私立東海大學資訊工程與科學研究所

碩士論文

指 導 教 授 ： 林　祝　興　博士
(Dr. Chu-Hsing Lin)

植基於智慧卡的無爭議簽章之研究
A Study of Undeniable Signature Schemes Based on Smart Cards

研 究 生 ： 李　　正　　隆
(Cheng-Lung Lee)

中 華 民 國 九 十 一 年 十 月 二 十 五 日

謝誌

這本論文的完成代表了我通過了碩士班所必經的歷練，在這兩年之間，除了學習到專業知識外，也學到了做研究的方法與態度；而首先要感謝的人就是我的指導教授 林祝興 博士，老師在兩年間給我相當多的意見和鼓勵，讓我獲益良多；此外，我也相當感謝琇緯、鎮宇、國鴻、保利、子杰、峰菖與其他學弟們的協助，讓我的論文得以順利完成。

最後，感謝所有陪伴在我身邊的家人與朋友，願所有人都能健康快樂。

摘要

在本論文中，我們提出了兩種適用於智慧卡的無爭議簽章方法，在所提出的第一個方法之中，我們考慮到智慧卡的運算能力並不適用於處理大量的數學運算，因此運用了伺服器輔助運算的技術配合以 RSA 為基礎的無爭議簽章，設計出一個新的無爭議簽章方法；此外，我們將該方法運用在軟體反盜版的機制。

而我們所提出的第二個無爭議簽章方法則是植基於橢圓曲線離散對數，在該方法中，驗證者利用智慧卡與簽章者互相驗證對方的身分，而只有持合法智慧卡的驗證者才能夠驗證簽章的正確性。

關鍵字：無爭議簽章、智慧卡、橢圓曲線密碼系統、伺服器輔助運算、軟體反盜版、以 RSA 為基礎之無爭議簽章、指定證實者簽章。

Contents

Abstract 1

1. Introduction..2

2. Background 4

  2.1. One-Way Hash Function..4

  2.2. Symmetric Key Crytptography 4

  2.3. Digital Signature 5

  2.4. Undeniable Signature 6

  2.5. Designated Confirmer Signature...8

    2.5.1. Signature Generation Phase 9

    2.5.2. Signature Verification Phase...10

  2.6. Elliptic Curve Cryptography..12

  2.7. Smart Card 15

3. RSA-Based Undeniable Signature and Server-Aided Computations 18

  3.1. RSA-Based Undeniable Signature 18

    3.1.1. Key and Signature Generation Phase..............................18

    3.1.2. Signature Confirmation Protocol 19

    3.1.3. Signature Deniable Protocol 20

  3.2. Server-Aided Computations 21

    3.2.1. Notation..22

    3.2.2. The Computation Protocol...23

4. Proposed Schemes..26

  4.1. RSA-Based Undeniable Signature Using Smart Card 26

    4.1.1. Notations 26

    4.1.2. Signature and Smart Card Generation Phase 26

    4.1.3. Confirmation Protocol 27

    4.1.4. Deniable Protocol..30

    4.1.5. Signatures Request Protocol 32

    4.1.6. Applications on software protection...............................33

4.2. Elliptic Curve Undeniable Signatures Using Smart Cards 35

4.2.1. Notations 35

4.2.2. Proposed Scheme 36

5. Complexity and Security Analysis...43

5.1. Complexity Analysis of the RSA-Based Undeniable Signature Scheme Using Smart Card 43

5.2. Security Analysis of the RSA-Based Undeniable Signature Scheme Using Smart Card 43

5.3. Security Analysis of Elliptic Curve Undeniable Signatures Scheme Using Smart Cards 44

6. Conclusion...47

Reference 48

Appendix A: Elliptic-Curve Undeniable Signature..54

A.1. Elliptic-curve Undeniable Signatures Based on Chaum' s...............54

A.2. Elliptic-curve Group Undeniable Signatures 55

List of Figures

Figure 1. The verification protocol of designated confirmation signature 11
Figure 2. The addition of two distinct points R=P+Q 14
Figure 3. Doubling of a point R=P+P 14
Figure 4. Smart card physical appearance 15
Figure 5. RSA-based undeniable signature scheme – confirmation protocol 19
Figure 6. RSA-based undeniable signature scheme – deniable protocol ...20
Figure 7. The server aided computation protocol for RSA encryption 23
Figure 8. Confirmation Protocol – Step 1 27
Figure 9. Confirmation Protocol – Step 2 28
Figure 10. Confirmation Protocol – Step 3,4 29
Figure 11. Deniable Protocol – Step 2 30
Figure 12. Deniable Protocol – Step 3,4 31
Figure 13. Signatures Request Protocol 32
Figure 14. The system model of software protection system 34
Figure 15. Software Protection Model – Permission Phase 34
Figure 16. Verifier Authentication Scheme 39
Figure 17. Elliptic curve designated confirmer signature scheme --- verification phase 42

Abstract

In this thesis, we proposed two undeniable signature schemes suitable for implementing on the smart card. In the first scheme, because the exponential computation on undeniable signature scheme needs the heavy computation power, the smart card is not proper to do it. To reduce the computing load of the smart card, we applied the server-aided computation technology and the RSA-based undeniable signature scheme to design a new undeniable signature protocol. The second scheme is based on elliptic curve discrete logarithm problem. In this scheme, the signer and verifier could authenticate to each other via the smart card, and only the verifier with legitimate smart card could verify the signature. Besides, by the first proposed protocol, we applied it for the software anti-piracy.

Keyword: Undeniable Signature, Smart Card, Elliptic Curve Cryptosystem, Server-Aided Computation, Software Anti-Piracy, RSA-Based Undeniable Signature, Designated Confirmer Signature.

1. Introduction

Due to the World Wide Web’s popularity, it is important to provide some security measures to protect information transferring over the internet. The digital signature schemes have the features: integrity, authentication and non-repudiation. Although the digital signature scheme is convenient to use, it is improper for some applications. For example, a software vendor hopes that there is a digital signature on its software and only legitimate users can verify the validity of the signature. Undeniable signature is well suited to such applications because it is unable to be verified without the signer’s assistance. Undeniable signature scheme was first introduced by D. Chaum and H. van Antwerpen in 1989[8], and Harn and Yang proposed a group oriented undeniable signature scheme using Chaum’s scheme in 1992[16]. Afterward, some variations of undeniable signature were proposed such as convertible undeniable signature[13][14][19][26][20] and designated confirmer signature[11][21][31]. The proposed variational schemes may more flexibility than original scheme, but there are still some important problems which need to be overcome. One of it is the authentication to the verifier and another is the protection of the signed message. In this thesis, we propose two schemes to settle the above problems respectively. Both schemes take advantage of the smart card as the auxiliary device to achieve the objectives and are based on two different cryptosystems: RSA and Elliptic Curve Cryptosystem respectively. In the RSA based scheme, because the computing power of smart card is low, we apply the server-aided computation scheme to design the protocol. The scheme will let the smart card to transform the heavy computing data with some specific functions, and then send the transformed data to the highly computing power devices to compute and get the result. The others can't get the confidential data and the legal user can't easily copy the smart due for the security of smart card. This scheme can prevent the attacks proposed by Desmedt and Yung [33] and Jakobsson [17]. In the Elliptic Curve undeniable signature scheme, the signer and verifier could authenticate to each other via the smart card, and only the verifier with legitimate smart card could verify the signature.

Organization of the Thesis: The remainder part of this thesis is organized as follows: In Chapter 2, we introduce the background of the related technologies used in this thesis. Then we describe Server-Aided in Chapter 3. The proposed schemes are describe in Chapter 4 and their complexity and security analysis are in Chapter 5. The conclusion is given in Chapter 6.

2. Background

2.1. One-Way Hash Function

One-way hash function is an important part in modern cryptosystems. It may have different names in different situation to use it. Its name could be compression function, contraction function, message digest, fingerprint, cryptographic checksum, and message integrity check. The most important characteristic of one-way hash function is that it is easy to compute a hash from a message, but it is hard to generate the original message from the hash. We say a one-way hash function is collision-free if it is hard to find two messages with the same hash value.

In this thesis, we use the one-way hash function to generate the message digest and use it to make some secure value with some secret parameter.

