_A, PGCert)$, only h_CapRes_MCert can be leaked to anyone with K_B^{-1} .

Analyses of LOI and its impersonation in the whole transaction process:

From the LOI analysis above, we know that the LOI almost exists everywhere in the whole transaction process. This is a pitfall for the SET protocol but not a big problem since the important information can never be disclosed to the intruder. The covered information includes $CACctNo$, $CACctInfo$ and $CCertRes$ in Cardholder Registration, $MACctInfo$, $MCertRes$ in Merchant Registration, PI (see Table 5.1), $CACctInfo$, $CapTok$, and $CapRes$ in PR-PA-PC processes. In fact, $CCert$ from $CCertRes$ and $MCert$ from $MCertRes$, which cannot be exposed in Cardholder Registration and in Merchant Registration respectively, can be exposed in PR-PA-PC. Under such circumstances, those can be exposed in Cardholder Registration and Merchant Registration are less important. On the other hand, only those covered in Cardholder / Merchant Registration are used in the PR-PA-PC process. Therefore, the LOI in both Cardholder Registration and Merchant Registration cannot actually affect the PR-PA-PR at all.

In the PR-PA-PC process, besides $PInitReq$, $PInitRes$ and OI (see Table 5.1), all of the LOI are hash values and certificates from which the intruder actually cannot know the real information he wants to know. The leakage of $PInitReq$ and $PInitRes$ is less important than the masquerade/modification of $PInitReq$ and $PInitRes$ that were analyzed

before. The leakage of OI cannot be used to impersonate at a later time because the remaining part of the PR-PA-PC does not use OI any more.

Apparently, LOI is useful for Masquerade and MOI, because something leaked before may be used for Masquerade, the MOI and whatever later on. From the above analyses, we do not need to consider the first transmission in the three phases any more, since Masquerade / MOI / LOI can occur together in it, which is actually much like only one of them occurred, we analyzed these before. On the one hand, the items in Cardholder Registration that can be masqueraded are CAcctNo, RegFormReq and CAcctInfo which cannot be leaked. The item in Merchant Registration that can be masqueraded is MAcctInfo which cannot be leaked. The items in PR-PA-PC that can be masqueraded are CAcctInfo and CapTok which cannot be leaked. Therefore, LOI cannot be used by Masquerade. On the other hand, there is not any other MOI in the three phases, therefore, LOI cannot be used by MOI.

5.2.4 Repudiation

Repudiation means a participant to transaction or communication activity later falsely denies that the activity occurred. It occurs when an actual participant denies having originated, submitted or received a message.

The models, simulations and results are much like those in Masquerade although there is an actual intruder in Masquerade. The sender may use a virtual intruder to masquerade the sender itself. Here the sender is only considered since the receiver is also

a sender for the next transmission of the SET protocol. Therefore, wherever masquerades can successfully occur, repudiations can also successfully occur.

5.3 Summary

This chapter presented the workflows of SET payment transactions and four related potential security threats within these transactions. It also presented the simulation processes of transactions (not all transactions) and discussed the results of simulation implementations for the four security threats in all transactions.

The key process of most transactions of the SET protocol could be described by using the following example of black boxes. We assume that there are two locked black boxes, one is big and the other is small. The key for big box is Key_b . The big box is holding the small box, the key for the small box (Key_s), and others (O). The small box is only holding something (V) which can verify the O. The sender S sends these two boxes to the receiver R. Here, the random Key_s is unique for one process. On the other hand, the receiver R receives these two boxes from the sender S. R can get Key_b and thus know Key_s and O, and R can know V eventually. Between S and R, there may be an intruder. The intruder cannot get Key_b since Key_b is encrypted with the R's public key, which means only R can know Key_b . But the intruder may pretend to be S. That is why R should know V that can verify whether or not the O is tampered by the intruder.

Although Masquerade, Modification of Information and Repudiation can occur or partially occur in some transmissions of all sub-transactions, the entire payment transactions of SET can still process safely because the most important data including

CACctNo, CACctInfo, MACctInfo and PI can never be disguised or modified. Leakage of Information can occur or partly occur in almost all transmissions of all sub-transactions. But the most important information can never be disclosed to the intruder.

