Overview and security analysis of electronic payment systems

Majeri, I.

Award date:
2006

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Master's thesis:

Overview and Security Analysis of Electronic Payment Systems

By
Ikbal Majeri

Supervisor: Dr. Sjouke Mauw

Eindhoven August 2006

Dedication

I dedicate this work with honor and respect for my parents for the assistance, the affection and the unbounded love they surrounded me with since my youth age.

I also dedicate this thesis to my brother Ihsen for his support and encouragement which stimulated me throughout the preparation of this thesis.

Acknowledgement

In 2003, I have started my master program of Information Security Technology at Eindhoven University of Technology (TU/e). From all the various courses that were encountered throughout the time I spent there I was always fond of the ones related to the field of electronic payment systems. One of the lectures I followed was the course cryptographic systems by Dr. Benne de Weger that gave me a taste of the development of Electronic payment systems, that course was followed one trimester later by the course Information Security Technology Seminar where I made a presentation on an attack on electronic payment systems in France. After discussing it further with my professor at that time ir. Jan Verschuren and my supervisor Dr. Sjouke Mauw, we decided to use it as basis for my master's thesis and we came out with the title of overview and security analysis of electronic payment systems.
This master thesis will not have been conducted successfully without the help of my supervisor Dr. Sjouke Mauw who helped me a lot by giving me advises and reviewing my thesis without forgetting the fact that he helped me in the personal development level. I am also grateful to Mr Wil Linders and Mrs Henny Van Keulen for their help during the pre and post preparation of this master's thesis with the administrative matters. Finally I would like to thank all my friends I met at university: it was really a great experience to work within such a motivating international environment as the one offered by Eindhoven University of Technology.

TABLE OF CONTENTS

List of Acronyms

3-D	Three Domains
AAC	Application Authentication Cryptogram
AC	Application Cryptogram
ACH	Automatic Clearing House
ACS	Access Control Server
AHS	Authentication History Server
ARPC	Authorization ResPonse Cryptogram
ARQC	Authorization ReQuest Cryptogram
B2B	Business-to-Business
B2C	Business-to-Consumer
C2C	Consumer-to-Consumer
CA	Certification Authority
CAVV	Cardholder Authentication Verification Value
CDA	Combined DDA and application cryptogram generation
CPA	Common Payment Application
DDA	Dynamic Data Authentication
EMV	Europay Mastercard Visa
IACS	Issuer's Access Control Server
IC	Integrated Circuit
MAC	Message Authentication Code
MSC	Message Sequence Chart
PA	Payment Authorization
PAN	Primary Account Number
PAR	Payment Authorization Request
PC	Personal Computer
PG	Payment Gateway
PI	Payment Information
PIN	Personal Identification Number
PKI	Public Key Infrastructure
POS	Point Of Sale
SA	Secret Authentication
SET	Secure Electronic Transaction
SDA	Static Data Authentication
SSL	Secure Sockets Layer
TC	Transaction Certificate
TLS	Transport Layer Security

List of figures

CHAPTER 1 INTRODUCTION

The goal of this chapter is to describe the motivation that led to conduct this research project (section 1.1) followed by the main contributions that will come up from this thesis (section 1.2) and subsequently we will conclude by the overall structure of the thesis (section 1.3)

1.1. Motivations

When it comes to payment options, nothing is more convenient than electronic payment [11]: all what has to be done is to type some information on a special device, confirm the transaction and we are done, hence no need to make use of "traditional means of payment" (see section 3.1 for details). This easy-to-use-convenience makes more and more people attracted to use this kind of payments. Electronic payments have multiple benefits for the businesses as well by lowering the costs: The more payments they can process electronically, the less they spend on paper and postage [11].
Despite these mutual benefits for both people and businesses, electronic payments are facing some drawbacks especially for the customers. These drawbacks are mainly about privacy issue and also people's concern about their money especially with the booming of Internet. In order to win people's confidence in electronic payments several ways to enhance security in electronic payment have been developed depending on the platform that is been used in. For example regarding doing secure online credit card payments (see section 3.3), SET [5] has been developed but due to reasons that will be detailed later in this thesis it has been abandoned and replaced by the use of SSL/TLS[14]. Also regarding doing POS-based transaction there was the use of magnetic strip cards but all the methods above did not give the insurance needed by the customer due to its shortage in security or to discovery of new security flaws.
Nowadays tendency is to replace the use of magnetic strip cards by chip-based ones or what is also called smart cards which is adopted by EMV [7-10] and CPA [6] for POS transactions and also the VISA's 3-D SECURE [15-17] to address security issues in online credit card payments but the questions rises up again: will these new methods provide the security insurance needed by the customers to enhance a faster popularity in the use of electronic payments?
The work done through this thesis will try to give a convincing answer to this question by doing a security analysis of each of the protocols above; we will also try to use the scyther tool in the modeling and analysis of each protocol.

1.2. Main contributions

The main task of this thesis is to perform a security analysis of three protocols to conduct secure electronic transactions.
The main contributions of this thesis are the following:

- Security requirements to conduct secure electronic transactions have been gathered, identified and described.
- Three protocols namely EMV, CPA and VISA 3-D SECURE will be described and a security analysis on each one of them will be performed to see how the security requirements are fulfilled.
- The scyther tool will be used as a modeler and security checker of each part of the protocol that can be modeled via scyther.

1.3. Structure of the thesis

The thesis is organized in the following way:
Chapter 2 is a security framework that will introduce the reader to the notion of security threats, requirements and mechanisms that are relevant for electronic payment systems.
Chapter 3 will give the reader an overview about electronic payments
Before moving to the description and security analysis of each protocol, a chapter 4 will give an overview about the scyther tool that will be used later on for analysis.
In chapter 5, I will propose an overview of the EMV card payment transaction procedure followed by a security analysis of how the EMV matches the identified security requirements both by hand and by Scyther.
In chapter 6, I will propose an overview of the CPA card payment transaction procedure followed by a security analysis of how the CPA matches the identified security requirements.
In chapter 7, I will propose an overview of the 3-D SECURE card payment transaction procedure followed by a security analysis of how it matches the identified security requirements.
Finally, chapter 7 will conclude this thesis.

CHAPTER 2 SECURITY FRAMEWORK

This chapter will introduce the reader to the basic security knowledge about the requirements and mechanisms that are relevant to the field of electronic payment systems.

2.1. Introduction

This chapter consists of two parts: in section 2.2 we will define the general security requirements that need to be fulfilled by electronic payment systems; it will be followed in section 2.3 by the security mechanisms that are used to implement them.

2.2. Security Requirements

According to [2] “electronic payment systems must exhibit integrity, authorization, confidentiality, availability and reliability” to which we can add access control and non-repudiation. These of course are general security requirements so we will try in the rest of the section to give detailed description for each of these security requirements following the definitions given in [1] and [2].

2.2.1. Confidentiality

The purpose of this service is to provide insurance that the data exchanged can only be viewed by the authorized parties. This service consists of four subdivisions [1]. The first one is called connection confidentiality which means that confidentiality will be granted for all user data on a connection. The second subdivision is connectionless confidentiality in which confidentiality will be granted to all user data in a single data block. The third subdivision is selective field confidentiality and here confidentiality will be granted to a selection of fields within the user data whether on connection or in a single data block. And finally the last subdivision is traffic flow confidentiality and by that we mean no useful information can be extracted by observing the flow of traffic.

2.2.2. Integrity

The purpose of this service is to provide insurance that the data coming from the sender was not subject to any modifications on the way to the receiver. This service is made up of five subdivisions according to [1]. The first one is called connection integrity with recovery: This means that all user data on connection will be checked to detect any kind of modification, insertion, deletion or replay of data within the entire user data and in case this is detected then recovery attempt will be initiated. The second subdivision is called connection integrity without recovery which is same

as above but this time without recovery attempt. The third subdivision is selective field connection integrity and here integrity checking will be performed on a selection of fields within the user data on connection. The fourth subdivision is connectionless integrity where integrity checking will be performed on a single connectionless user data block. And finally we have selective field connectionless integrity where integrity checking will be performed on a selection of fields within the connectionless user data.

2.2.3. Authorization

This service provides that the receipt of payments should not be allowed without an explicit permission. This service is made up of three subdivisions [2]. The first one is called out-band authorization where the verifying party will inform the authorizing party of the transaction occurrence which then will either deny or authorize the transaction using a secure channel (hence the name “out-band”). The second subdivision is called password authorization where the authorization will be subject to a cryptographic checking using a shared secret between the verifying and the authorizing parties. And finally we have signature authorization where the verifying party will require a digital signature based authorization from the authorizing party.

2.2.4. Availability and reliability

When doing an electronic payment transaction, it is required that the transaction is done atomically: that means that either the transaction is done in total or not at all. No user would accept a loss of money due to a network or system crash [2].

2.2.5. Access control

This service provides insurance that for anyone who has access to a certain resource will be checked to see how he/she got access to this resource and what can or cannot be done with this resource.

2.2.6. Non-repudiation

This service provides insurance that the no one from the entities involved in a transaction can later deny that he/she was not involved in the transaction. This service is made up of two subdivisions [1]. The first one is called non-repudiation of origin which provides proof that a message was indeed sent by that specific party. And we have also non-repudiation of destination to provide a proof that a message was indeed received by that specific party.

2.3. Security Mechanisms

In order to enforce the security requirements defined above, we need to make use of what is called security mechanisms. The ones that will be defined are the following: encryption (both symmetric and asymmetric), digital signature scheme, MAC (Message Authentication Codes) and finally hash functions.

2.3.1. Symmetric encryption

As the name suggests it, in a symmetric encryption we will make use of one key to encrypt a plaintext and transform it into a ciphertext. Once the ciphertext reaches its destination the receiver will use the same key to decrypt the ciphertext in order to recover the plaintext (see figure 2.1).

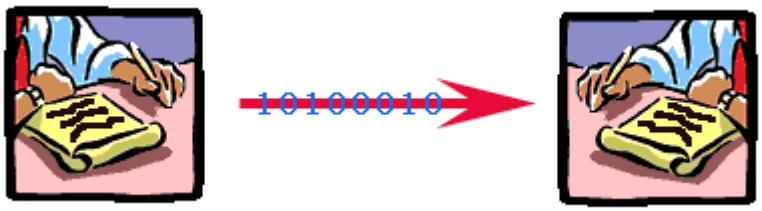

Figure 2.1: Symmetric encryption (source: http://www.davidreilly.com)

The reader might notice here that since one key is used for both encryption and decryption of messages, this key has to be kept secret between the two communicating parties.
For more details on this type of encryption we refer the reader to [13].

2.3.2. Asymmetric encryption

Unlike the previous encryption mode, we will now make use of two keys: one key called public key and, as its name suggests, will be made public to everyone and another key called secret key which will be known only the owner of the public key.
The encryption works in the following way (see figure 2.2): imagine we have Alice who wants to communicate with Bob. Alice knows Bob's public key so she will send him a message m encrypted with Bob's public key which will result in a ciphertext. Once Bob gets this ciphertext, he will be able to recover the initial message by decrypting the ciphertext using his secret key.

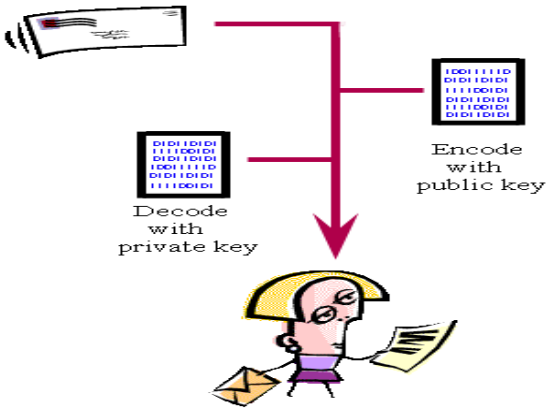

Figure 2.2: Asymmetric Encryption (source: http://www.davidreilly.com)

For more details, we refer the reader to [13].

2.3.3. Digital signature scheme

A digital signature scheme consists of three algorithms namely:

- Key generation algorithm: Same steps as for asymmetric encryption (see section 2.3.2).
- The signature generation algorithm: Given a message m, we will sign this message using the secret key and we will out put both the message and it signature.
- The signature verification algorithm: Once the message and its signature are received we will then apply the public key on the signed message and compared to the message m sent along. If the message recovered is the same as the message that has been sent then we get a proof that the message has not been modified on the way.

These steps are shown in figure 2.3 below.