2.2. Symmetric Key Crytptography

Symmetric key cryptography is a traditional form of cryptography, in which people can use a key to encrypt a message and decrypt the message with the same key. The advantage of symmetric key cryptography is that it is much faster than asymmetric key cryptography. However, there is an important issue in symmetric key cryptography: How two ends share an agreed secret key without anyone else getting it? Some techniques could solve this problem with eliminating the misgiving of eavesdropping. There two basic types of symmetric key cryptography: block cipher (such as DES, 3DES, and AES) and stream cipher (such as A5 used in GSM).

2.3. Digital Signature

We have used handwritten signatures as the proof of authorship for a very long time. Until now, we still use it frequently in our daily life. B. Schneier describes the general properties of handwritten signature as following [3]:

1. The signature is authentic.
2. The signature is unforgeable.
3. The signature is not reusable.
4. The signed document is unalterable.
5. The signature cannot be repudiated.

However, in the digital world, the handwritten signatures also become infeasible because it will be copied easily. In contrast, the digital signature is difficult to forge without the secret information used in making the digital signature. There are many ways to make a digital signature such as one-way hash function and public-key cryptography. Besides, all kinds of digital signature have the features: confidentiality, integrity, authentication and non-repudiation. These two properties make digital signature very useful in many applications but not all. Hence, many variant of digital signature are created such undeniable signature, proxy signature, group signature, multiple signature, and blind signature. The undeniable signature is the key point in this thesis and it will be described in detail in section 2.4.

2.4. Undeniable Signature

Digital signature has been widely applied in the world today. However, the conventional digital signature isn’t suitable for some specially requirement. Some people don’t want their signatures to be verified by anyone using his public key. For example, a software company will embed a digital signature into its system to prevent being embedding some virus or Trojan horse codes in its system. They hope that the legal user can verify the digital signature of its system, not the general user can do. The undeniable signature scheme can solve the problem mentioned above. The verifier can’t verify an undeniable signature without the signer’s assistance. The concept of the undeniable signature was first introduced by D. Chaum and H. van Antwerpen in 1989[8]. D. Chaum also proposed another undeniable signature scheme with the property of zero-knowledge in 1990[9].

Chaum’s scheme has a problem that the verifier must execute confirmation and disavowal protocols both in some situation. Therefore, in [1] the author proposed another undeniable signature scheme that put the confirmation and disavowal of a signature in the same protocol. Another serious problem of Chaum’ s scheme is blackmailing [33][10][17]. The major cause of blackmailing is that the signer never knows what signature is verified. However, there are still ways to overcome this drawback such as the designated verifier proof [18].

In order to make the undeniable signature more flexible to be suitable for real case in the Internet, several variant undeniable signatures were proposed in the later years. Such as the convertible undeniable signature [13][14][19][26][20], the designated confirmer signature[11][21][31], and the group-oriented undeniable signature [16][24][25][6]. A convertible signature could be converted the undeniable signature to the conventional signature and the verification of a designated confirmer signature could be cooperated by a verifier and a confirmer delegated by the original signer when the signer is absent or refuses to cooperate with the verifier. Moreover, a designated confirmer signature also has the property the same as the convertible undeniable signature that it could be converted into conventional signature.

After the concept of undeniable signature has been proposed, the security requirements of undeniable signature schemes began to be considered. In [21], the authors point out that a confirmer signature schemes must meet the four security requirements:

Unforgeability of signatures: It' s very obviously that no signature can be forged.

Invisibility of signatures: This means that the no verifier can verify the signature on his own.

Consistency of verification: This means that the signer or the confirmer cannot prove that a valid signature is invalid or an invalid signature is invalid.

Non-transferability: This means that verifier participating in the verification protocol cannot convince the others the validity of a signature.

In other types of undeniable signature scheme have the same requirements.

Undeniable signature and its variants are very suitable for some specific applications such as the undeniable certificates [27], the fair exchange protocol [15], and the fair payment protocol [4].

2.5. Designated Confirmer Signature

Designated confirmation signature is a variant of undeniable signature proposed by D. Chaum[11]. This type of scheme provides additional flexibility. A signer could delegate another party named a designated confirmer to prove the validity of a signature to the verifier. Moreover, Okamoto proved that a designated confirmer signature scheme and a public-key encryption scheme are equivalent [31].

M. Michels and M. Stadler proposed a generic construction for confirmer signature schemes[21]. In the paper, they introduced a new tool named confirmer commitments. The signer first generates a confirmer commitment $d = Com(m, y_c)$ using the confirmer's public key y_c. Instead of signing the message directly, the signer signs d using the conventional signature scheme such as RSA and DSA. Therefore, the verifier could check that the signature is the correct signature on d without others' assistance. And then, the verifier asks the confirmer to prove that d is a commitment for m.

Here, we show their confirmer signature scheme using a confirmer commitment. Their scheme applies Schnorr's identification scheme [7] and a convertible undeniable signature scheme [20].

2.5.1. Signature Generation Phase

Step 1: Let p be a large prime and g be a primitive element selected in $GF(p)$. The signer selects an arbitrary number x_s where $x_s \in Z_p^*$ as his secret key, and computes $y_s = g^{x_s} \pmod p$ as his public key. The confirmer selects another arbitrary number x_c where $x_c \in Z_p^*$ as his private key, and computes $y_c = g^{x_c} \pmod p$ as his public key.

Step 2: The signer computes $d_1 = g^{t+H(M\|r)} (\text{mod } p)$, $d_2 = y_c^t (\text{mod } p)$, and let $d=$ $Com(m \| r, y_c)$ $= (d_1, d_2)$.

Step 3: The signer selects a random number c Î Z_p^* and compute $r= g^c (\text{mod } p)$, $s=c\text{-}H(d)$ ‧ x_s $(\text{mod } p)$, where $H(\cdot)$ is an one way hash function.

Step 4: The signer publishes his confirmer signature (s, d).

2.5.2. Signature Verification Phase

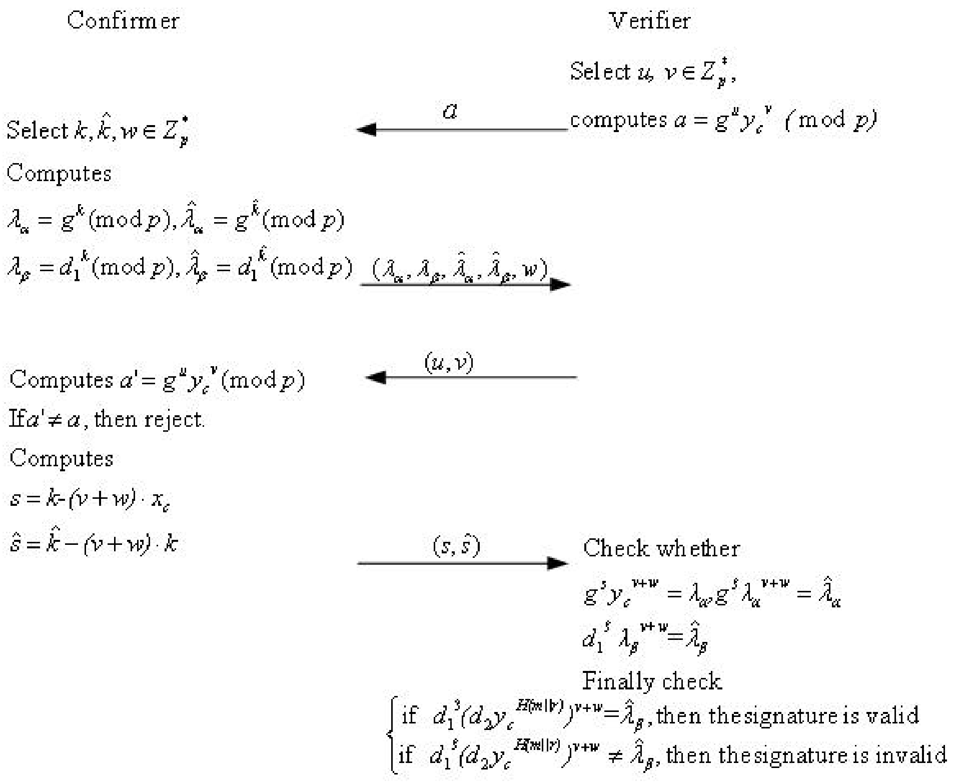
Step 1: The verifier chooses two random number $u, v \in Z_p^*$, computes $a = g^u y_c^v (\text{mod } p)$ and send a to the confirmer.