The main security measures in SET for preventing potential threats include multi-way authentication and cryptography. Authentication uses RSA cryptography, digital signatures including single hashing and / or dual hashing, and CA verifications. Particularly in transactions of Purchase Request, dual signature is used to prevent the merchant from reading the account information in the Payment Information and to prevent the acquirer from knowing the detail in the Order Information. Cryptography uses both symmetric cryptography by random numbers and asymmetric cryptography by private / public keys. All these security mechanisms make secure transaction over open networks.

Secure Sockets Layer (SSL), SET's closest competitor, is an encryption protocol primarily used to secure data as it travels over the points between. It is widely used and will likely continue to play a key role with non-credit card applications.

Besides the above analyses and verifications of application level security of SET, other level securities described in Chapter one should be considered before using SET. For example, it is necessary to build a secure web site. According to [22], most security leakage of credit card on Internet has not been caused by hackers picking card numbers off the wire. Rather, the leakage has resulted from poor security at the merchant's Web site. Hackers have broken in and copied files with card numbers right off the Web server's disk.

From the SET protocol modeling, we know Pr/T Nets can handle rather complex systems in a succinct and manageable way. Reachability analysis is one of the most straightforward and easily protocol verification methods. The SET protocol verification is done through the simulation in PROD which is based on reachability analysis. From the simulation of the SET protocol, we conclude that PROD can produce a reachability graph for the SET protocol. Within a graph, a node exists for each reachable state and an arc exists for each possible change of state. The graph shows the whole process of the SET protocol, including all nodes with places / transactions, and all transmissions. In other words, we can view the SET protocol in a succinct way, understand easily how it works, investigate whether or not an attacker occurs and verify the potential threats from the attacker. All in all, PROD makes all these things much easier than just analyzing a simple examination of the SET protocol specification. Furthermore, the correctness and the security of the SET protocol are verified by the PROD simulation results along with logic analyses.

Chapter 6

Conclusions

In this thesis, the fundamental issues addressed are the verifications of security in the Secure Electronic Transaction (SET) protocol. The key payment transactions are modeled and simulated by the use of Predicate Transition Nets approach and its reachability analyzer PROD. Although the SET protocol provides its inherent security mechanism which may prevent the intruder from attacking the information, application level security issues, in terms of masquerade, modification of information, leakage of information, and repudiation, are investigated with Pr/T Nets and PROD in every transmission of the SET protocol. The verification processes include the system modeling, the simulation, the experiment/result and the analysis of results by using the Logic of Authentication.

From the Pr/T Net analysis, we can determine if a violation takes place. As the Pr/T Net result we got before, when there is a kind of threat in the SET protocol, there is going to be an additional transaction and the change of one or more items as well (compared to the normal situation). All these can be seen from the Pr/T Net. Therefore, it is very useful if the transaction process of the SET protocol can be monitored by the Pr/T Nets in actual applications which could be the future work of the thesis. Table 6.1 summaries the conclusions of the verifications of the SET protocol.

Table 6.1 Conclusions of the Verifications of the SET Protocol

Transmission	Masquerade	MOI	LOI	Repudiation
one	yes	yes	yes	yes
two	no	no	yes	no
three	yes	no	no	yes
four	no	no	yes	no
five	yes	no	yes	yes
six	no	no	yes	no
seven	yes	yes	yes	yes
eight	no	no	yes	no
nine	yes	no	yes	yes
ten	no	no	yes	no
eleven	yes	yes	yes	yes
twelve	no	no	yes	no
thirteen	yes	no	yes	yes
fourteen	no	no	yes	no
fifteen	no	no	no	no
sixteen	yes	no	yes	yes
seventeen	yes	no	yes	yes
eighteen	no	no	no	no
nineteen	no	no	yes	no
twenty	yes	no	no	yes
twenty one	no	no	no	no

From table 6.1 and the verifications/analyses in Chapter 5, we know, Masquerade, Modification of Information and Repudiation can occur or partially occur in some transmissions of the SET protocol, but the important data can never be disguised or modified. Leakage of Information can occur or partly occur in almost all transmissions of the SET protocol, but the most important information can never be disclosed to the intruder. All in all, the SET protocol is a reliable card payment transaction protocol on the Internet.