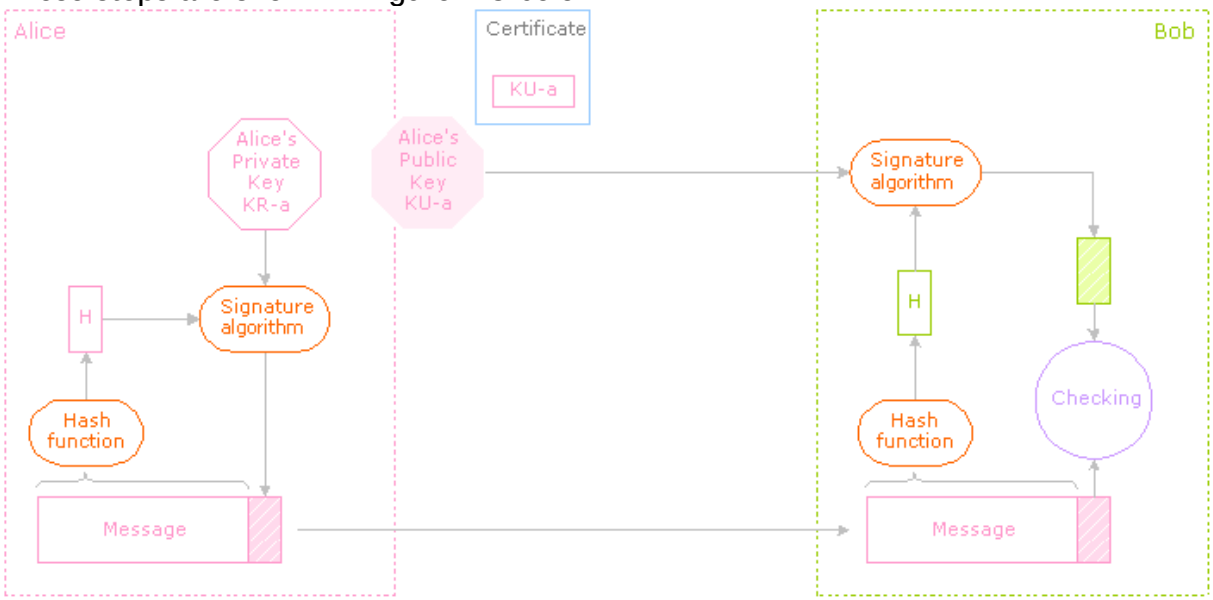

Figure 2.3: Digital Signature (source: http://www.tech-invite.com)

For more details on this type of encryption we refer the reader to [13].

2.3.4. Message Authentication codes MAC

According to [17], “a MAC algorithm accepts as input a secret key and an arbitrary length message to be authenticated and out puts a MAC” (see figure 2.4). The purpose of using MACs s to provide integrity and authenticity of the messages exchanged as well as allowing the verifying parties (under the assumption that they have the secret key) to verify if the message has not been altered during transit.

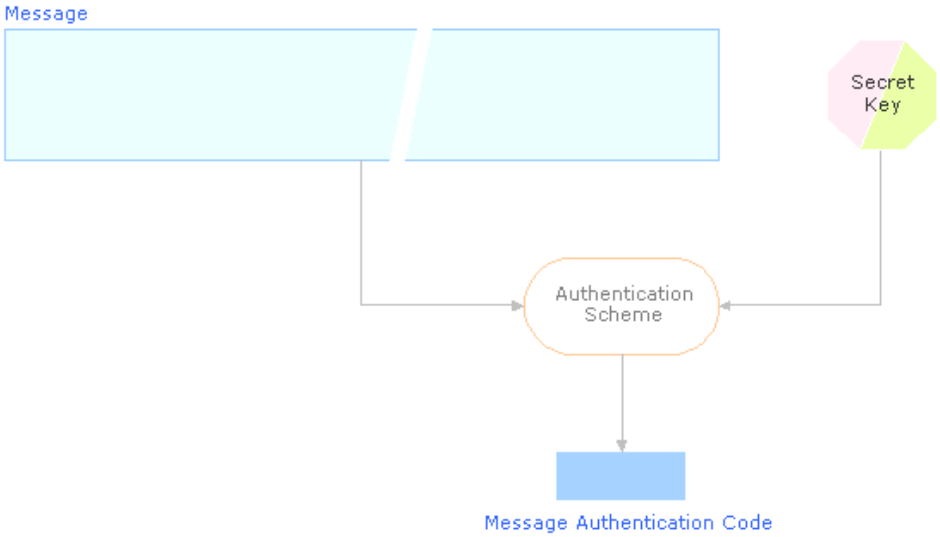

Figure 2.4: Message Authentication Code (source: http://www.tech-invite.com)

A MAC is considered to be secure in the case it withstands existential forgery [18]. For more details on MAC, we refer the reader to [13].

2.3.5. Hash Functions

A cryptographic hash function (see figure 2.5) is an algorithm that takes as input a message m of variable length and outputs a fixed-length `hash code'. Cryptographic hash functions are used as a building block to help construct other cryptographic mechanisms. For example, hash functions are used as a component in almost all practical digital signature schemes.
A cryptographic hash function is considered secure if it satisfies the following properties [13].

- *Pre-image resistance*: Given a random hash code, it must be computationally infeasible to find an input which the hash function maps to that hash code,
- *2nd pre-image resistance*: Given any input, it must be computationally infeasible to find a second input which gives the same hash code, and
- *Collision resistance*: It must be computationally infeasible to find two inputs that give the same hash code.

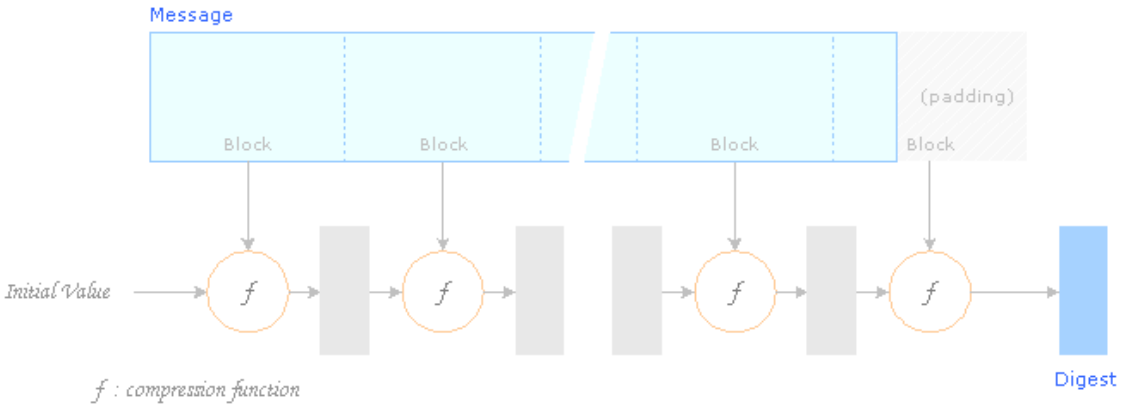

Figure 2.5: Hash Function (source: http://www.tech-invite.com)

Again for more details on the different types of hash functions we refer to [13].

CHAPTER 3 ELECTRONIC PAYMENT SYSTEMS

In this chapter electronic payment systems are introduced.
We will first talk about the different payment modes (section 3.1). A generic model of an electronic payment system is then provided (section 3.2). This is followed by a description of some of the properties that distinguish the various types of electronic payment systems (section 3.3).
This chapter will be based on the work done by Mansour Al-Meaither during his PhD thesis [1] hence the terminology used by him will be followed.

3.1. Traditional VS. Electronic payment:

Doing a payment transaction depends on the mode of payment being used. We have two distinct modes of payment: “traditional mode of payments” and “electronic payment” which is done as the name suggests only electronically. Although these two modes are different, they have one thing in common: the fact that the actual flow of money takes place from the payer's account to the payee's account [1]. The traditional mode of payment consists of cash and cheques while the electronic one consists of giros, Automatic Clearing House or in short ACH, wire transfer and debit/credit cards. We will detail each one of them below:

- Cash: Cash is the easiest way to conduct payments. They are relatively inexpensive to process and cost retailers less than other types of payments [1]. However they are subjects to theft, forgery and counterfeiting.
- Cheques: When a person can not carry too much cash along then it is better to make use of cheques where a person just need to put the right amount of money and a signature on the cheque. It looks convenient however according to [1] they are costly for both the user and banks in addition to the fact that cheque clearing is time consuming.
- Giro payments: A giro is an instruction to the payer's bank to transfer funds to the payee's bank [1]. This is under the condition of course that the payer has the funds required to do the giro payment.
- Automatic Clearing House (ACH): The growing number of cheque and giro payments has made paper-based clearing increasingly difficult; this has led to the development of ACH payments [1]. ACH payments are used for small and mid value transactions.
- Wire transfer: Unlike ACH, wire transfer payments are used for high value transactions. In these transactions the risk level is high (because of high transaction value), and thus different procedures involving more security are required [1].
- Debit/credit card payments: Debit/credit cards appear to be the most expensive form of payment, with a transaction cost five times higher than the cost of a cash transaction [1]. However debit/credit cards have became popular due to strong demand from consumers (see section 1.1).

Electronic payment systems have a valuable advantage over their paper-based equivalent when it comes to processing costs. In fact, the costs associated with

electronic payments usually lie between a third and a half of the costs posed by cheques or other paper based transactions [1].

3.2 Electronic payment model

In this section we will describe an electronic payment model. The model shown in figure 3.1 involves interactions between four players: a payer, a payee, an issuer, and an acquirer. Below we will describe the role of each player in the model:

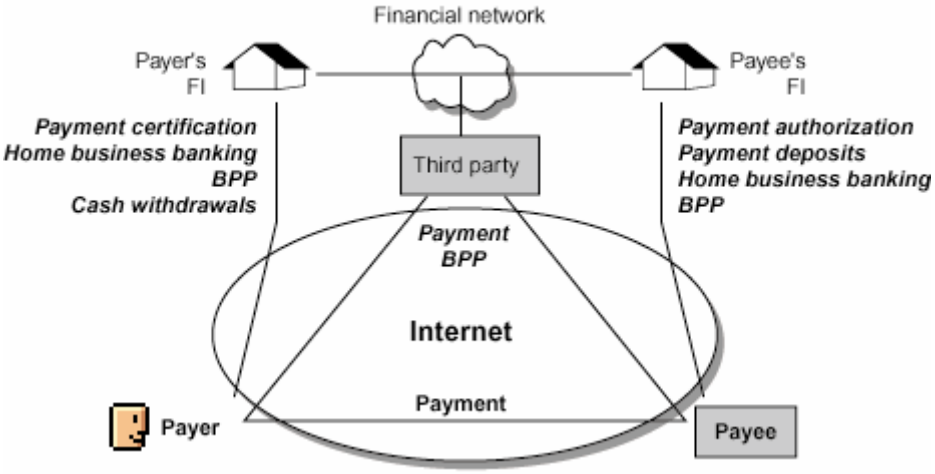

Figure 3.1: Model of an electronic payment system (source: www.ibm.com)

- Payer: More famously known as buyer or consumer. This is the entity that will make the payment by means of an electronic payment instrument obtained from the issuer.
- Payee: Also famously known as seller or merchant. This entity will receive the funds resulting from the payment. The payee sends the received payment instrument, received from the Payer, to the acquirer for clearance and settlement.
- Issuer: This is a financial institution that provides electronic payment instruments to the payer to use in a payment [1].
- Acquirer: This is a financial institution associated with the payee that verifies the validity of the deposited payment instrument by clearing it with the issuer. After the settlement of funds between issuer and acquirer, the acquirer credits the payee's account with the monetary value stated by the deposited instrument [1].

3.3 Characteristics of electronic payment systems

After having described the payment model in section 3.2 and the players involved in it, we will now describe the characteristics of electronic payment systems. According to [1]:“These characteristics can be used to analyze an electronic payment system. Moreover, these properties help to determine the interpretation of the overall model and the security requirements of participants in such a payment system”.

3.3.1 Flow of Information

We can classify electronic payment systems depending on the way money transits. We have three types of Information flow namely cash based, account based and indirect payments.

Cash based systems

In this type of system, the payer will first buy a token from his/her bank (issuer); who will immediately debits the payer's account. Secondly, the payer gives the token to the payee who will deposit it at his/her bank (acquirer) which will credit the payee's account and then do the clearing with the issuer
Telephone cards and chipknip are examples of such systems.

Account-based systems

In this type of system, the payer will give a token to the payee who will deposit the token at his/her bank (acquirer). The acquirer will then do the crediting of the payee's account then send the token to the issuer. In that phase, the issuer will then debit the payer's account and then both issuer and acquirer will do clearing.
Credit card payments and cheques are examples of such systems.

Indirect payment systems

The payer will instruct the issuing bank to transfer an amount of money from his/her account to the payee's account at that acquiring bank.
Home/Internet banking and giro payments are examples of such systems.

3.3.2 Type of transactions

Another way to classify electronic payment systems is regarding the relationship between the merchant and the consumer. According to [1] we have the following categories:
1. Business-to-Consumer (B2C), where the seller is a business organization and the buyer is a consumer.
2. Business-to-Business (B2B), where both the buyer and the seller are business organizations and relations are typically characterized by long-term stability.
3. Consumer-to-Consumer (C2C), where both the seller and the buyer are consumers.