Step 2: The confirmer chooses three random numbers $k, \hat{k}, w \in Z_p^*$ and computes $l_a = g^k (\text{mod } p)$, $\hat{l}_a = g^{\hat{k}} (\text{mod } p)$, $l_b = d_1^{\,k} (\text{mod } p)$, $\hat{l}_b = d_1^{\,\hat{k}} (\text{mod } p)$ and sends *(* l_a , l_b , $\hat{l}_a$, $\hat{l}_b$ *,w)* to the verifier.

Step 3: After receiving (l_a , l_b , $\hat{l}_a$, $\hat{l}_b$, w), the verifier sends (u,v) to Confirmer.

Step 4: After receiving (u,v), the confirmer computes $a' = g^u y_c^v (\text{mod } p)$. If $a' \neq a$, then the confirmer rejects the following communication, else he computes $s = k - (v + w)x_c$, $\hat{s} = \hat{k} - (v + w)k$, and sends $(s, \hat{s})$ to the verifier.

Step 5: The verifier first checks whether $g^s y_c^{v+w} = \boldsymbol{l}_a$, $g^{\hat{s}} \boldsymbol{l}_a^{v+w} = \hat{\boldsymbol{l}}_a$, $d_1^{\hat{s}} \boldsymbol{l}_b^{v+w} = \hat{\boldsymbol{l}}_b$, and finally checks that if $d_1^{\,s}(d_2 y_c^{\,H(m\|r)})^{v+w} = \boldsymbol{l}_b$, then the verifier is convinced that the signature is valid, else he is convinced that the signature is invalid.



Confirmer
Verifier
Select $u, v \in Z_p^*$,
computes $a = g^u y_c^{\,v}\ (\bmod p)$
a
Select $k, \hat{k}, w \in Z_p^*$
Computes
$\lambda_\alpha = g^k (\bmod p), \hat{\lambda}_\alpha = g^{\hat{k}} (\bmod p)$
$\lambda_\beta = d_1^{\,k} (\bmod p), \hat{\lambda}_\beta = d_1^{\,\hat{k}} (\bmod p)$
$(\lambda_\alpha, \lambda_\beta, \hat{\lambda}_\alpha, \hat{\lambda}_\beta, w)$
(u, v)
Computes $a' = g^u y_c^{\,v} (\bmod p)$
If $a' \neq a$, then reject.
Computes
$s = k - (v + w) \cdot x_c$
$\hat{s} = \hat{k} - (v + w) \cdot k$
$(s, \hat{s})$
Check whether
$g^s y_c^{\,v+w} = \lambda_\alpha, g^{\hat{s}} \lambda_\alpha^{\,v+w} = \hat{\lambda}_\alpha$
$d_1^{\,\hat{s}} \lambda_\beta^{\,v+w} = \hat{\lambda}_\beta$
Finally check
if $d_1^{\,s}(d_2 y_c^{\,H(m\|r)})^{v+w} = \hat{\lambda}_\beta$, then the signature is valid
if $d_1^{\,s}(d_2 y_c^{\,H(m\|r)})^{v+w} \neq \hat{\lambda}_\beta$, then the signature is invalid

Figure 1. The verification protocol of designated confirmation signature

2.6. Elliptic Curve Cryptography

Elliptic Curve Cryptosystem (ECC) was proposed by Neal Koblitz[22] and Victor Miller[32] in 1985. The security of elliptic curve cryptosystem is based on the difficulty of computing an elliptic curve discrete logarithm problem (ECDLP)[2]. Due to numerous researches have been done on its security and efficient implementation, ECC has accepted by standard organizations. Such as Elliptic Curve Digital Signature Algorithm (ECDSA)[12] proposed in 1992 by Scott Vanstone[29] was accepted in 1998 as an ISO standard (ISO 14888-3), accepted in 2000 as an IEEE standard (IEEE P1363) and a FIPS standard (FIPS 186-2).

In this section we will give a quick introduction to elliptic curve. Let E be a elliptic curve over Z_p, and B be a point on E of order n, i.e. $B \in E(Z_p)$. In general applications, p is typically a power of 2 or an odd prime number. Then we can choose a number l and let $Q = lB$, where $0 \le 1 \le n-1$ and Q is also a point on E. If n and p are large enough, it is hard to find l with knowing E, Q, and B. This is called the elliptic curve discrete logarithm problem (ECDLP) [23][32].

An elliptic curve E over Z_p is defined as following equation

$y^2 = x^3 + ax + b$

where $a, b \in Z_p$ and $4a^3 + 27b^2 \neq 0(\text{mod } P)$, and all the points (x, y), $x \in Z_p$, $y \in Z_p$, form the set of $E(Z_p)$ containing a point O called the point at infinity. When a point B on the elliptic curve E multiplied by a number l, it is equivalent to adding B to itself l times, and will yield another point on the curve. A rule, called chord-and-tangent rule, is utilized to add two points on an elliptic curve to get another point. We now describe the addition formula on the elliptic curve.

If $B = (x, y) \in E(Z_p)$ then $B + O = O + B = B$ and $B + (-B) = O$, where $-B = (x, -y)$ called the negative of B.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E, i.e. $P, Q \in E(Z_p)$. The formula for adding P and Q are described as follows:

$R = P + Q = (x_3, y_3)$, where $P \neq -Q$, and

$$\begin{cases} x^3 = d^2 - x_1 - x_2 \\ y^3 = d(x_1 - x_3) - y_1 \end{cases} \quad \text{and} \quad d = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & if \quad P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & if \quad P = Q \end{cases}$$

Figure 2 and 3 are the geometric description of the additio n rule. In Figure 2, P ¹ Q, drawing a line through P and Q, it will intersect the elliptic curve in third point. Then R is the reflection of the third point in the x-axis.

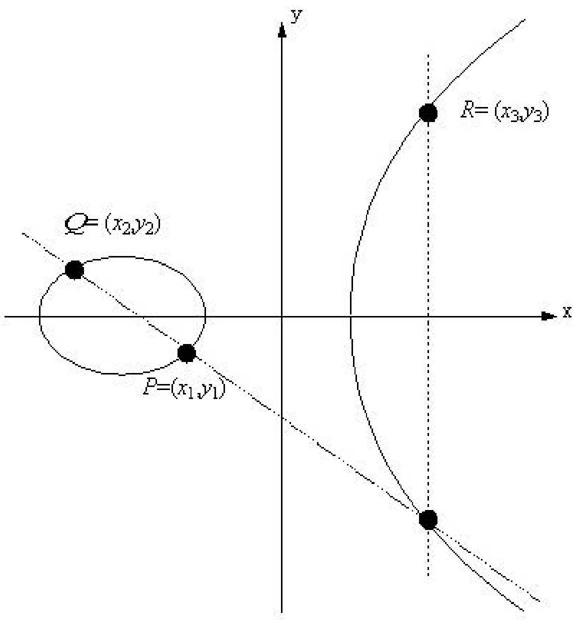


Figure 2. The addition of two distinct points R=P+Q

In Figure 3, $P = Q$, drawing a tangent line to the elliptic curve at P, it will intersect the elliptic curve in the second point. Then R is the reflection of the third point in the x-axis.

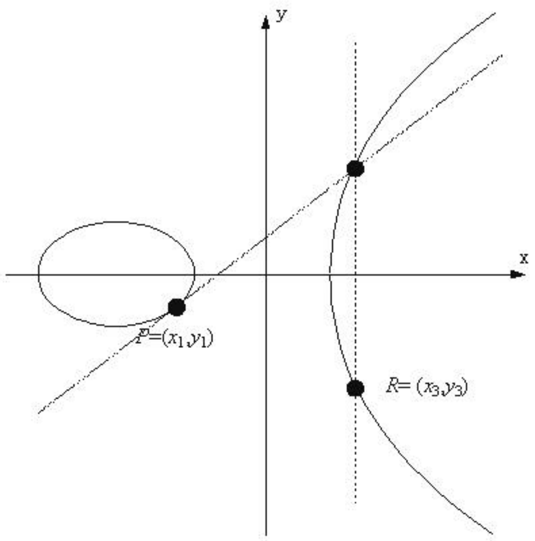


Figure 3. Doubling of a point R=P+P

2.7. Smart Card

The smart card, an intelligent token, is a credit card sized plastic card embedded with an integrated circuit chip. It provides not only memory capacity, but computational capability as well. Nowadays, the size of storage and ability of computation of smart card continue to increase. Besides, the chip on smart card also allows the implementation of cryptographic and authentication scheme. Hence, in this thesis, we propose two cryptographic protocols based smart card. The Figure 4 shows the physical appearance of smart card.