The following is the summary of contributions:

- Some kinds of threat in payment transactions (Chapter 2) and the detailed key payment transactions in SET (Chapter 3) are summed up.
- The workflows for all payment transactions and sub-transactions in SET with Predicate Transition Nets approach are analyzed and modeled (Chapter 5).
- The payment transactions in SET are simulated with PROD (Chapter 5).
- The workflows for all payment transaction with five kinds of threat on every transmission are analyzed, modeled with Pr/T Net and simulated with PROD (Chapter 5).
- the security issues in the payment transaction processes of SET are verified by using PROD and those conclusions are analyzed(Chapter 5).

The methods used in this thesis can also be used to analyze other communication protocols. They are summarized as follows:

- The protocols can be analyzed and modeled by using the Predicate Transition Nets approach.
- The modeled Pr/T Nets of the protocols can be simulated and further analyzed by PROD directly.
- The PROD results of the Pr/T Nets of the protocols can be used to verify the correctness of the protocols graphically.
- The above principles can also be used to verify the potential threats that might occur.
- The Logic of Authentication can be used in analyzing/verifying the protocols.

References

- [1] American National Standards Institute, ANSI X3.92: *American National Standard, Data Encryption Algorithm*, 1981
- [2] American National Standards Institute, ANSI X9.9: *American National Standard for Financial Institution Message Authentication (Wholesale)*, 1986
- [3] Aura, Tuomas., *Modelling the Needham-Schröder authentication protocol with high level Petri nets*. Technical Report B14, Helsinki University of Technology, Digital Systems Laboratory, Espoo, Finland, September 1995.
- [4] Berthelot, G. and Terrat, R., *Modeling and Proofs of a Data Transfer Protocol by Predicate/Transition Nets*. Informatik-Fachberichte 52: Application and Theory of Petri Nets. — Selected Papers from the First and Second European Workshop on Application and Theory of Petri Nets, Strasbourg, Sep. 23-26, 1980, Bad Honnef, Sep.28-30, 1981
- [5] Brassard, G., *Modern Cryptology: A Tutorial*, Lecture Notes in Computer Science 325, Berlin: Springer-Verlag, 1988.
- [6] Burrows, M., Abadi, M., and Needham, R., *A Logic of Authentication*. ACM Transactions on Computer Systems, Vol. 8, No. 1, 1990, pp.18-36.
- [7] Diaz, M., *Petri Net Based Models for the Specification and Validation of Protocols*. Lecture Notes in Computer Science Vol.255. Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, September 1986.
- [8] Diffie, W., and Hellman, M., *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol. IT-22, no.6 (1976), pp.644-654; Diffie, W., *The First Ten Years of Public Key Cryptology*, in Gustavus J. Simmons (Ed.), Contemporary Cryptology: The Science of Information Integrity (New York: IEEE Press, 1992), pp.136-175.
- [9] Dijkstra, E.M., *Hierarchical Ordering of Sequential Processes*. In Hoare, C.A.R., and Perrott, R.H.(Eds.), Operating Systems Techniques, Proceedings of a Seminar held at Queen's University, Belfast 1971, APIC Studies in Data Processing No.9, Academic Press. London 1972, pp.72-93.
- [10] Ford, W., and Baum M.S., *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures & Encryption*, Prentice-Hall, Inc., 1997.
- [11] Genrich, H. J., *Equivalence Transformation of PrT-Nets*, No.424, Lecture Notes in Computer Science, 1989, pp.179-208