3.3.3 Type of Authorization

When it comes to do electronic payment transactions, a question arises about whether or not an authorization from the Issuer is needed to complete a transaction. The advantage is that the issuer will be aware of the transaction at the moment of occurrence however this has two main disadvantages: the first one requires that the Issuer should be available all the time: if it happens that communication with issuer is cut due to technical problems for example then the transaction will not be completed creating annoyance for both the merchant and the customer. Also we can

add the fact that online authorization is expensive [1] so unless the transaction value is high it is not advisable to use online authorization.
Offline authorization transaction requires only the payer and the payee to be present [1] and verification is insured using smart cards.

3.3.4 Hardware versus software

Depending on how the transaction is conducted, we need to use a special device to play a role in the transaction. In some transactions we can just make use of special software or a special hardware [1].
The security requirement for software based payment is that it must be tamper resistant regarding data present.
If we have to use special hardware we also need it to be tamper proof but this time against double spending when it comes to electronic cash for example [1]

3.3.5 Size of the payment

Electronic payment can also be classified according to the amount of money exchanged.
If large amount of money is exchanged then we talk about macro payment system but if small amount of money is exchanged then we call it micro payment system [1].

3.3.6 Transaction medium

Each electronic payment system has its own specific transaction medium. At the moment, we are all aware of the use of Internet, mobile phone, Point of Sale terminals and pay per view TV [1]. With a remark that use of mobile phones to conduct transaction is currently limited to micro payments.

CHAPTER 4 THE SCYTHER TOOL

The purpose of this chapter is to introduce the reader to the scyther tool [3] that will be used to analyze some parts of the payment protocols that will be introduced later on. An overview of the scyther tool is first given in section 4.1 followed by an example of how to model a protocol in scyther (section 4.2). After that we will interpret the scyther analysis of the protocol (section 4.3).
Note that no manual has been issued at the time of this research regarding scyther since it is still in development [3].

4.1. Introduction

The scyther tool is a program written in C that allows the user to analyze security protocols. My first use of this program was during the course verification of security protocols at Eindhoven University of Technology. It is currently under development by Cas Cremers from the Eindhoven Computer Security Group as part of his PhD thesis [3]. At the moment of writing, the software is in beta 3 version [3].

4.2. Example of use: Needham-Schroeder protocol

In this section we will introduce the reader on how to use scyther through an example. For that purpose we will make use of the Needham-Schroeder protocol.

4.2.1. Needham-Schroeder protocol description

This description of the protocol follows the one made in [4].
A Needham-Schroeder protocol purpose is to allow two parties A and B to exchange their respective nonces in a secure way. It works in the following way (see figure 4.1):

- In the first step, A will generate a random nonce na, concatenate it with his/her identity and then send it to B encrypted with B's public key.
- Upon reception of that encrypted message, B will generate his/her own random nonce nb, concatenate it with A's nonce na and send this new message to A using A's public key.
- To confirm acknowledge of receiving the nonce from the other party, A will then reply by sending back the nonce received from B encrypted with B's public key.

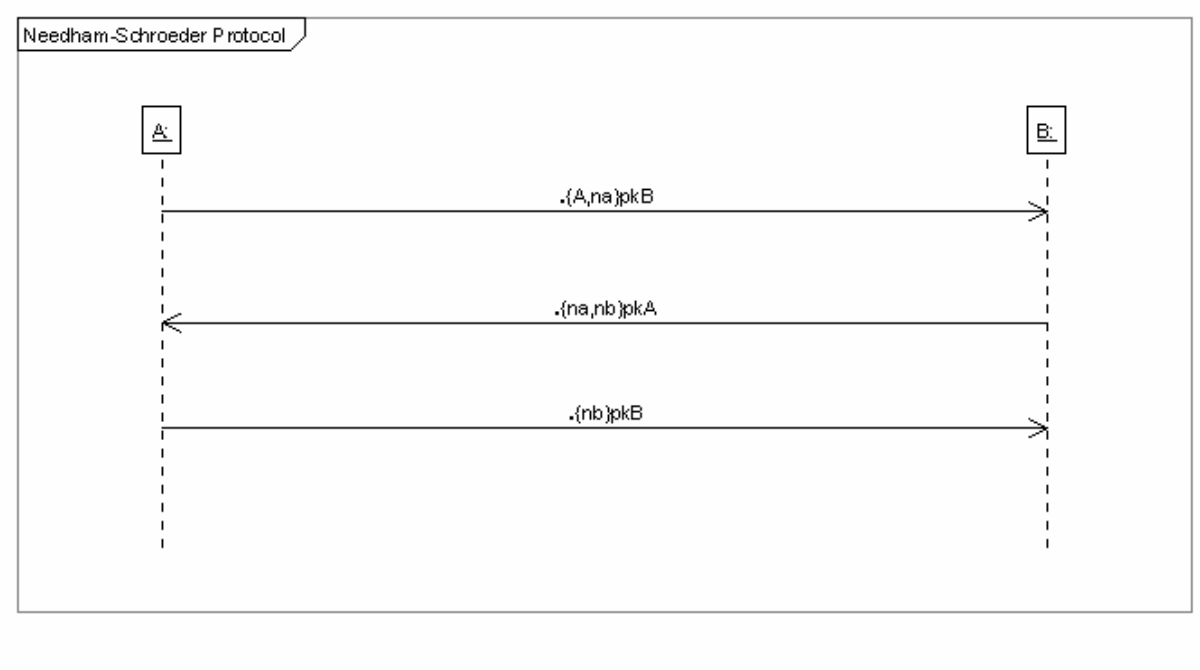

Figure 4.1: Needham-Schroeder protocol

4.2.2. Scyther modeling of Needham-Schroeder protocol

In this section we will show the source code that we will give as input to scyther on order to analyze it. The name of the file is ns3.spdl. [3]

```
/*
 * Needham-Schroeder protocol
 */

// PKI infrastructure

const pk: Function;
secret sk: Function;
inversekeys (pk,sk);

// The protocol description

protocol ns3(I,R)
{
        role I
        {
                const ni: Nonce;
                var nr: Nonce;

                send_1(I,R, {I,ni}pk(R) );
                read_2(R,I, {ni,nr}pk(I) );
                send_3(I,R, {nr}pk(R) );
```



```
            claim_i1(I,Secret,ni);
            claim_i2(I,Secret,nr);
            claim_i3(I,Niagree);
            claim_i4(I,Nisynch);
        }

        role R
        {
            var ni: Nonce;
            const nr: Nonce;

            read_1(I,R, {I,ni}pk(R) );
            send_2(R,I, {ni,nr}pk(I) );
            read_3(I,R, {nr}pk(R) );

            claim_r1(R,Secret,ni);
            claim_r2(R,Secret,nr);
            claim_r3(R,Niagree);
            claim_r4(R,Nisynch);
        }
}

// An untrusted agent, with leaked information

const Eve: Agent;
untrusted Eve;
const ne: Nonce;
compromised sk(Eve);
```

4.2.2. Scyther compilation and results

In this section, we show how to compile the spdl code using scyther and output the result.

```
[s032015@svstud ~]$ scyther ns3.spdl
claim   ns3,R   Nisynch_r4      -       Fail    [at least 1 attack]
claim   ns3,R   Niagree_r3      -       Fail    [at least 1 attack]
claim   ns3,R   Secret_r2       nr      Fail    [at least 1 attack]
claim   ns3,R   Secret_r1       ni      Fail    [at least 1 attack]
claim   ns3,I   Nisynch_i4      -       Ok      [proof of correctness]
claim   ns3,I   Niagree_i3      -       Ok      [proof of correctness]
claim   ns3,I   Secret_i2       nr      Ok      [proof of correctness]
claim   ns3,I   Secret_i1       ni      Ok      [proof of correctness]
```

4.2.3. Attack drawing

By using the command “scytherview ns3.spdl”, we get the following attack drawing. Detailed explanation of this attack will be given in the following section.

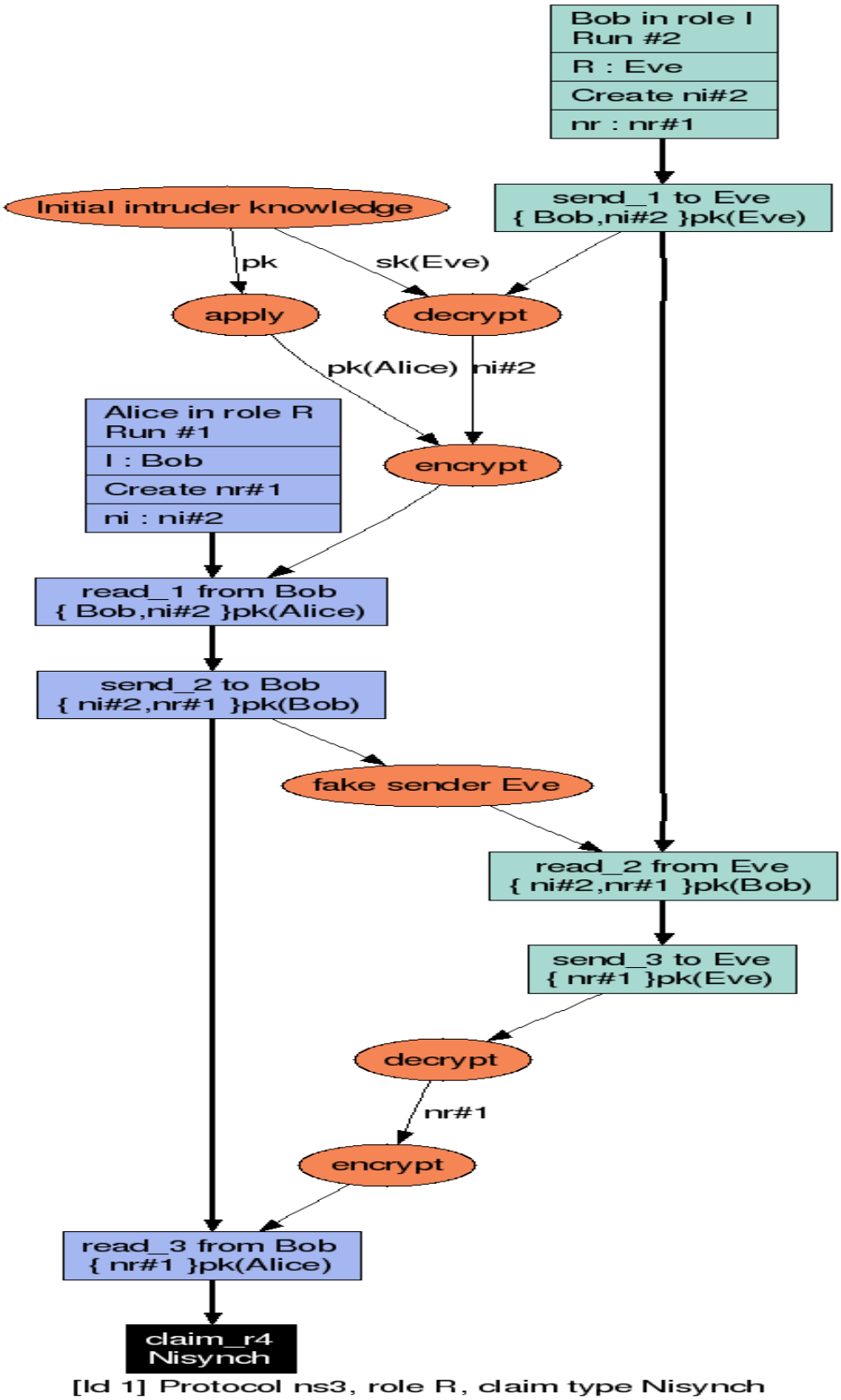

Figure 4.2: Scyther attack drawing on Needham-Schroeder Protocol [3]

4.3. Scyther analysis

Figure 4.3 below shows how the attack is happening using Message Sequence Chart:

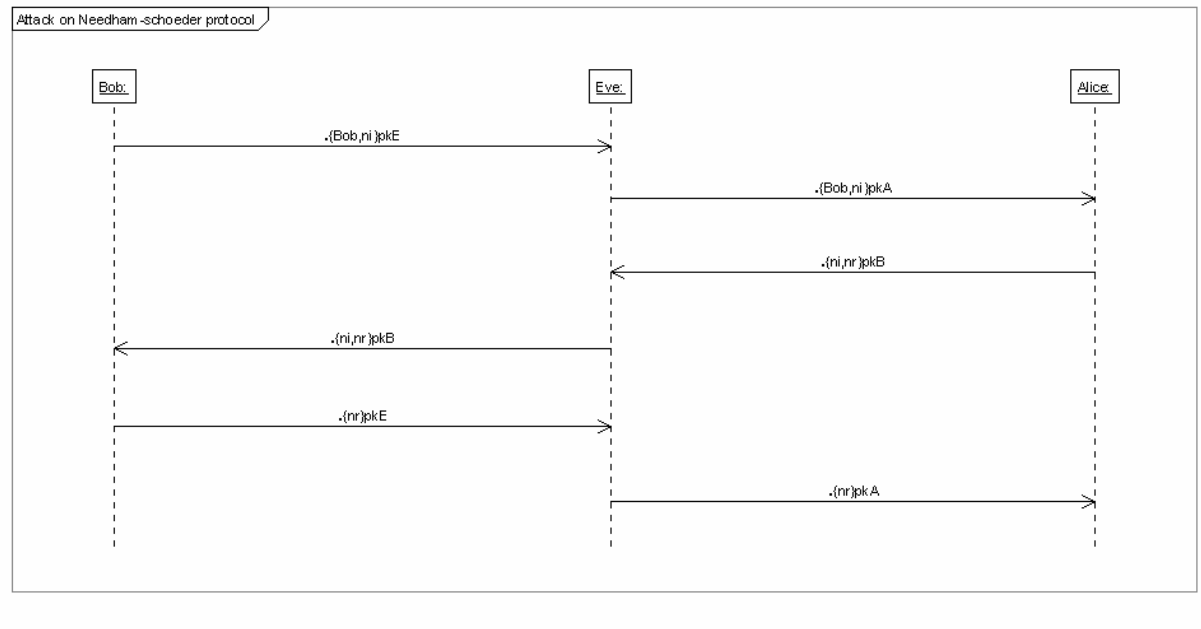

Figure 4.3: Attack drawing on Needham-Schroeder Protocol using MSC

The attack works as follows:

- Bob will send to Eve his name plus a fresh nonce named ni encrypted with the public key of Eve
- Eve will decrypt the message using her secret key, encrypt the recovered message with the public key of Alice and send it to her.
- After decrypting the message received, Alice will reply (thinking that she is talking to Bob) by sending an encrypted message using Bob's public key that contains the nonce ni Bob sent plus a nonce nr created by Alice.
- Eve will just forward the message to Bob.
- Once the message received by Bob, he will reply by sending an encrypted message using Eve's public key that consists of the nonce nr created by Alice.
- Eve will recover the nonce nr then will send it encrypted to Alice using Alice's public key.

The false assumptions that will result from this to Alice is that she thinks she is talking to Bob while in fact she is talking to Eve and also the fact that the nonce nr is secret while it is learnt by Eve from this attack.

CHAPTER 5 EMV OVERVIEW AND SECURITY ANALYSIS

This chapter proposes a security analysis of using EMV cards to conduct secure transactions at the point of sale. An overview of the EMV card payment transaction procedure is first given in section 5.2. Finally, we analyze how the EMV matches the identified requirements (section 5.3).

5.1 Introduction

Conducting Debit/credit card transaction at POS have been subjects to several security shortages. First there was the use of magnetic strips which revealed to be unsuccessful then use if card verification values but the problem remained the same. With the advance in smart cards technologies, Europay MasterCard and Visa decided to join their effort to define the EMV specifications [7, 8, 9, 10] which will be managed by a company named EMVCO.
The main objectives for the introduction of EMV by the card associations were [1]:

- To establish standards for global interoperability that allows a single merchant terminal to be used with cards issued by different card issuing banks and different brands.
- To reduce the number of on-line authorizations, leading to reduce acquiring costs for merchants.
- To reduce the fraud costs associated with counterfeit cards.
- To provide a secure platform to enable off-line cardholder verification.
- To allow card issuers to manage card parameters remotely.

5.2. EMV transaction flow

We will in this section describe the steps shown in figure 5.1.

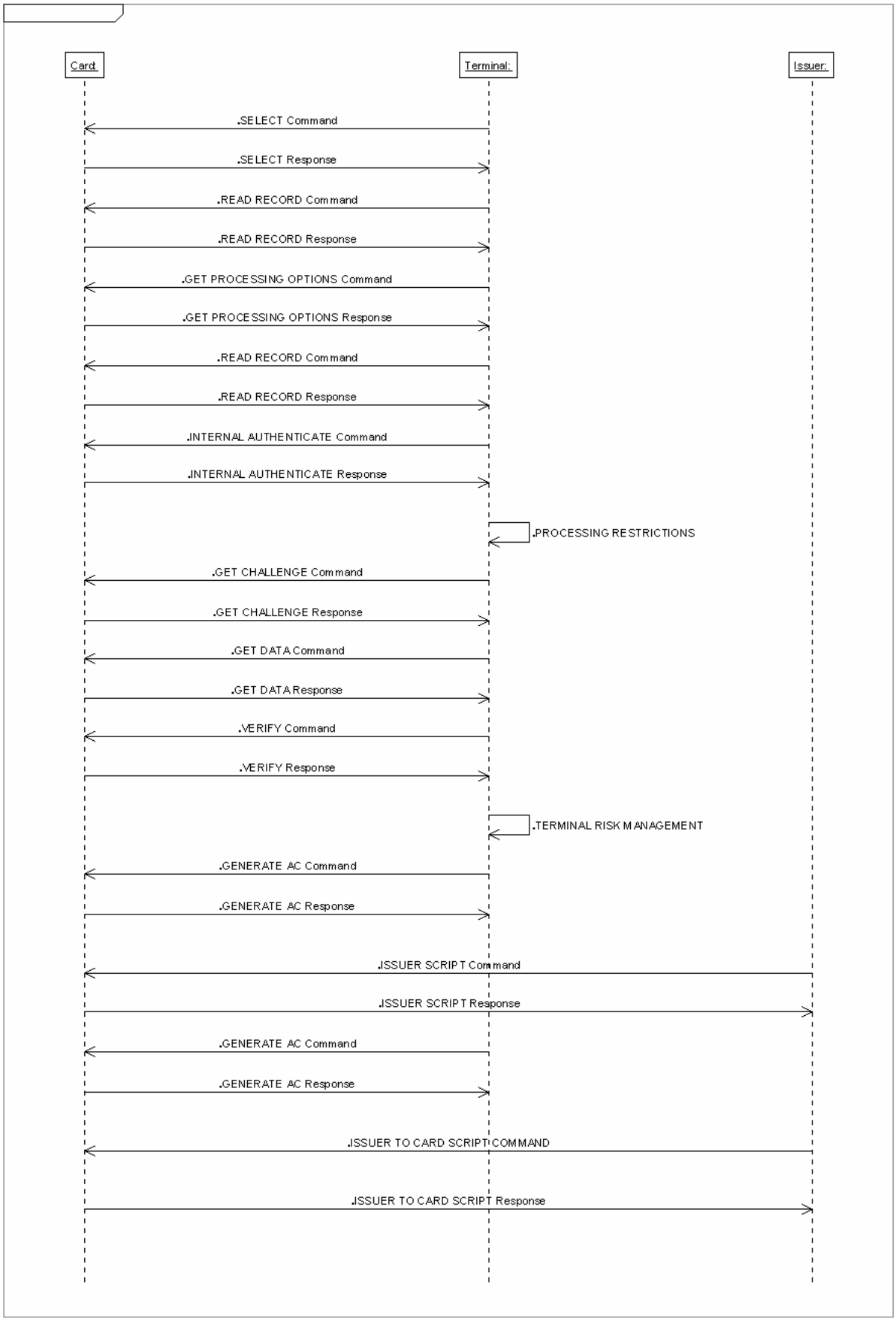

Figure 5.1: EMV transaction (adapted from specifications)

An EMV transaction consists of the following operations:

1. Application selection (SELECT command and response): In this phase, the terminal while interacting with the card will try to determine which applications are commonly supported. Three possibilities can come up from this application selection: the first one is that there is no application in common and hence the transaction will be terminated. The second case is that there is only one application in common and in that case depending on each terminal capability it will be either selected automatically or prompted for cardholder approval: in the case, the cardholder refuses to use that application in common then the transaction is terminated. The last case is that there is multiple applications in common then the terminal will either select the application with higher priority or prompted to the cardholder with the list of applications classified by priority order. If the cardholder refuses any of the application then transaction terminated.
2. Payment initiation: this phase consists of the following. First, initialization of the transaction context followed by a start of the EMV card transaction and finally reading the card data and process restrictions.
3. Card authentication: used to authenticate the card to the terminal. More details in section 5.3.1.
4. Cardholder verification: used to verify the identity of the cardholder. More details in section 5.3.2.
5. Terminal risk management: The terminal will check during this phase the floor limits, select a random transaction and do a velocity checking. The purpose is to ensure that transactions could go online in certain period of time in order to prevent offline attacks that might go undetected.
6. Online processing: In this phase, a transaction will be handled online.
7. Transaction completion: Once all the steps above completed successfully, the transaction will be completed.

5.3. Security mechanisms

5.3.1. Card-terminal Authentication

We have a choice here between three types of card authentication methods: SDA, DDA and CDA. We note here that all cards support SDA while DDA and CDA are optional.
For SDA authentication (see figure 5.2), the card does not have its own key pair. In order to do SDA, the terminal will do the following:

- Retrieve Issuer public key from the card signed by the certification authority
- Retrieved the static data application signed by the Issuer secret key.

Figure 5.2: Static data authentication

For DDA authentication (see figure 5.3), the card has this time its own key pair so now in order to do DDA the following steps will occur:

- Retrieve Issuer public key from the card signed by the certification authority
- Retrieve the Card public key signed by the Issuer secret key.
- Terminal sends a challenge to the card using INTERNAL AUTHENTICATE COMMAND.
- The card replies by concatenating its signed static data application by Issuer with the data from terminal's command and sign the block with its secret key

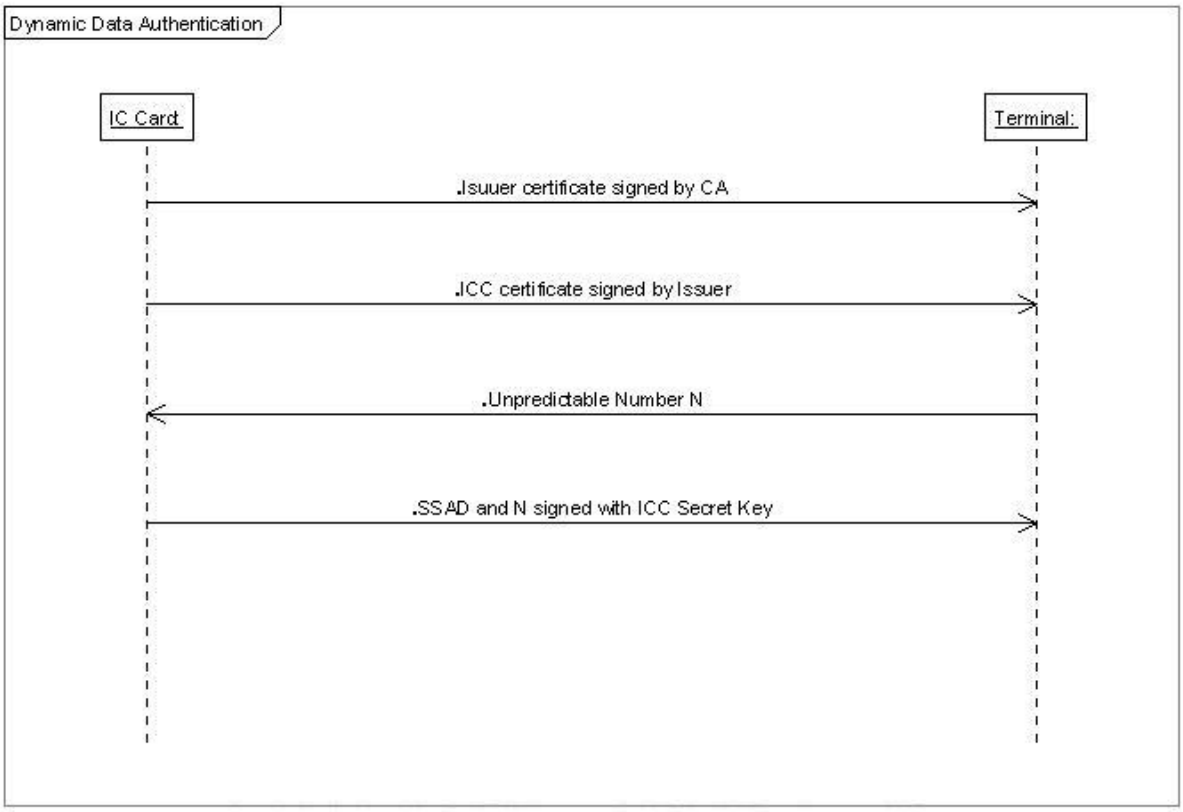

Figure 5.3: Dynamic Data Authentication

5.3.2. Cardholder Verification

This is based on PIN verification. It could be done either online or offline depending on the card and the terminal capabilities.