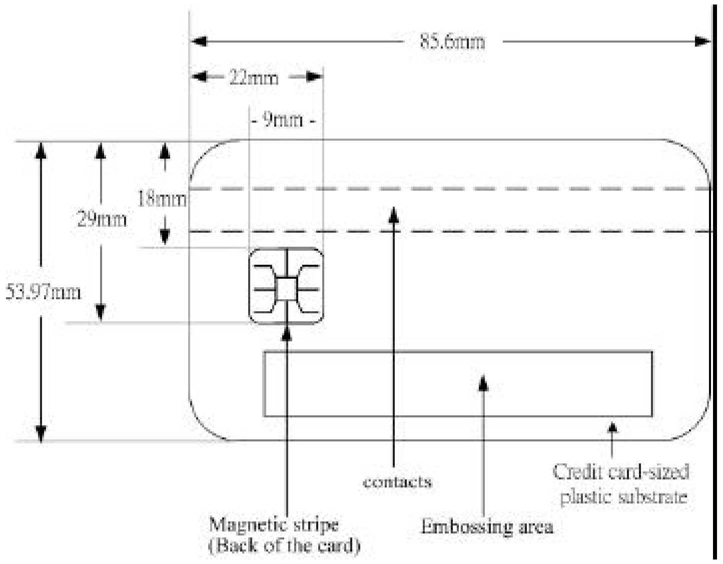

85.6mm
22mm
- 9mm -
18mm
29mm
53.97mm
contacts
Credit card-sized plastic substrate
Magnetic stripe (Back of the card)
Embossing area

Figure 4. Smart card physical appearance

In general, smart card should have the ability of tamper resistance to prevent some malicious explore to the data in the smart card. There are several types of smart card:

- memory cards
- processor cards
- electronic purse cards
- security cards
- Java Card

With the development of new technology, there are many smart card related standards. We describe these standards below.

Horizontal standards

- ISO 7816 – This describes the lowest-level interface to a smart card. It is at this level that data bytes are transferred between card reader and card and it is the most important standard defining the characteristic of chop cards that have electrical contacts. ISO 7816[34] covers various aspects of smart cards:
    - Part 1 – physical characteristics
    - Part 2 – dimensions and location of the contacts
    - Part 3 – electronic signals and transmission protocols
    - Part 4 – interindustry commands for interchange
    - Part 5 – application identifiers
    - Part 6 – interindustry data elements
    - Part 7 – interindustry commands for SCQL
- PC/SC – It is the standard for communicating with smart cards connected to personal computer system. [35]

- PKCS #11 – This is an interface between application and all kinds of portable cryptographic devices. [36]
- OCF – OCF is an all-Java interface for communicating with smart cards from a Java environment. [37]
- Java Card – It describes the Java Card and what it supports. [38]
- Multos – It is a multi-application operation system for smart cards. [39]

Vertical standards

- Mondex – A kind of digital cash that uses smart cards only. The Mondex approach does not allow cash to exist outside of the card. [40]
- CEPS – The main purpose of the common electronic purse specifications (CEPS) is to define requirements for all components needed by an organization to implement a globally interoperable electronic purse program and to maintain full accountability and auditability. [41]
- MPCOS-EMV – This is a general-purpose card that lets you implement your own type of currency or token. [42]

3. RSA-Based Undeniable Signature and Server-Aided Computations

3.1. RSA-Based Undeniable Signature

The RSA-based undeniable signature scheme has two protocols, confirmation protocol and deniable protocol, be used to verify and to deny the signature. The confirmation protocol let the verifier can verify the signature. The deniable protocol let the signer can deny the signature.

3.1.1. Key and Signature Generation Phase

Key Generation Phase:

Step 1: Randomly choose two large prime numbers, p and q at least 512 bits long. Compute the product:

$$n = pq .$$

Step 2: Randomly choose a prime number, e , such that e and $(p-1)(q-1)$ are relatively prime. Compute a integer d , such that

$$ed = 1 \bmod (p-1)(q-1) .$$

Step 3: Choose a pair (W , S_W), such that

$$W \in Z_n^*, W \neq 1, S_W = W^d \bmod n .$$

Let the parameters, n , W and S_W , be the public key and the parameters, d and e , be the private key.

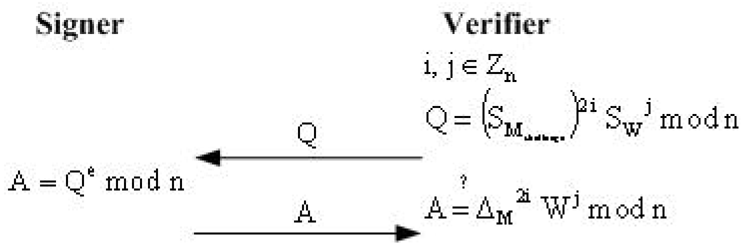
Signature Generation Phase:

Let M be a document and $\mathrm{H}(\bullet)$ be a one-way hash function.

Step 1: The signer uses a hash function to compute the message digest of a document, $\ddot{A}_M = H(M)$.

Step 2: Let the signature on the document M is S_M , $S_M = \ddot{A}_M{}^d \bmod n$.

3.1.2. Signature Confirmation Protocol



Signer
Verifier
$i, j \in Z_n$
$Q = (S_{M_{challenge}})^{2i} S_W{}^j \bmod n$
Q
$A = Q^e \bmod n$
A
$A \overset{?}{=} \Delta_M{}^{2i} W^j \bmod n$

Figure 5. RSA-based undeniable signature scheme – confirmation protocol

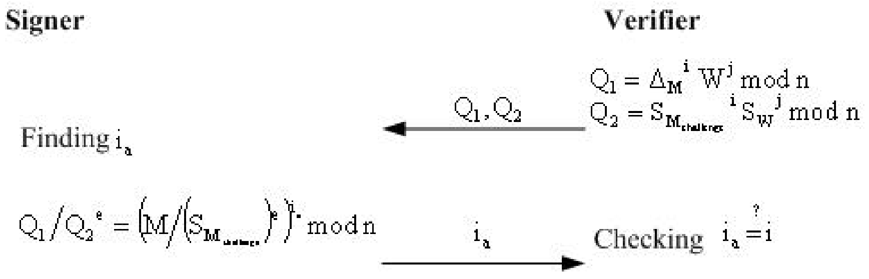
The Figure 5 shows how the signer confirms the validity of the signature for the verifier. If the equality holds, then the verifier accepts $\mathrm{S}_{\mathrm{M}_{\mathrm{challenge}}}$ be the signature on the document M ; otherwise undetermined. (According to the Lemma 1, if the equality holds, then it means that $\mathrm{S}_{\mathrm{M}_{\mathrm{challenge}}}$ is equal to S_M .)

Lemma 1. Let $Q = S_M{}^{2i} S_W{}^{j} \bmod n$, $S_M = \ddot{A}_M{}^{d} \bmod n$ and $S_W = W^{d} \bmod n$ such that $Q^{e} \bmod n = \ddot{A}_M{}^{2i} W^{j} \bmod n$.