- [12] Genrich, H. J., *Predicate/Transition Nets*, in [4.6], pp.3-43.
- [13] Genrich, H. J., Lautenbach, K., *System Modelling with high-level Petri Nets*, Theoretical Computer Science, vol.13, 1981, pp.109-136
- [14] Heljanko, K., *Model Checking the Branching Time Temporal Logic CTL*. Master's thesis, Helsinki University of Technology, Department of Computer Science and Engineering, 1997. Also available as Research Report A45, Digital Systems Laboratory, Helsinki University of Technology.
- [15] <http://cacr.math.uwaterloo.ca/hac/>
- [16] <http://edifact.standards.com.au/>
- [17] <http://saturn.tcs.hut.fi/~petrinet/>
- [18] <http://www.activmedia.com/press/webrevenues.html>
- [19] <http://www.capitolnet.com/press/funion.html>
- [20] <http://www.mastercard.com/shoponline/set/set.html>
- [21] <http://www.pcworld.com/news/daily/data/0297/970131142704.html>
- [22] <http://www.standishgroup.com/settogo.htm>
- [23] <http://www.uncitral.org/>
- [24] Jensen, K., Rozenberg, G. (Eds.), *High-level Petri Nets: Theory and Application* Springer-Verlag 1991
- [25] Kalakota, R. and Whinston, R., *Electronic Commerce: A Manager's Guide*. Massachusetts: Addison Wesley Longman, Inc., 1997
- [26] Kou, W., *Network Security and Standards*, Kluwer Academic Publishers, 1997.
- [27] Loeb, L. H., *Secure Electronic Transactions: Introduction and Technical Reference*. Artech House, Inc., 1998
- [28] Loyer, B. et al., *Communication Protocol Specification and Validation by Means of Petri Nets: Applications to the CSN Subscriber Connection Unit of the E10 and MT Systems*. Rev. FITCE(Belgium), Vol.25, No. 1, pp.8-17(Jan.-March, 1986)
- [29] Menezes et al. *Handbook of Applied Cryptography*.

- [30] Murata, T., *Petri Nets: Properties, Analysis and Applications*, Proceedings of the IEEE, Vol. 77, No 4, April, 1989, pp.541-580.
- [31] Peterson, J. L., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, N.J., 1981.
- [32] Prinoth, R., *Specification and Verification of Communication Protocols*. Proc. Of the Workshop Intr. of High Level Protocol Standards for Open systems Interconnection, Paris, June 27-29, 1983, pp.335-352(1983)
- [33] Pyssysalo, T., Ojala, L., *A High-Level Net Model of a Video on Demand System*. Research Report A36, Digital Systems Laboratory, Helsinki University of Technology, 1995.
- [34] Pyssysalo, T., Ojala, Leo., *Causal Modeling of a Video on Demand System Using Predicate/Transition Net Formalism*. In Proceedings of the 22nd Euromicro Conference, Prague, Czech Republik, September 2-5, 1996, pp.6, Los Alamitos, California, 1996. The Institute of Electrical and Electronics Engineers (IEEE) Computer Society Press.
- [35] Reisig, W., *Petri Nets, An Introduction*, EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin, 1985
- [36] Rivest, R.L., Shamir, A., and Adelman, L., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol.21, No.2, February 1978, pp.120-126.
- [37] Stallings, W., *Network and Internetwork Security: Principles and Practice*, Prentice-Hall, Inc., 1995.
- [38] Schneier, B., *Applied Cryptography*, 2nd edition, New York: Wiley, 1995.
- [39] *SET Secure Electronic Transaction Specification, Book 1: Business Description*. USA, May, 1997
- [40] *SET Secure Electronic Transaction Specification, Book 2: Programmer's Guide*. USA, May, 1997
- [41] *SET Secure Electronic Transaction Specification, Book 3: Formal Protocol Definition*. USA, May, 1997
- [42] Tanenbaum, A. S., *Computer Networks* 3rd Edition, Prentice Hall, Inc. 1996
- [43] Tanir, O., Agarwal, Vinod K., Bhatt, P.C.P: *A Specification-Driven Architectural Design Environment*, pp.26-35, June, 1995 IEEE Computer vol.28 No.6

- [44] Valmari, A., *Alleviating state explosion during verification of behavioral equivalence*. Report A-1992-3, University of Helsinki, Department of Computer Science, 1992.
- [45] Valmari, A., *A stubborn attack on state explosion*. *Formal Methods in System Design*, 1(4): pp.297-322, 1992.
- [46] Varpaaniemi, K., *On computing symmetries and stubborn sets*. Technical Report B12, Helsinki University of Technology, Digital Systems Laboratory, Espoo, Finland, April 1994.
- [47] Varpaaniemi, K., Halme, J., Hiekkänen, K. and Pyssysalo, T., *PROD Reference Manual*, Technical Report B13, Helsinki University of Technology, Digital Systems Laboratory, August 1995