5.3.3. Application cryptograms

The purpose of these application cryptograms is to guarantee integrity and origin authentication of transaction messages exchanged between the card and the Issuer. The EMV specifications mentioned four types of application cryptograms namely:

- Transaction certificate: generated when the transaction is approved offline.
- Application authentication cryptogram: generated when the transaction is rejected offline.
- Authorization Request cryptogram: generated when the transaction need to go online and contact the Issuer.
- Authorization Response cryptogram: it is the response from the Issuer on whether the transaction is approved or rejected which respectively generates TC and AAC.

5.4. Scyther modeling

- Static data authentication modeling:

In this part we show the input source code for Scyther regarding SDA.

```
/*
 * Static Data Authentication Protocol
 */

// PKI infrastructure
const pk: Function;
const hash: Function;
secret sk: Function;
secret unhash: Function;
inversekeys (pk,sk);
inversekeys (hash,unhash);
// The protocol description

protocol sda(ICC,I,T,CA)
{
	role ICC
	{
		const header, signedstaticdata, footer: Nonce;

		send_1(ICC,T,{pk(I)}sk(CA));
```



```
            send_2(ICC,T,{header, signedstaticdata, footer}sk(I));
            claim_icc1(ICC, Nisynch);
            claim_icc2(ICC, Niagree);
        }

        role T
        {
            var header, signedstaticdata, footer: Nonce;

            read_1(ICC,T,{pk(I)}sk(CA));
            read_2(ICC,T,{header, signedstaticdata, footer}sk(I));
            claim_t1(T, Nisynch);
            claim_t2(T, Niagree);
        }

}

// The agents in the system

const ICC,I,T,CA: Agent;

// An untrusted agent, with leaked information

const Eve: Agent;
untrusted Eve;
const ne: Nonce;
compromised sk(Eve);
```

- Dynamic Data authentication modeling :

In this part we show the input source code for Scyther regarding DDA.

```
/*
 * Dynamic Data Authentication Protocol
 */

// PKI infrastructure
const pk: Function;
const hash: Function;
secret sk: Function;
secret unhash: Function;
inversekeys (pk,sk);
inversekeys (hash,unhash);
// The protocol description

protocol dda(ICC,I,T,CA)
{
        role ICC
        {
```



```
            const header, signedstaticdata, footer: Nonce;
            var unpredictablenumber: Nonce;

            send_1(ICC,T,{pk(I)}sk(CA));
            send_2(ICC,T,{pk(ICC)}sk(I));
            read_3(T,ICC,unpredictablenumber);

        send_4(ICC,T,{header,signedstaticdata,unpredictablenumber,footer}sk(ICC));
            claim_icc1(ICC,Nisynch);
            claim_icc2(ICC,Niagree);
        }

        role T
        {
            var header,signedstaticdata,footer: Nonce;
            const unpredictablenumber: Nonce;

            read_1(ICC,T,{pk(I)}sk(CA));
            read_2(ICC,T,{pk(ICC)}sk(I));
            send_3(T,ICC,unpredictablenumber);

        read_4(ICC,T,{header,signedstaticdata,unpredictablenumber,footer}sk(ICC));
            claim_t1(T,Nisynch);
            claim_t2(T,Niagree);
        }

}

// The agents in the system

const ICC,I,T,CA: Agent;

// An untrusted agent, with leaked information

const Eve: Agent;
untrusted Eve;
const ne: Nonce;
compromised sk(Eve);
```

- Attack drawing on SDA :

The figure 5.4 shows the output of the SDA attack detected by Scyther. Details about the SDA attack will be left for Scyther analysis section.

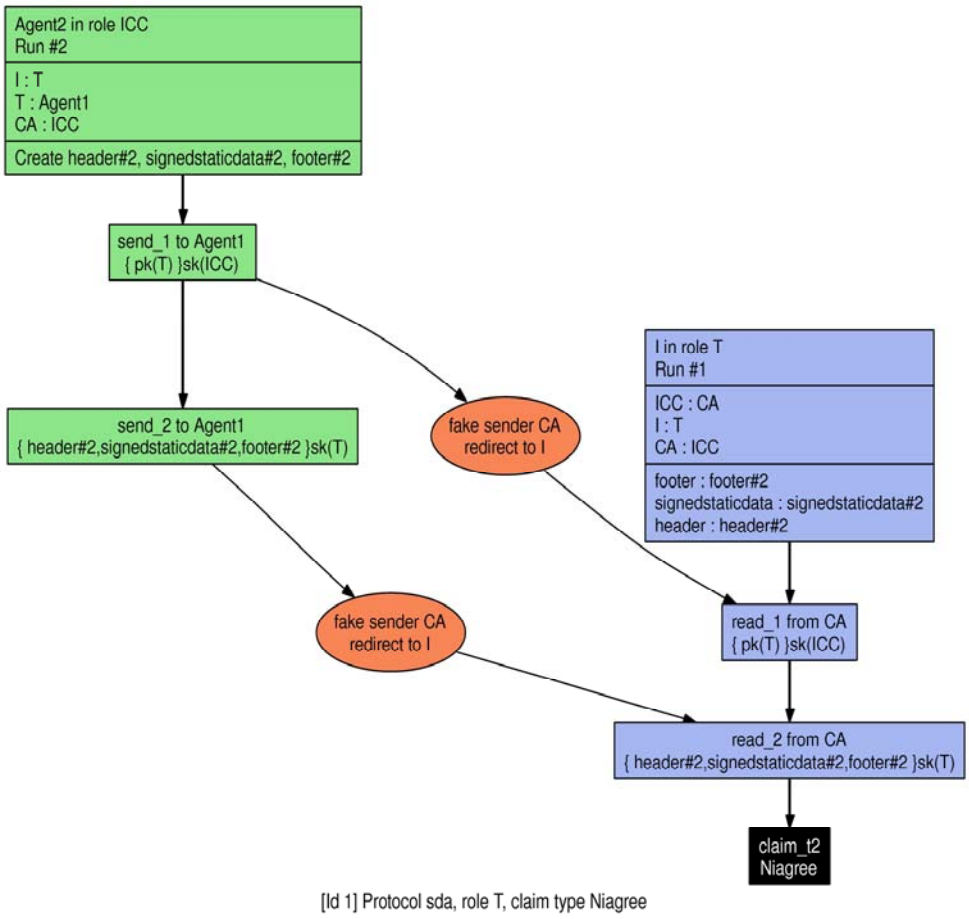

Figure 5.4: Scyther attack on SDA

- Attack drawing on DDA:

The figure 5.5 shows the output of the DDA attack detected by Scyther. Details about the DDA attack will be left for Scyther analysis section.

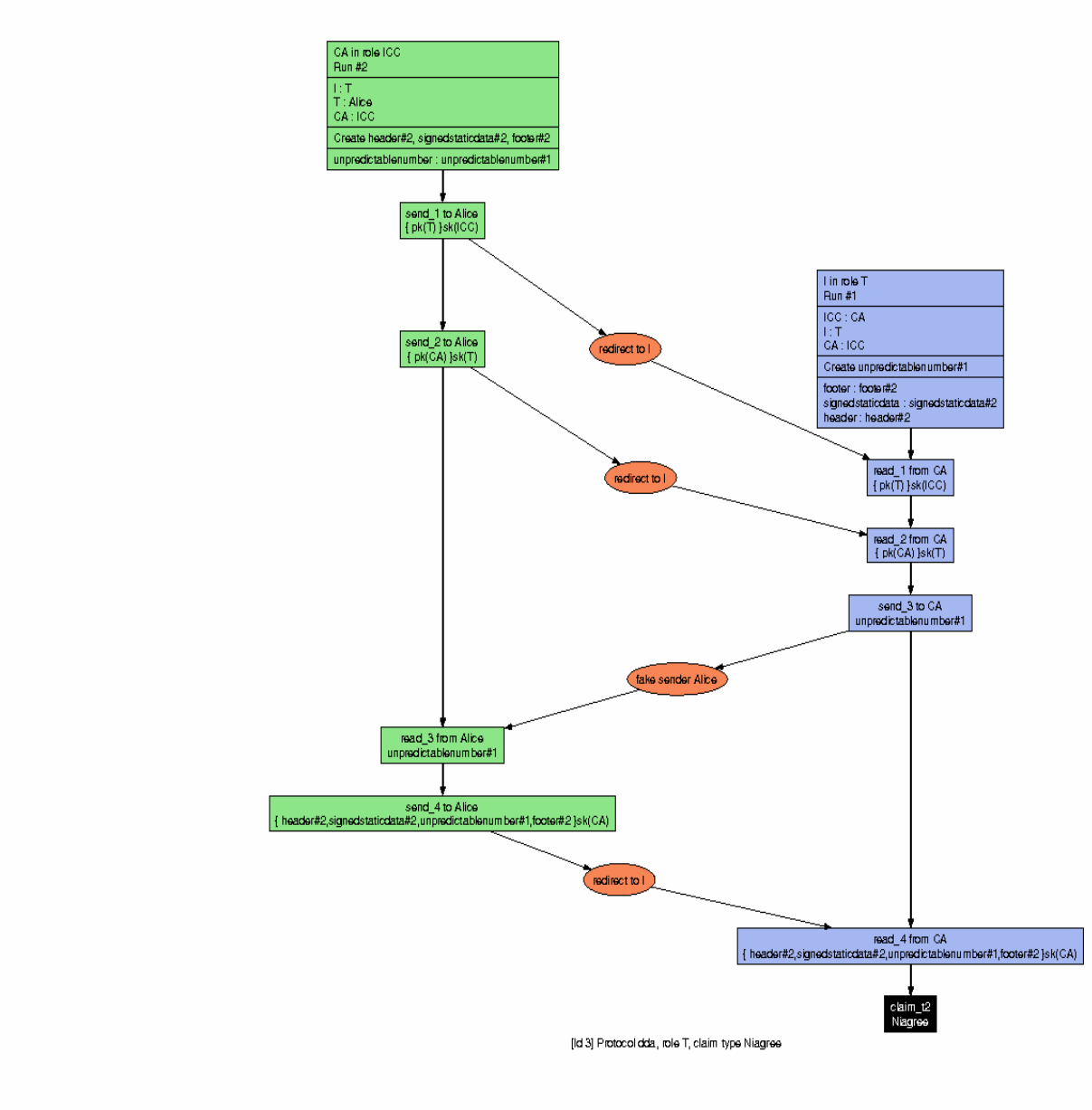

Figure 5.5: scyther attack drawing on DDA

5.5. Security Analysis

The security analysis performed in this section analyzes whether the security requirements stated in Section 2.3 are fulfilled by the EMV transaction.

5.5.1 Authentication

According to our manual analysis, we can confirm that EMV insure entity authentication of the card to the terminal is by the use of DDA and CDA. This is done by the terminal when it verifies that the card public key is signed by the CA and by verifying the response to the INTERNAL AUTHENTICATE command sent to the card by the terminal. However this security requirement is not met by SDA since the data

on the card is static so it can just be copied to a fake card and used on behalf of the legitimate user.
When it comes to the cardholder the authentication is met by the use of the PIN code since the number of unsuccessful PIN codes is limited unless an attacker was either lucky or got his hands on the PIN code.
The issuer can verify that the cardholder sent a payment instruction based on the ARQC and TC.
The cardholder can verify it identity to the issuer by use of PIN code.
However, EMV does not provide mechanisms to authenticate the merchant terminal to the cardholder or to the issuer. Scyther has pointed out that in the attacks drawings in figure 5.2 and 5.3.

5.5.2 Confidentiality

According to our manual analysis, we can confirm that EMV insure that the PIN remains confidential. If the card has computational power then the terminal will use the card public key to encrypt the PIN and the card will decipher the PIN to verify its correctness.
However EMV does not provide any order information encryption and hence it can be read by the terminal, the acquirer, and the issuer.

5.5.3 Integrity

According to our manual analysis, we can confirm that EMV insure that the card payment authorization sent to the issuer is protected by the use of a MAC. In fact, since the card and the issuer share the secret key they can verify that the message has not been altered.

5.5.4 Non-repudiation

According to our manual analysis, we can confirm that EMV insure the cardholder payment authorization has indeed been sent by the cardholder to the Issuer since when the cardholder will type the correct PIN code after getting the price of the goods a TC and an ARQC will be sent by the car to the issuer.

5.6. Scyther Analysis

5.6.1. SDA attack

The main problem with SDA is that it deals only with static data residing on the card. As scyther pointed out in Figure 5.4, an attacker eavesdrop the two messages sent by the card and send them on behalf of the cardholder later on in to impersonate the cardholder.

5.6.2. DDA attack

The attack shown in Figure 5.5 is a bit more complicate since now unlike we have interaction between terminal and card so the attack detected by the scyther for SDA will not work in this case in DDA. The attack seems to be as an attacker who communicates with a terminal on behalf of the cardholder and when it comes to getting the challenge, the attacker will forward it to the cardholder as if it is coming from the terminal directly to the cardholder the attacker will then record the response and sends it to the terminal he is communicating to. In other words, the genuine cardholder is used as an oracle by the attacker.