Proof.

$$
\begin{aligned}
& Q^{e} \bmod n \\
&= \left(S_M{}^{2i} S_W{}^{j}\right)^{e} \bmod n \\
&= \left(\left(\ddot{A}_M{}^{d}\right)^{2i} \left(W^{d}\right)^{j}\right)^{e} \bmod n \\
&= \left(\left(\ddot{A}_M{}^{2i}\right)^{d} \left(W^{j}\right)^{d}\right)^{e} \bmod n \\
&= \ddot{A}_M{}^{2i} W^{j} \bmod n
\end{aligned}
$$

3.1.3. Signature Deniable Protocol



Signer
Verifier
$Q_1 = \Delta_M{}^{i} W^{j} \bmod n$
$Q_2 = S_{M_{challenge}}{}^{i} S_W{}^{j} \bmod n$
Q_1, Q_2
Finding i_a
$Q_1/Q_2{}^{e} = \left(M/\left(S_{M_{challenge}}\right)^{e}\right)^{i} \bmod n$
i_a
Checking $i_a \overset{?}{=} i$

Figure 6. RSA-based undeniable signature scheme – deniable protocol

The Figure 6 shows how the signer denies the validity of the signature for the verifier. If the equality holds, then the verifier reject the $S_{M_{challenge}}$ be the signature on M; otherwise undetermined. According to the Lemma 2, if the $S_{M_{challenge}}$ is not the signature on M, then Q_1 is not equal to Q_2. The signer can use the trial-and-error to find i_a such that i_a be equal to i.

Lemma 2. Let $Q_1 = Ä_M{}^i W^j \bmod n$, $Q_2 = S_M{}^i S_W{}^j \bmod n$ such that $Q_1 = Q_2{}^e$.

Proof.

$$
\begin{aligned}
&Q_2{}^e \\
&= \left(S_M{}^i S_W{}^j\right)^e \bmod n \\
&= \left(\left(Ä_M{}^d\right)^i \left(W^d\right)^j\right)^e \bmod n \\
&= \left(\left(Ä_M{}^i\right)^d \left(W^j\right)^d\right)^e \bmod n \\
&= \left(Ä_M{}^i\right)\left(W^j\right) \bmod n \\
&= Q_1
\end{aligned}
$$

3.2. Server-Aided Computations

Matsumoto proposed a method to speed up secret computation on insecure auxiliary devices [30]. The performance of Matsumoto's method was discussed by Kawamura [28]. Afterward Lin and Chang proposed another server-aided computation protocol for RSA enciphering algorithm. The protocol will let the devices with less computation power such as smart card to transform the heavy computing data with some specific functions, and then send the transformed data to the highly computing power devices to compute and get the result. The others can't get the confidential data and the legal user can't easily copy the smart card due to the security of smart card. We will describe the Lin and Chang's protocol as follows:

3.2.1. Notation

- p , q : the larger prime numbers: $n = pq$.
- (e , d): (the encryption key, the decryption key) and satisfies $ed = 1 \bmod (p-1)(q-1)$.
- M : the plaintext, $M = C^d \bmod n$.
- C : the ciphertext, $C = M^e \bmod n$.
- A : a vector $A = (a_0, a_1, .., a_t)$ where a_i randomly generated by client and $p \nmid a_i$ and $q \nmid a_i$, and $a_0 = \left(M \prod_{i=1}^{r} a_i \right) \bmod n$: $i = 1, .., t$ and $r \le t$.
- B : a vector $B = (b_0, b_1, .., b_t)$ where $b_i = a_i^2 \bmod n$, $i = 0, .., t$.
- B′ : a vector $B' = (b_0', b_1', .., b_t')$ and $B' = \ddot{O}(B)$.
- ($\ddot{O}$, $\ddot{O}^{-1}$): (a randomly permutation function, the corresponding inverse permutation function).
- Ψ : $X' = \varnothing(X)$, $X' = (x'_0, x'_1, .., x'_t)$ and $X = (x_0, x_1, .., x_t)$ where

$x'_i = (x_i)^{(e-1)/2} \bmod n, i = 0,..,t$.

- $\grave{E}$: $X' = \grave{E}(X, Y)$ where $X = (x_0, x_1, .., x_t)$ and

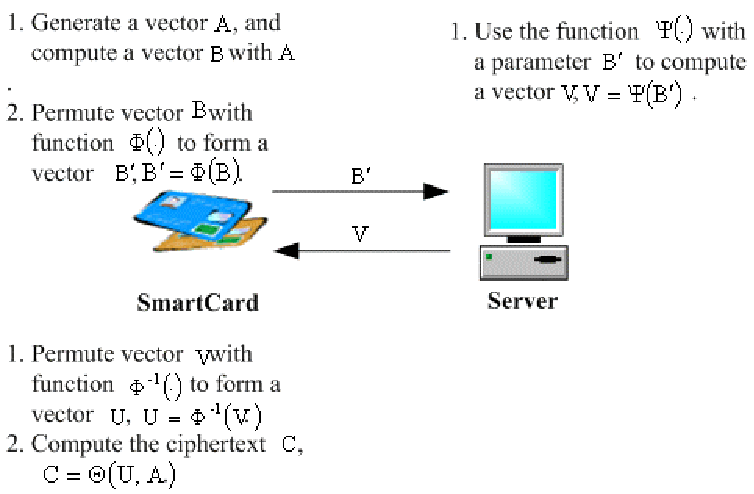
  $Y = (y_0, y_1, .., y_t)$ .where $X' = (x_0 y_0)\left(\left(\prod_{i=1}^{r} x_i\right)\left(\prod_{i=1}^{r} y_i\right)\right)^{-1} \bmod n$.

- V : a vector $V = (v_0, v_1, .., v_t)$ and $V = Ø(B')$.

- U : a vector $U = (u_0, u_1, .., u_t)$ and $U = \ddot{O}^{-1}(V)$.

3.2.2. The Computation Protocol

C.H. Lin and C.C. Chang [5] proposed a protocol of RSA-based server aided computation. The protocol will use the highly computing power device (called server) to assist the computation of $C = M^e \bmod n$ without leaking the plaintext M and the ciphertext C .



1. Generate a vector A, and compute a vector B with A
.
2. Permute vector Bwith function $\Phi(\cdot)$ to form a vector B', $B' = \Phi(B)$.

1. Use the function $\Psi(\cdot)$ with a parameter B' to compute a vector V, $V = \Psi(B')$.

1. Permute vector Vwith function $\Phi^{-1}(\cdot)$ to form a vector U, $U = \Phi^{-1}(V)$
2. Compute the ciphertext C, $C = \Theta(U, A)$

Figure 7. The server aided computation protocol for RSA encryption

In Figure 7, according to the computation rule on the above notation, the client randomly generates a vector A, computes a vector B, uses the function $\ddot{O}(\,)$ to permute vector B to form a vector $B^{'}$ and then send the vector $B^{'}$ to the server. V using the function $Ø(\,)$ with parameter $B^{'}$ and sends the vector V back to the client.

The client permutes the received vector V using the function $\ddot{O}^{-1}(\,)$ to form a vector U and computes the ciphertext C using the function $\grave{E}(\,)$ with two parameters U and A.

Lemma 3. Let vectors U, A and B defined on the notation such that $\grave{E}(U,A) = M^{e} \bmod n$.

Proof.

$$\grave{E}(U,A)$$

$$= (u_0 a_0)\left(\left(\prod_{i=1}^{r} u_i\right)\left(\prod_{i=1}^{r} a_i\right)\right)^{-1} \bmod n$$

$$= \left(b_0^{(e-1)/2} a_0\right)\left(\left(\prod_{i=1}^{r} b_i^{(e-1)/2}\right)\left(\prod_{i=1}^{r} a_i\right)\right)^{-1} \bmod n$$

$$= \left(a_0^{(e-1)} a_0\right)\left(\left(\prod_{i=1}^{r} a_i^{(e-1)}\right)\left(\prod_{i=1}^{r} a_i\right)\right)^{-1} \bmod n$$

$$= \left(a_0^{(e-1)} a_0\right)\left(\left(\prod_{i=1}^{r} a_i^{(e-1)}\right)\left(\prod_{i=1}^{r} a_i\right)\right)^{-1} \bmod n$$

$$= \left(a_0^{e}\right)\left(\prod_{i=1}^{r} a_i^{e}\right)^{-1} \bmod n$$

$$= \left(M\prod_{i=1}^{r} a_i\right)^{e}\left(\prod_{i=1}^{r} a_i^{e}\right)^{-1} \bmod n$$

$$= M^{e}\prod_{i=1}^{r} a_i^{e}\left(\prod_{i=1}^{r} a_i^{e}\right)^{-1} \bmod n$$

$$= M^{e} \bmod n$$

$$= C$$

4. Proposed Schemes

4.1. RSA-Based Undeniable Signature Using Smart Card

In this section, we describe the first proposed scheme. This scheme is based on RSA-based undeniable signature and this scheme has been described in section 3.1.

We use three roles to describe the protocol in detail: smart card, terminal and server. The smart card needs the terminal to do some exponential computation and the server to help it to confirm/deny the signature.

4.1.1. Notations

The notations in this section are the same as section 3.2.1.

4.1.2. Signature and Smart Card Generation Phase

The process of key generation is the same as the section 3.1.1. The public key is the triplet (n, W, S_W), the private key is the pair (e, d).

The signer signs a document M, according to the signature generation process mentioned in section 3.1(We modified the computation of $Ä_M$ as follows). The undeniable signature on the document M is the (n, W, S_W, S_M), where $S_M = \ddot{A}_M{}^d \bmod n$ and $\ddot{A}_M = H(H(M)||ID_{card}||P_{secret})$

Let ID_{card} be the card identification number, the symbol P_{secret} be a secret parameter of the signer and the symbol $||$ be the concatenation operation. The smart card contains the public key, (n , W , S_W); the signature on M , (n , W , S_W , S_M), and the secret parameter, P_{secret} in it.

4.1.3. Confirmation Protocol

Step 1: The smart card sends ID_{card} to the terminal and then the terminal forwards it to the server.

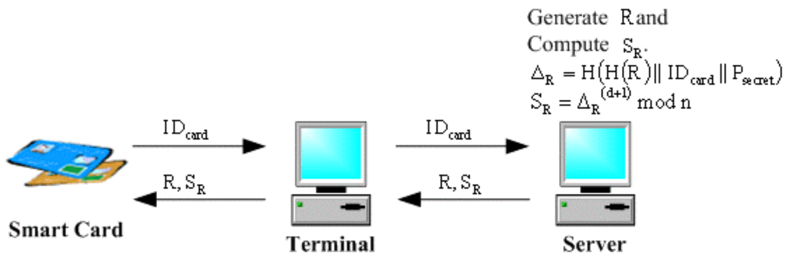


Figure 8. Confirmation Protocol – Step 1

(1.1) If ID_{card} is correct, then the server generates a random number R and computes $\ddot{A}_R$ and S_R, and then sends R and S_R to the terminal.

$$\ddot{A}_R = H(H(R) || ID_{card} || P_{secret})$$

$$S_R = \ddot{A}_R{}^{d+1} \bmod n$$

(1.2) The terminal forwards S_R and R to the smart card.

Step 2: When the smart card receives S_R and R; it does the following steps.

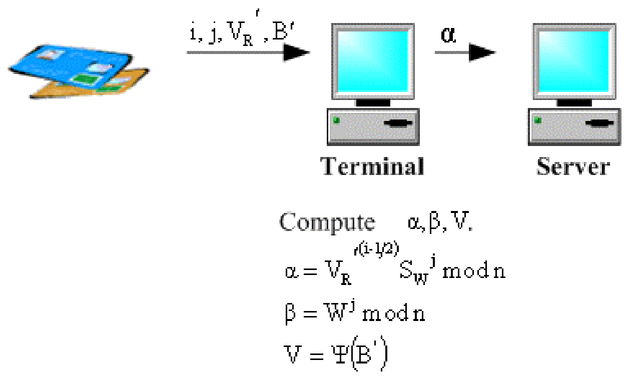


Figure 9. Confirmation Protocol – Step 2

(2.1) The smart card computes $Ä_M$, $Ä_R$ and V_R according the received data, S_R and R, from the terminal.

$$Ä_M = H(H(M)||ID_{card}||P_{secret})$$

$$Ä_R = H(H(R)\,||\,ID_{card}\,||\,P_{secret})$$

$$V_R = \left(S_M \bullet \left(S_R / Ä_R\right)\right)^2 \bmod n$$

(2.2) According the process described in section 3.2.2, the smart card generates $t+1$ random numbers a_1, a_2,..., a_t and r and computes a_0 by the following equations.

$a_0 = \left((Ä_M Ä_R)\prod_{i=1}^{r} a_i\right) \bmod n$. (Note that this computation is different from that described in section 3.2.1).

(2.3) The smart card computes vectors, B, B' (the computation is the same as the notation in section 3.2.1) and V_R'.

$V_R' = V_R^2 \bmod n$

(2.4) The smart card generates i and j where $i, j \in Z_n$ and i be the odd number, then sends i, j, V_R' and B' to the terminal.

(2.5) The terminal computes α, β and V, and then sends α to the server.

$\acute{a} = V_R'^{(i-1)/2} S_W^{\,j} \bmod n$

$\beta = W^j \bmod n$

$V = \varnothing(B')$

Step 3: After the server receives α, it does the following steps.

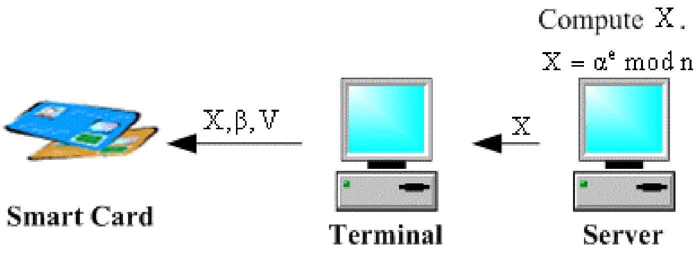


Figure 10. Confirmation Protocol – Step 3,4

(3.1) The server computes X and then sends it to the terminal.

$X = \acute{a}^e \bmod n$

(3.2) The terminal sends X, β and V to the smart card.

Step 4: After the smart card received X , $\mathbf{b}$ and V , it does the following steps.

(4.1) The smart card computes U and g .

$$U = \ddot{O}^{-1}(V)$$

$$g = \grave{E}(U, X)$$

(4.2) The smart card checks whether $\mathbf{b} \bullet \mathbf{g}$ is equal to X . If the answer is equal, the signature is valid; otherwise undetermined.

Lemma 4. Let $\hat{a} = W^{j} \bmod n$, $\tilde{a} = \grave{E}(U, X)$ and $X = \acute{a}^{e} \bmod n$ such that $\hat{a} \cdot \tilde{a} = X$.

$$\begin{aligned} & \hat{a} \cdot \tilde{a} \\ & = W^{j} \grave{E}(U, X) \bmod n \\ & = W^{j}\left(u_0\left(\prod_{i=1}^{r} u_i\right)^{-1}\right) \bmod n \\ & = W^{j}(\ddot{A}_M \ddot{A}_R)^{i-1} \bmod n \end{aligned}$$

$$\begin{aligned} & X = \acute{a}^{e} \bmod n \\ & = \left((V_R^{'})^{(i-1)/2} S_W{}^{j}\right)^{e} \bmod n \\ & = \left((V_R)^{i-1} S_W{}^{j}\right)^{e} \bmod n \\ & = W^{j}(\ddot{A}_M \ddot{A}_R)^{i-1} \bmod n \end{aligned}$$

4.1.4. Deniable Protocol

Step 1: The process is the same as the step 1 of the confirmation protocol.

Step 2:

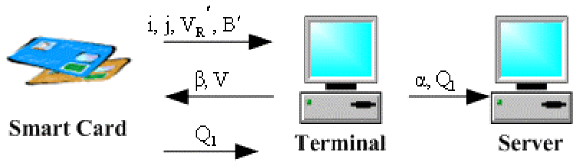


Figure 11. Deniable Protocol – Step 2

(2.1) The terminal computes the one-way hash function on the faked message, generates a random number and signs the random number. And then it sends the results, $H(R)$, S_{R_1} and R, to the smart card.

(2.2) The smart card computes $Ä_{M_{challenge}}$, $Ä_R$ and V_R', where

$$Ä_{M_{challenge}} = H\left(H\left(M_{challenge}\right) || ID_{card} || P_{secret}\right)$$

(2.3) The terminal does the same steps as the confirmation protocol to compute a, b and V; then sends b and V to the smart card.