CHAPTER 6 CPA OVERVIEW AND SECURITY ANALYSIS

This chapter proposes a security analysis of using CPA cards to conduct secure transactions at the point of sale. An overview of the CPA card payment transaction procedure is first given in section 6.2 followed by a description of the security mechanisms (section 6.3). We will show how to model parts of the CPA protocol in scyther (section 6.4) then we analyzes how the CPA matches the identified requirements (section 6.4) and finally we conclude with the scyther analysis (section 6.5).

6.1 Introduction

This chapter deals with, the Common Payment Application Specification (CPA) [6]: it defines the data elements and functionality for an application that complies with the EMV Common Core Definitions (CCD). It focuses on the functions performed by the integrated circuit card (ICC) and the interaction between the ICC and terminal at the point of transaction.
The objectives of the Common Payment Application Specification are to [6]:

- Describe the functionality of a CCD-compliant implementation of EMV to ease vendor development efforts
- Specify a core set of functionalities that issuers can rely on having available in every implementation of CPA
- Specify an implementation that can be personalized with the same data elements to meet the business requirements of multiple payment systems

Since CPA is based on EMV and CCD, we will refer to the previous chapter on EMV for common functionalities and we will introduce the extra ones specific to CPA.

6.2. EMV transaction flow

We will in this section describe the steps shown in figure 6.1.

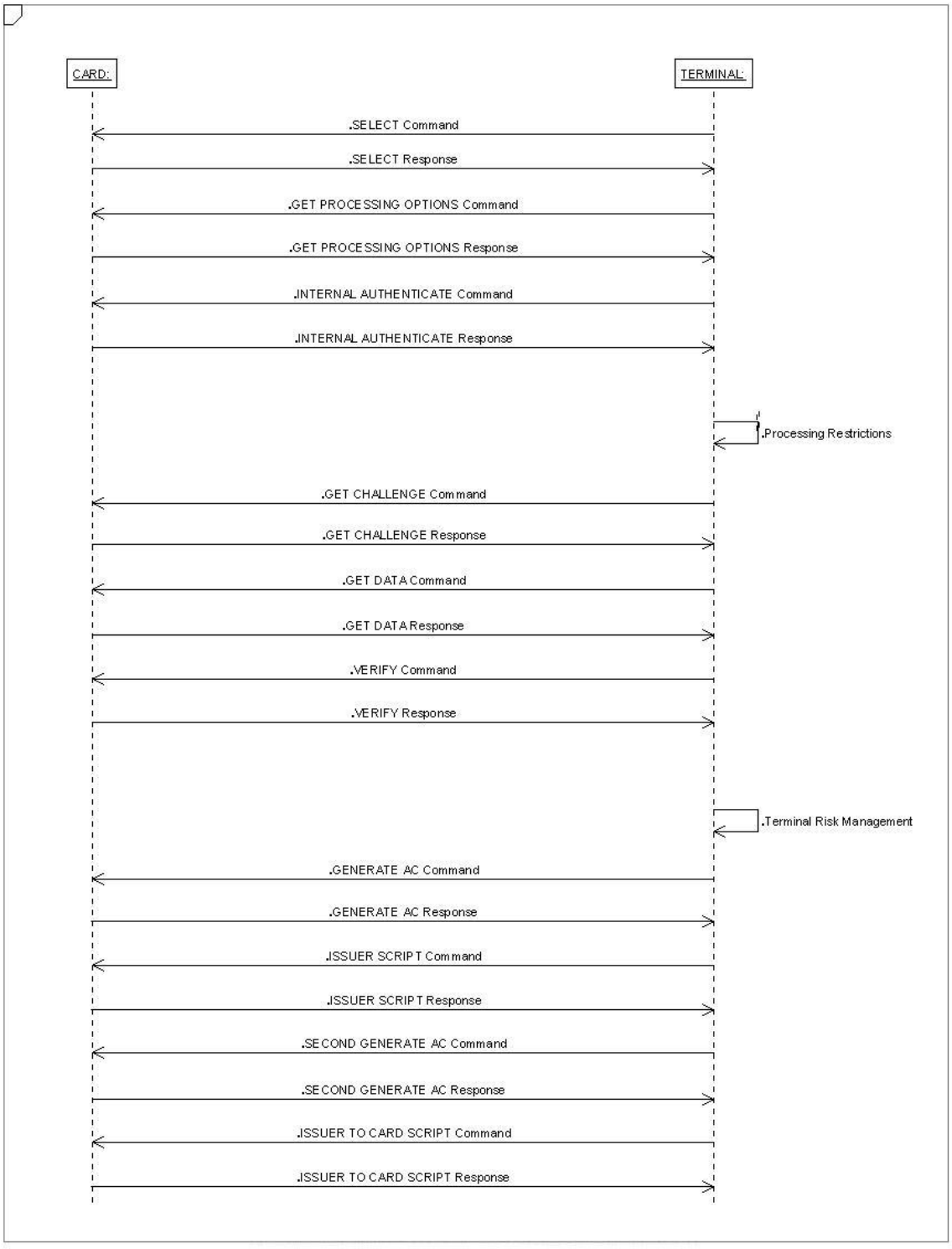

Figure 6.1: CPA transaction

A CPA transaction consists of the following operations:

1. Application selection: In this phase, the terminal while interacting with the card will try to determine which applications are commonly supported.

2. Payment initiation: this phase consists of the following. First, initialization of the transaction context followed by a start of the EMV card transaction and finally reading the card data and process restrictions.
3. Card authentication: used to authenticate the card to the terminal. More details in section 6.3.1.
4. Cardholder verification: used to verify the identity of the cardholder. More details in section 6.3.2.
5. Terminal risk management: The terminal will check during this phase the floor limits, select a random transaction and do a velocity checking.
6. Online processing: In this phase, a transaction will be handled online.
7. Transaction completion: Once all the steps above completed successfully, the transaction will be completed.

6.3. Security mechanisms

6.3.1. Card-terminal Authentication

Same as for EMV, we have a choice here between three types of card authentication methods: SDA, DDA and CDA. We note here that all cards support SDA while DDA and CDA are optional.
For SDA authentication (see figure 6.2), the card does not have its own key pair. In order to do SDA, the terminal will do the following:

- Retrieve Issuer public key from the card signed by the certification authority
- Retrieved the static data application signed by the Issuer secret key.

Figure 6.2: Static data authentication

For DDA authentication (see figure 6.3), the card has this time its own key pair so now in order to do DDA the following steps will occur:

- Retrieve Issuer public key from the card signed by the certification authority
- Retrieve the Card public key signed by the Issuer secret key.
- Terminal sends a challenge to the card using INTERNAL AUTHENTICATE COMMAND.
- The card replies by concatenating its signed static data application by Issuer with the data from terminal's command and sign the block with its secret key

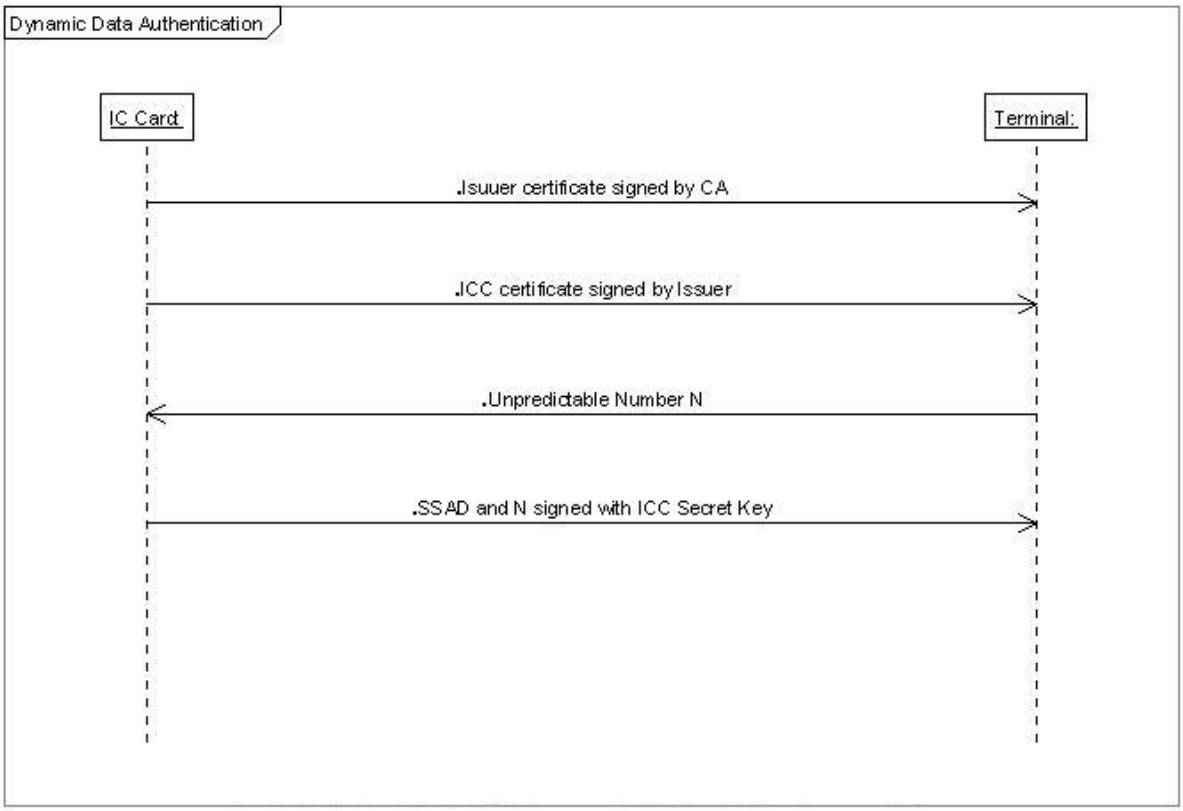

Figure 6.3: Dynamic Data Authentication

6.3.2. Cardholder Verification

This is based on PIN verification. It could be done either online or offline.

6.3.3. Application cryptograms

The purpose of these application cryptograms is to guarantee integrity and origin authentication of transaction messages exchanged between the card and the Issuer. The CPA specifications mentioned four types of application cryptograms namely:

- Transaction certificate: generated when the transaction is approved offline.
- Application authentication cryptogram: generated when the transaction is rejected offline.
- Authorization Request cryptogram: generated when the transaction need to go online and contact the Issuer.
- Authorization Response cryptogram: it is the response from the Issuer on whether the transaction is approved or rejected which respectively generates TC and AAC.

6.4. Scyther modeling

- Static data authentication modeling:

In this part we show the input source code for Scyther regarding SDA.

```
/*
 * Static Data Authentication Protocol
 */

// PKI infrastructure
const pk: Function;
const hash: Function;
secret sk: Function;
secret unhash: Function;
inversekeys (pk,sk);
inversekeys (hash,unhash);
// The protocol description

protocol sda(ICC,I,T,CA)
{
	role ICC
	{
		const header, signedstaticdata, footer: Nonce;

		send_1(ICC,T,{pk(I)}sk(CA));
		send_2(ICC,T,{header, signedstaticdata, footer}sk(I));
		claim_icc1(ICC, Nisynch);
		claim_icc2(ICC, Niagree);
	}

	role T
	{
		var header, signedstaticdata, footer: Nonce;

		read_1(ICC,T,{pk(I)}sk(CA));
		read_2(ICC,T,{header, signedstaticdata, footer}sk(I));
		claim_t1(T, Nisynch);
		claim_t2(T, Niagree);
	}

}

// The agents in the system

const ICC,I,T,CA: Agent;

// An untrusted agent, with leaked information
```



```
const Eve: Agent;
untrusted Eve;
const ne: Nonce;
compromised sk(Eve);
```

- Dynamic Data authentication modeling :

In this part we show the input source code for Scyther regarding DDA.

```
/*
 * Dynamic Data Authentication Protocol
 */

// PKI infrastructure
const pk: Function;
const hash: Function;
secret sk: Function;
secret unhash: Function;
inversekeys (pk,sk);
inversekeys (hash,unhash);
// The protocol description

protocol dda(ICC,I,T,CA)
{
	role ICC
	{
		const header, signedstaticdata, footer: Nonce;
		var unpredictablenumber: Nonce;

		send_1(ICC,T,{pk(I)}sk(CA));
		send_2(ICC,T,{pk(ICC)}sk(I));
		read_3(T,ICC,unpredictablenumber);

	send_4(ICC,T,{header,signedstaticdata,unpredictablenumber,footer}sk(ICC));
		claim_icc1(ICC,Nisynch);
		claim_icc2(ICC,Niagree);
	}

	role T
	{
		var header,signedstaticdata,footer: Nonce;
		const unpredictablenumber: Nonce;

		read_1(ICC,T,{pk(I)}sk(CA));
		read_2(ICC,T,{pk(ICC)}sk(I));
		send_3(T,ICC,unpredictablenumber);

	read_4(ICC,T,{header,signedstaticdata,unpredictablenumber,footer}sk(ICC));
```



```
            claim_t1(T,Nisynch);
            claim_t2(T,Niagree);
        }

}

// The agents in the system

const ICC,I,T,CA: Agent;

// An untrusted agent, with leaked information

const Eve: Agent;
untrusted Eve;
const ne: Nonce;
compromised sk(Eve);
```

- Attack drawing on SDA :

The figure 6.4 shows the output of the SDA attack detected by Scyther. Details about the SDA attack will be left for Scyther analysis section.