(2.4) Te smart card computes g and Q_1; and sends Q_1 back to the terminal, where

$$Q_1 = g \cdot b \bmod n$$

(2.5) The terminal send a and Q_1 to the server.

Step 3: After the server receives a and Q_1, it does the following steps.

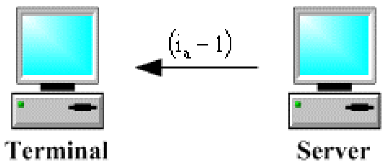


Figure 12. Deniable Protocol – Step 3,4

(3.1) The server computes Q_2 and finds some i_a satisfied the following condition; then sends $(i_a - 1)$ back to the terminal.

$$Q_1 / Q_2 = \left(Ä_{M_{challenge}} / Ä_M \right)^{(i_a - 1)} \bmod n .$$

$$Q_2 = á^e \bmod n$$

Step 4: The terminal checks whether $(i_a - 1)$ is equal to $(i - 1)$. If the answer is positive, the signature is invalid; otherwise undetermined.

4.1.5. Signatures Request Protocol

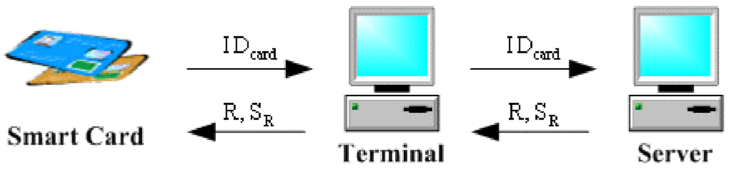


Figure 13. Signatures Request Protocol

Step 1: The smart card sends ID_{card} to the terminal and the terminal forwards it to the server.

Step 2: If ID_{card} is valid, the server generates a random number and computes S_{R_1}, $Ä_{M_{new}}$ and ΔR. And then it sends S_{R_1} and R to the terminal and the terminal forwards them to the smart card, where

$$Ä_R = H\left(H(R) \| ID_{card} \| P_{secret} \right)$$

$$Ä_{M_{new}} = H\left(H\left(M_{new}\right)\|ID_{card}\|P_{secret}\right)$$

$$S_R = Ä_R\left(Ä_{M_{new}}\right)^d \mod n$$

Step 3: The smart card computes $S_{M_{new}}$ and it is the signatures of a new message M_{new}, where

$$S_{M_{new}} = \left(S_R / Ä_R\right) \mod n$$

Lemma 5. Let $S_{M_{new}} = \left(S_R / Ä_R\right) \mod n$. Then $S_{M_{new}} = \left(Ä_{M_{new}}\right)^d \mod n$.

Proof.

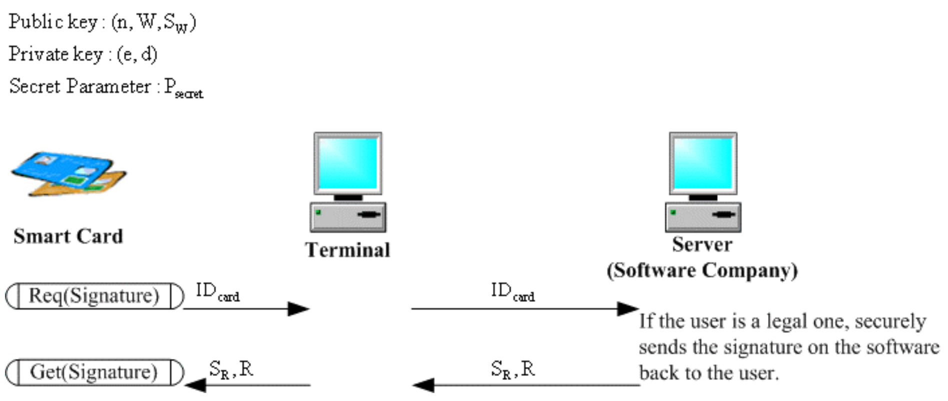
$$S_{M_{new}} = \left(S_R / Ä_R\right) \mod n$$

$$= \left(Ä_R\left(Ä_{M_{new}}\right)^d / Ä_R\right) \mod n$$

$$= \left(Ä_{M_{new}}\right)^d \mod n$$

4.1.6. Applications on software protection

By our proposed protocol, one can use it to protect the intellectual property of a software. In such application, a software company must generate the signature for each of his software product, the key pair and the secret parameter for each user. With the secret parameter, the software company builds a secure tunnel with the users and uses it to securely transmit the signatures on the Internet. If the user has the valid signature, he will have the permission for using the software product.



Public key : (n, W, S_W)
Private key : (e, d)
Secret Parameter : P_{secret}
Smart Card
Terminal
Server
(Software Company)
Req(Signature)
ID_{card}
ID_{card}
If the user is a legal one, securely sends the signature on the software back to the user.
Get(Signature)
S_R, R
S_R, R

Figure 14. The system model of software protection system

In Figure 14, it shows how the user requests the valid software to the server. The process is the same as the signature request protocol described in the section 4.1.5.

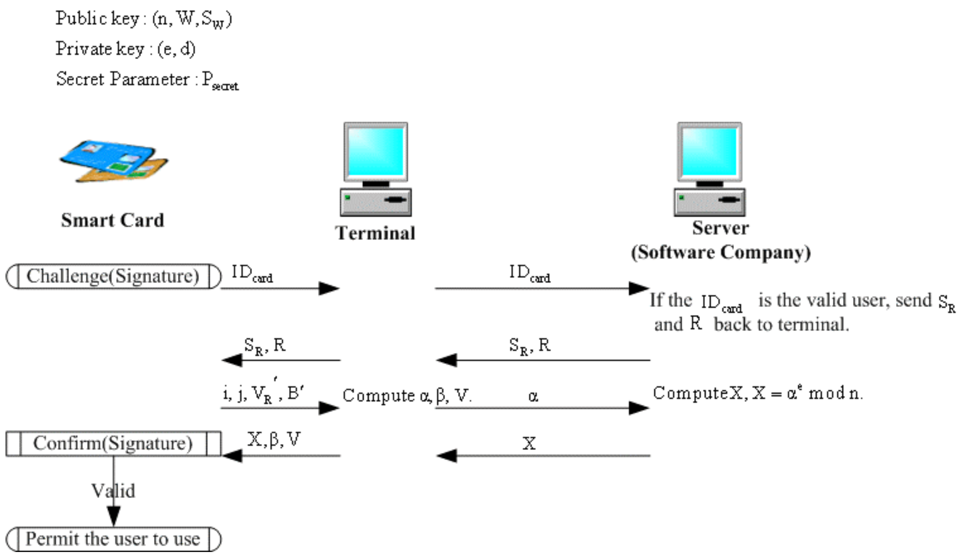


Figure 15. Software Protection Model – Permission Phase

In Figure 15, it shows how the user uses the signature to get the permission of the software. If the user is the legal one, he can get the valid signature on the software during the signature request phase. By the confirmation protocol, the software uses the signature to confirm that the signature is valid. If signature is valid, the software can be executed by the user.

4.2. Elliptic Curve Undeniable Signatures Using Smart Cards

In this section, we will describe the notation briefly first and then introduce the proposed scheme. The proposed designated confirmer signature scheme is based on ECC (The elliptic curve undeniable signature is described in Appendix A) and use the smart card as the authentication tool. Thus, there is an authentication scheme using smart card inside the designated confirmer signature scheme. The authentication scheme will provide mutual authentication and let the both ends share a common secret data that is useful in the designated confirmer signature scheme.

4.2.1. Notations

- E : An elliptic curve defined over Z_p where Z_p denotes the multiplicative group modulo p.

- G : A base point $G \in E(Z_p)$ of order n which is prime.
- (d_u, Q_u) : The key pair of a user. The d_u is the private key, Q_u is the public key and $Q_u = d_u \times G$ ("$\times$" indicates the multiplication of a number and an elliptic curve point).
- $H(m)$: An one way hash function with collision resistant and with the input m.
- $Cert_{user}$: The certificate of the user and there are following elements in it: user's personal information, user's public key, CA's information, and CA's public key.
- $E_x(m)$: The encrypted function using cipher key x and m is the plaintext.
- $D_x(m)$: The decrypted function using cipher key x and c is the ciphertext.
- $A \| B$: The concatenation of A and B.