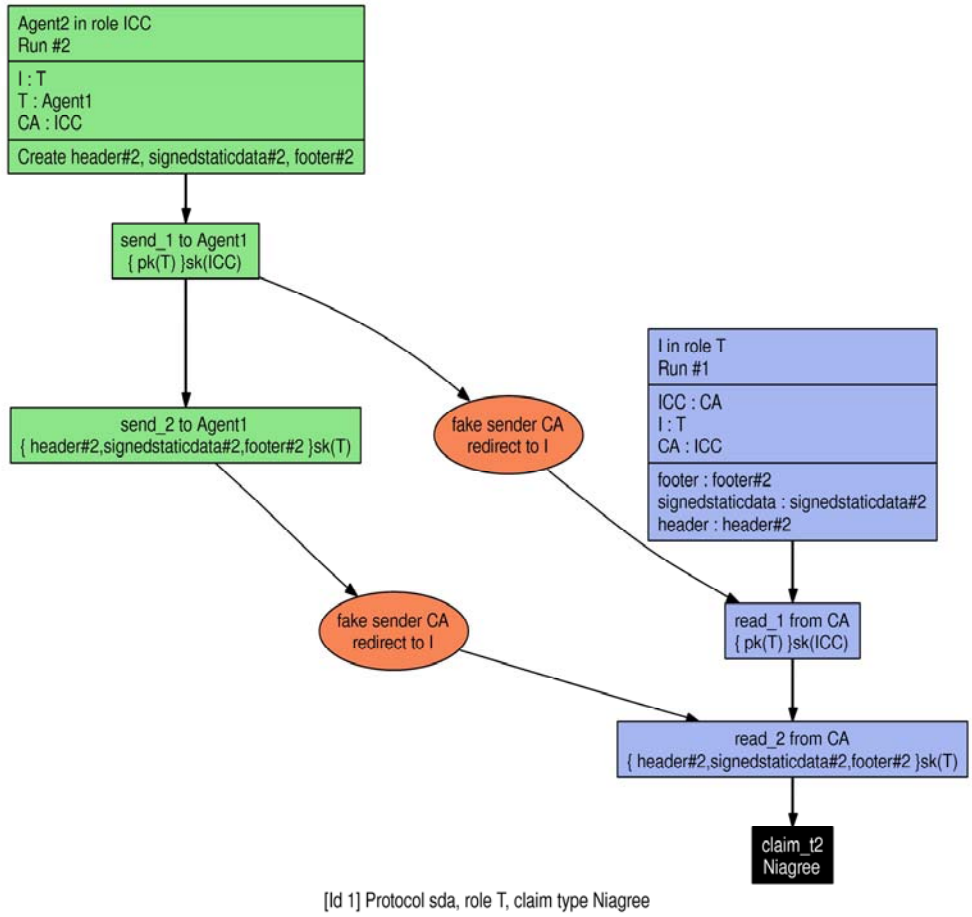

Figure 6.4: Scyther attack on SDA

- Attack drawing on DDA:

The figure 6.5 shows the output of the DDA attack detected by Scyther. Details about the DDA attack will be left for Scyther analysis section.

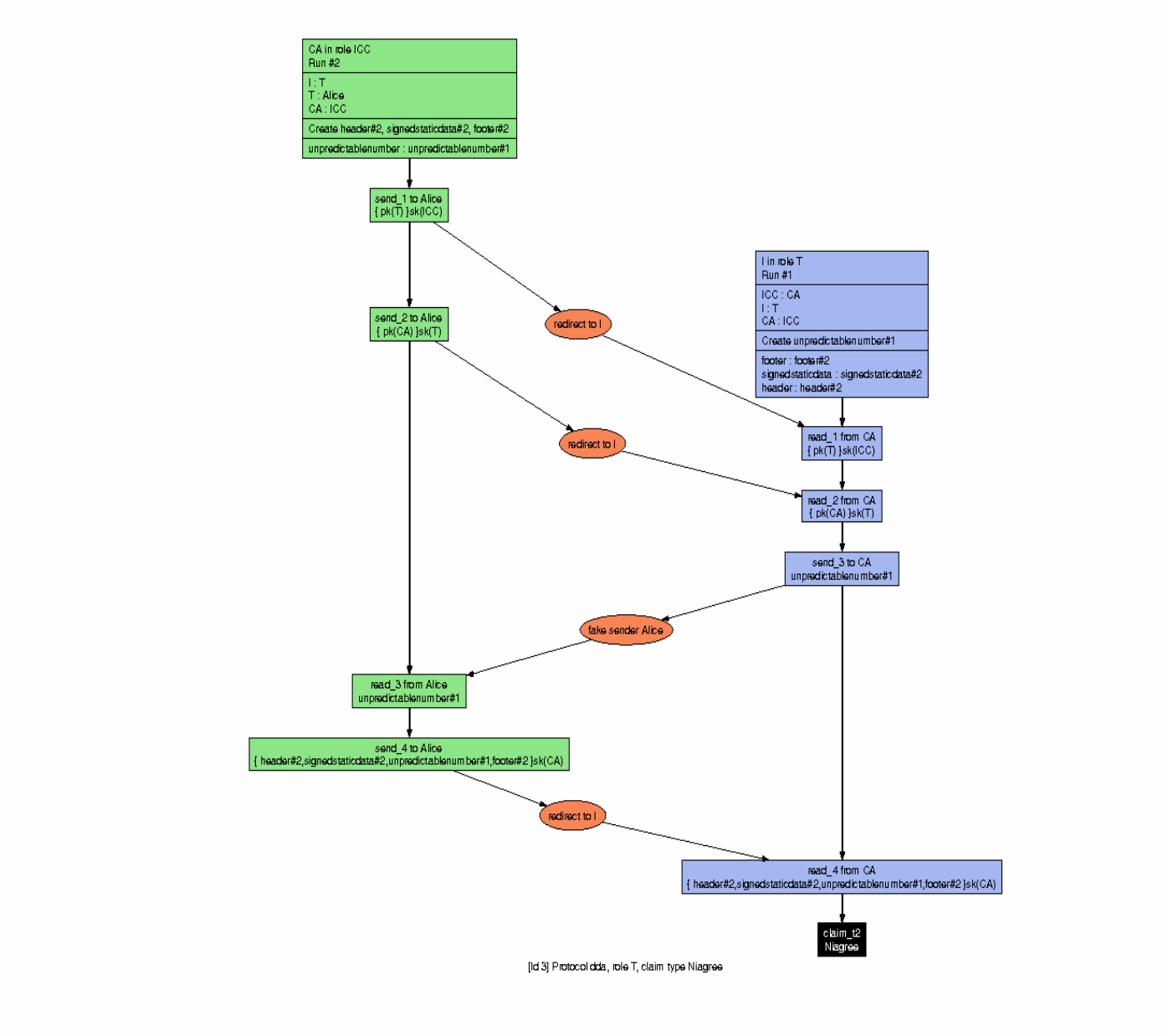

Figure 6.5: scyther attack drawing on DDA

6.5. Security Analysis

The security analysis performed in this section analyzes whether the security requirements stated in Section 2.3 are fulfilled by the CPA transaction.

6.5.1 Authentication

According to our manual analysis, we can confirm that CPA insures entity authentication of the card to the terminal is by the use of DDA and CDA. This is done by the terminal when it verifies that the card public key is signed by the CA and by verifying the response to the INTERNAL AUTHENTICATE command sent to the card by the terminal. However this security requirement is not met by SDA since the data on the card is static so it can just be copied to a fake card and used on behalf of the legitimate user.
When it comes to the cardholder the authentication is met by the use of the PIN code since the number of unsuccessful PIN codes is limited unless an attacker was either lucky or got his hands on the PIN code.
The issuer can verify that the cardholder sent a payment instruction based on the ARQC and TC.
The cardholder can verify it identity to the issuer by use of PIN code.
However, CPA does not provide mechanisms to authenticate the merchant terminal to the cardholder or to the issuer. Scyther has pointed out that in the attacks drawings in figure 5.2 and 5.3.

6.5.2 Confidentiality

According to our manual analysis, we can confirm that CPA insure that the PIN remains confidential. If the card has computational power then the terminal will use the card public key to encrypt the PIN and the card will decipher the PIN to verify its correctness.
However CPA does not provide any order information encryption and hence it can be read by the terminal, the acquirer, and the issuer.

6.5.3 Integrity

According to our manual analysis, we can confirm that CPA insure that the card payment authorization sent to the issuer is protected by the use of a MAC. In fact, since the card and the issuer share the secret key they can verify that the message has not been altered.

6.5.4 Non-repudiation

According to our manual analysis, we can confirm that CPA insure the cardholder payment authorization has indeed been sent by the cardholder to the Issuer since when the cardholder will type the correct PIN code after getting the price of the goods a TC and an ARQC will be sent by the car to the issuer.

6.6. Scyther Analysis

6.6.1. SDA attack

The main problem with SDA is that it deals only with static data residing on the card. As scyther pointed out in Figure 6.4, an attacker eavesdrop the two messages sent by the card and send them on behalf of the cardholder later on in to impersonate the cardholder.

6.6.2. DDA attack

The attack shown in Figure 6.5 is a bit more complicate since now unlike we have interaction between terminal and card so the attack detected by the scyther for SDA will not work in this case in DDA. The attack seems to be as an attacker who communicates with a terminal on behalf of the cardholder and when it comes to getting the challenge, the attacker will forward it to the cardholder as if it is coming from the terminal directly to the cardholder the attacker will then record the response and sends it to the terminal he is communicating to. In other words, the genuine cardholder is used as an oracle by the attacker.

CHAPTER 7 VISA'S 3-D SECURE OVERVIEW AND SECURITY ANALYSIS

This chapter proposes a security analysis of using 3-D SECURE to conduct credit cards payment transactions online. An overview of the 3-D architecture (section 7.2) is first given. Then, we identify the security requirements (section 7.3) followed by a description of the 3-D SECURE protocol (section 7.4) and finally we analyze how 3-D SECURE matches them (section 7.5). One important point here is that due to that fact that the protocol specifications of VISA 3-D SECURE [17] are not publicly available (confidential data) hence we can not model using scyther in this thesis so we will just focus on the manual analysis in this chapter and leave the scyther modeling as future work.
Please note that a similar work has been done in [12].

7.1. Introduction

Nowadays the majority of e-commerce websites, which provides on line payment services, makes use of SSL (Secure Socket Layer) /TLS (Transport Layer Security) in order to protect sensitive information such as the credit card number. However these protocols were not conceived as of their origin to make secure payment but to ensure the confidentiality of the information exchanged between two entities. This method has various disadvantages:

- The transactions are done only between two entities (the payer and the payee) whereas generally, the banks would have to be involved.
- Risks related to the model of payment with credit card: with SSL and TLS, the card holders take the risk, that the merchant can make use of credit card numbers stored on his/her servers, and the merchants take the risk that a customer uses a fake credit card number by using credit card number generator.
- The merchant is aware of the customer's credit card number, information that only the banks should be informed of it.
- The authentication is not mandatory and the customer could be impersonated.

These protocols thus prove unsatisfactory when confronted with the security requirements necessary for protected payments (see section 2.1).
New protocols have been designed in order to counter those deficiencies and VISA's 3D SECURE is one of those protocols.

7.2. The 3-domain Architecture

In this section we will introduce the three domains plus the entities involved in the transaction

7.2.1. The three domains

The name 3-D in 3-D SECURE comes from the fact that we it is composed of three domains which are the Issuer Domain, the Acquirer Domain and the Interoperability domain. Their description is as follows [15]:

- Issuer Domain: Domain of interactions between the Issuer and the cardholder(s).
- Acquirer Domain: Domain of interactions between the Acquirer and the merchant(s).
- Interoperability Domain: Domain of interactions between the Issuer(s) and the Acquirer(s).

7.2.2. Entities involved

As pointed out in section 3.2, an electronic payment model consists of four parties: cardholder, merchant, issuer and acquirer. Their roles are defined in the following way [15]:

- Cardholder: He/she will do online shopping on the Merchant's website and provides information for authentication whenever asked.
- Merchant: He/she will be selling goods on the website to the cardholder and via MPI will be conducting payment authentication.
- Issuer: This is the issuer of the cardholder's VISA card needed to conduct transactions as well as two other tasks: enrolment procedure and authentication.
- Acquirer: This is the merchant's bank. Its task is to get payment authorizations from the merchant and forward them to the authorization authority as well as submit completed transactions to the settlement system.

7.3. Security requirements for 3-D SECURE

The following security requirements have to be met when using 3-D SECURE [17]:

- Cardholder authentication: we need to get a confirmation of the identity of the cardholder before any transaction.
- Confidentiality of payment and order information: these are very sensitive date so we need to insure that only the authorized parties can see them.
- Integrity of authentication response: this information sent by the Issuer need to be protected against alteration.

7.4. The 3-D SECURE protocol

The 3-D SECURE protocol consists of two phases: enrollment phase in which the cardholder will browse to the 3-D SECURE enrollment website in order to register itself to become part of any future 3-D secure based transaction. The second phase is called the transaction phase where the cardholder will do shopping via the merchant's website and the issuer will be involved in every step of the transaction.

7.4.1. Enrollment phase

This phase consists of four steps. We will follow the definitions given by [17]:
Step 1: The cardholder connects to the Issuer's 3-D SECURE enrollment web page.
Step 2: The cardholder supplies all necessary information needed for the enrollment such as payment card number in order to validate the identity of the cardholder and establish a shared secret.
Step 3: The issuer or third party will then validate the information provided by the cardholder in order to make sure that the cardholder in entitled to use his/her card during transaction.
Step 4: The information gathered is then stored to be used later on during authentication relative to a 3-D SECURE transaction.

7.4.2. Transaction phase

Having successfully enrolled in the previous phase, the cardholder can now participate in any 3-D SECURE transaction.

Figure 6.1 shows how 3-D SECURE transaction takes place following definitions given in [16]

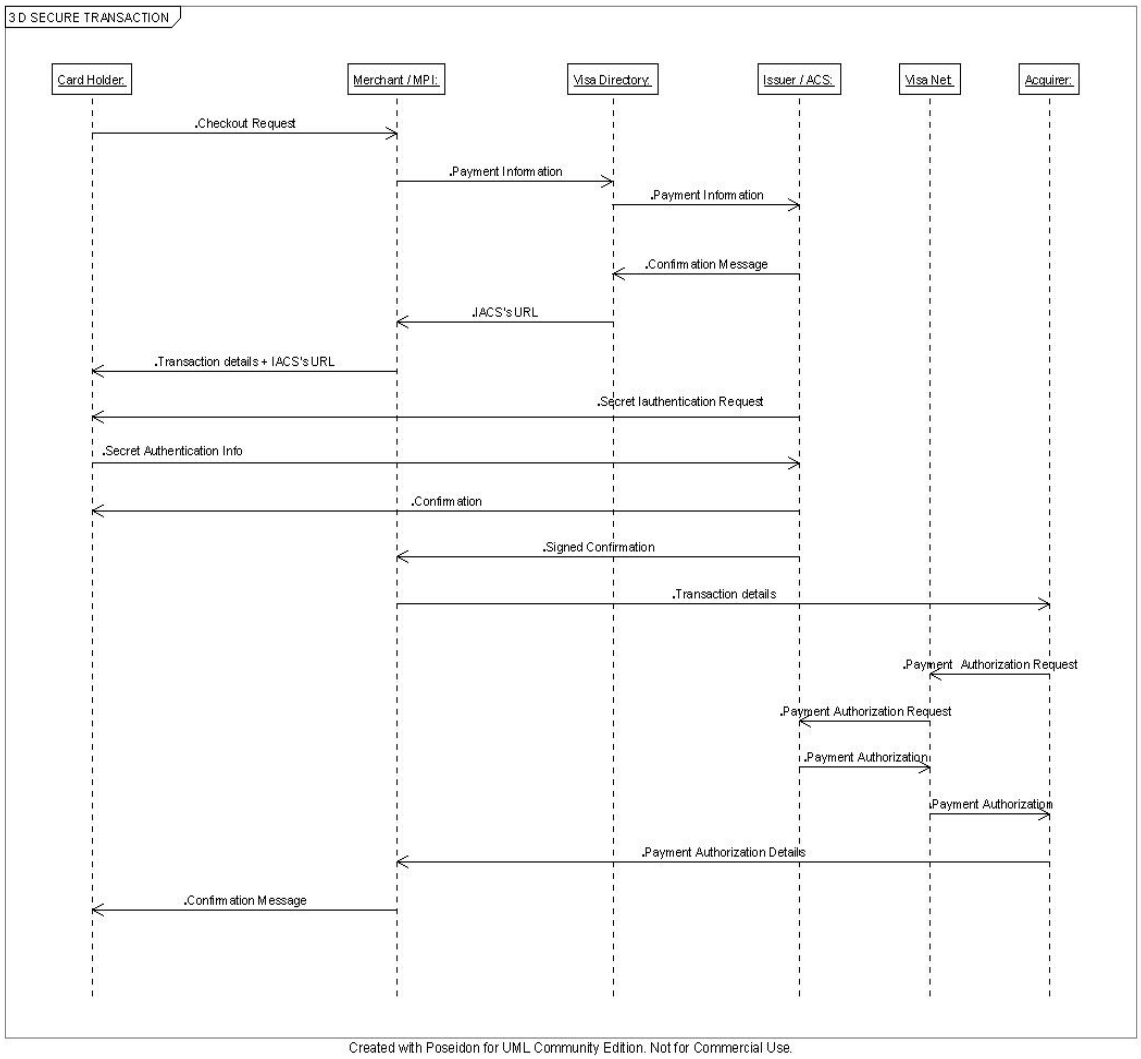

Figure 6.1: The 3-D SECURE transaction procedure

- The cardholder visits a merchant's website and, after having decided what to buy, submits a checkout request to finalize the transaction. The merchant will then be in possession of all the necessary data.
- The merchant plug-in MPI will send the PAN to Visa directory server.
- Having received this information from the MPI, the Visa directory server will then request from the Access Control Server ACS which authentication method is relative to the PAN received from the MPI.
- After checking the ACS will provide the authentication method to be used to the Visa directory server.
- The Visa directory server will then forward the ACS message to the MPI.
- The MPI will submit a Payer authentication request PAReq to the ACS via the cardholder's device
- The ACS gets the PAReq.
- The ACS will interact with the cardholder for authentication and the result of the authentication will be included in the payer authentication response PARes that the ACS will then sign.
- The ACS submits the PARes to the MPI again via the shopper's device and adds the data to the Authentication History Server.
- The MPI gets the PARes from the ACS.
- The MPI will then verify the signature of the PARes.
- The merchant will proceed with authorization exchange with its acquirer.
- The acquirer returns the authorization response to the merchant.
- The merchant issues a confirmation to the cardholder and the transaction is completed.

7.5. Analysis

In this phase, we will verify how does 3-D secure fulfill the security requirements invoked in section 7.3

7.5.1. Confidentiality

This requirement is met by the use of SSL/TLS. In fact, 3-D SECURE makes sure that sensitive date transmitted between the merchant and the cardholder is encrypted using strong cryptographic facilities [17]. However just as already pointed out in [12], once this data reaches the merchant website; the merchant can have access to this data just as in any normal online transaction involving use of SSL/TLS.

7.5.2. Integrity

This requirement is met by use of digital signature [17]. In fact, prior to submitting PARes to the MPI the issuer will first sign this PARes insuring that it did came from the Issuer [17].

7.5.3. Authentication

Here we have authentication in two ways: from merchant to cardholder and from cardholder to merchant.
By the use of SSL the cardholder can be sure he/she is talking to the merchant he/she intends to.
The merchant on the other hand can be sure he/she is talking to the cardholder via the response gathered from ACS: in fact the ACS will certify that the cardholder in indeed genuine and entitled to take part to the transaction.

7.5.4. Non-repudiation

By making use of identity verification methods described in [17] we can guarantee non repudiation of transaction for both the consumer and the merchant: the consumer will not then deny having made orders through the merchant website and the merchant can not deny the customer never visited his/her website.

CHAPTER 8 CONCLUSION

The aim of this chapter is to conclude and summarize the primary contributions of this thesis (Section 8.1). In Section 8.2, suggestions for future research are given.

8.1. Main Contributions

This thesis dealt with the security related problems of electronic payment and how the three protocols are helping to deal with those problems.
In chapter 2, we discussed the main security threats and the security requirements that must be fulfilled using the appropriate security mechanisms.
In chapter 3 we introduced electronic payment systems, and we gave a generic model for such systems. Characteristics that distinguish various types of electronic payment systems were also identified. These characteristics can be used to analyze an electronic payment system. Moreover, these properties help to determine the interpretation of the overall model.
In chapter 4 we introduced the reader to this new security protocols analyst program named scyther in order to use it
In chapter 5 we have proposed an overview of the EMV specifications to enable cards to conduct transactions at POS terminals. We described the payment method in detail, explained how it meets the identified security requirements and included a Scyther analysis of some parts of the protocol.
In chapter 6 we have proposed an overview of the CPA specifications to enable cards to conduct transactions at POS terminals. We described the payment method in detail, and explained how it meets the identified security requirements.
In chapter 7 we have proposed an overview of the 3D SECURE to conduct online credit card processing. We described the payment method in detail, and explained how it meets the identified security requirements.
Scyther has been successful in giving attacks on some security mechanisms on EMV that have not been discussed as far as I know before such as the attack on DDA.

8.2. Future Work

Despite the improvements that we can get from these three protocols in terms of security, there might be still some major new problems that will rise with the continuous and tremendous growth that electronic payment is facing nowadays. Two of the main issues that can arise are:

- Personal mobile devices such as PDA or mobile phone are becoming more and more affordable to everyone: it will be interesting if we can incorporate these devices to be used in any of these protocols. For example mobile phones are used to conduct micro payments such music download or paying parking via SMS.
- Biometric authentication as well as elliptic cryptography is being investigated by researchers due to their acutance and low computing power (ideal for smart cards) the question that might rise is if it is possible to incorporate

those new technologies and if so, will it affect positively the security of those protocols?

- EMV has proven to have increased security on smart cards payment, the question that might rise is if we could combine EMV with 3-D SECURE in order to enhance more security with this?

Due to current limitation of the scyther tool, we were not able to model the whole protocol in order to get a clearer picture of each of the protocols' behavior: this is specially pointing out at application cryptograms (EMV,CPA) since use of MAC is not available on the scyther version used at the time of writing so if scyther evolves to a much upgrading version, it will be interesting to investigate the results of modeling the missing parts.

As it has been pointed out in chapter 7, I was not able to use scyther for automatic analysis due to the fact that the protocol specifications unlike those of EMV and CPA are not publicly available and hence they are licensed so it would be interesting to model the 3-D SECURE protocol or at least parts of it in Scyther and see whether Scyther can discover any security flaws that were not detected or prevented previously.

BIBLIOGRAPHY

[1] M. Al-Meaither, *Secure Electronic Payments for Islamic Finance*. RHUL, 2004.

[2] Asokan, Janson, Steiner and Waidner, *State of the Art in Electronic Payment Systems*, 1999.

[3] C.J.F. Cremers, *Scyther: Automatic verification of security protocols*. Available at http://www.win.tue.nl/~ccremers/scyther/

[4] C.J.F. Cremers, S. Mauw, *Operational semantics of security protocols*. Available at http://www.win.tue.nl/~ecss/downloads/.

[5] G.N. Drew, *Using SET for secure electronic commerce*, 1997.

[6] EMVCo, *CPA Draft version 1.0*, July 2005. Available at http://www.emvco.com

[7] EMVCo. *EMV2004: Integrated Circuit Card Specification for Payment Systems: Book 1 | Application Independent IC Card to Terminal Interface Requirements, 2004*. Available at http://www.emvco.com

[8] EMVCo. *EMV2004: Integrated Circuit Card Specification for Payment Systems: Book 2 | Security and Key Management, 2004*. Available at http://www.emvco.com

[9] EMVCo. *EMV2004: Integrated Circuit Card Specification for Payment Systems: Book 3 | Application Specification, 2004*. Available at http://www.emvco.com

[10] EMVCo. *EMV2004: Integrated Circuit Card Specification for Payment Systems: Book 4 | Cardholder, Attendant, and Acquirer Interface Requirements, 2004*. Available at http://www.emvco.com

[11] J. Hord. *How electronic payment works*. Available at http://www.howstuffworks.com

[12] P. Jarupunphol and C. J. Mitchell, *Measuring 3-D Secure and 3D SET against e-commerce end-user requirements*. June 2003.

[13] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*, 1997

[14] E. Rescorla, *SSL and TLS: designing and building secure systems*, 2001

[15] VISA. *3-D Secure | Introduction Version 1.0.2*, September 2002. Available at http://www.visa.com/.

[16] VISA. *3-D Secure | Protocol Specification*.

[17] VISA. *3-D Secure / System Overview Version 1.0.2*, May 2003. Available at http://www.visa.com/

[18] WIKIPEDIA. http://en.wikipedia.org