4.2.2. Proposed Scheme

In this scheme, there are four roles: the signer, the confirmer, the verifier, and the smart card. If the verifier wants to verify a signature signed by the signer, he must first insert his smart card into a card reader that is attached to a terminal. Then the authentication scheme will be executed and the confirmer and the verifier will authenticate each other. There are four phases in the scheme: Smart card initialization phase, signature generation phase, verifier authentication phase and signature verification phase.

Smart Card Initialization Phase:

Step 1: The smart card selects a random number $d_v \in Z_n^*$ as the verifier's private key and computes $Q_v = d_v \times G$ as the public key.

Step 2: The smart card sends Q_v and verifier's information to the CA. After checking the information of the verifier, CA generates the certificate $Cert_{verifier}$ and sends it back to the smart card.

Step 3: Because the public key Q_v and other verifier's information have been included in the certificate, smart card just writes $(d_v, Cert_{verifier})$ into its memory.

Signature Generation Phase:

Step 1: The signer selects a random number $d_s \in Z_n^*$ as his private key and computes $Q_s = d_s \times G$ as his public key.

Step 2: The confirmer also generates his key pair (d_c, Q_c) and requests a certificate $Cert_{confirmer}$ from CA.

Step 3: The signer selects an arbitrary number $c \in Z_n^*$, computes $c \times G = (x_1, y_1)$, and let $r = x_1$.

Step 4: The signer computes $d_1 = (t + H(m \| r)) \times G$, $d_2 = t \times Q_c$, and let $d = (d_1, d_2) = Com(m \| r, Q_c)$, i.e. d is the confirmer commitment.

Step 5: The signer computes $s = c - H(d) \cdot d_s (\bmod n)$ and the signature is (s, d).

Verifier Authentication Phase:

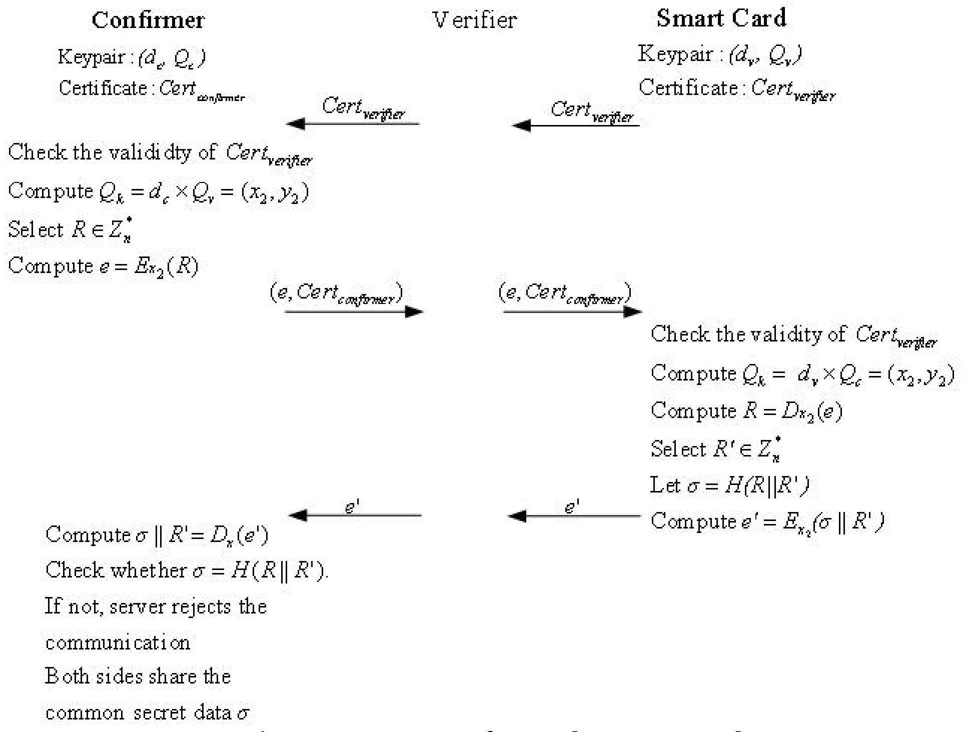
Step 1: If the verifier wants to verify a signature, he first inserts the smart card into the card reader. Then the verifier could get $Cert_{verifier}$ from the smart card and sends it to the confirmer.

Step 2: After confirming the validity of $Cert_{verifier}$, the confirmer computes $Q_k = d_c \times Q_v = (x_2, y_2)$, let x_2 be the mutually session key, and selects a random number $R \in Z_n^*$. Then he encrypts R using session key x_2, and sends the encrypted data $e = E_{x_2}(R)$ and $Cert_{confirmer}$ to the verifier.

Step 3: The smart card checks the validity of $Cert_{confirmer}$, and computes $Q_k = d_v \times Q_c = (x_2, y_2)$, $R = D_{x_2}(e)$ where x_2 is the authentication key. The smart card selects another random

$R' \in Z_n^*$ and computes $s = H(R \| R')$. Then the smart card encrypts σ and R' using the session key x_2 to obtain $e' = E_{x_2}(s \| R')$. Then the smart card sends e' back to the confirmer.

Step 4: After computing $s \| R' = D_{x_2}(e')$, the confirmer could get s and R' respectively. Then the confirmer checks whether $s = H(R \| R')$. If not, the confirmer reject the following communication, else the both sides accomplish the authentication to each other and they share the common secret data s.



Confirmer
Verifier
Smart Card
Keypair : (d_c, Q_c)
Certificate : $Cert_{confirmer}$
Keypair : (d_v, Q_v)
Certificate : $Cert_{verifier}$
$Cert_{verifier}$
$Cert_{verifier}$
Check the valididty of $Cert_{verifier}$
Compute $Q_k = d_c \times Q_v = (x_2, y_2)$
Select $R \in Z_n^*$
Compute $e = E_{x_2}(R)$
$(e, Cert_{confirmer})$
$(e, Cert_{confirmer})$
Check the validity of $Cert_{verifier}$
Compute $Q_k = d_v \times Q_c = (x_2, y_2)$
Compute $R = D_{x_2}(e)$
Select $R' \in Z_n^*$
Let $\sigma = H(R\|R')$
Compute $e' = E_{x_2}(\sigma \| R')$
e'
e'
Compute $\sigma \| R' = D_x(e')$
Check whether $\sigma = H(R \| R')$.
If not, server rejects the communication
Both sides share the common secret data σ

Figure 16. Verifier Authentication Scheme

Signature Verification Phase:

Step 1: The verifier select two random numbers $u, v \in Z_n^*$, and compute $a = (u \times G + v \times Q_c)$. The verifier sends a to smart card.

Step 2: The smart card computes $a' = \boldsymbol{s} \times a$, and sends a' to the confirmer through the verifier.

Step 3: The confirmer selects three random numbers $k, \hat{k}, w \in Z_p^*$, computes $\boldsymbol{l}_a = k \times G$, $\hat{\boldsymbol{l}}_a = \hat{k} \times G$, $\boldsymbol{l}_b = k \times d_1$, $\hat{\boldsymbol{l}}_b = \hat{k} \times d_1$ and sends ($\boldsymbol{l}_a$, $\boldsymbol{l}_b$, $\hat{\boldsymbol{l}}_a$, $\hat{\boldsymbol{l}}_b$, w) to the verifier.

Step 4: The verifier sends (u, v) to the confirmer.

Step 5: The confirmer checks if $a' = \boldsymbol{s} \times (u \times G + v \times Q_c)$. If it's incorrect, then reject, else the confirmer computes $s = k \cdot \boldsymbol{s} - (v + w) \cdot d_c$, $\hat{s} = \hat{k} \cdot \boldsymbol{s} - (v + w) \cdot k$. The confirmer sends $(s, \hat{s})$ to the verifier.

Step 6: He verifier computes $p_1 = s \times G + (v + w) \times Q_c$, $p_2 = \hat{s} \times G + (v + w) \times \boldsymbol{l}_a$, $p_3 = \hat{s} \times d_1 + (v + w) \times \boldsymbol{l}_b$, $p_4 = s \times d_1 + (v + w) \times (H(m \| r) \times Q_c + d_2)$, and sends (p_1, p_2, p_3, p_4) to smart card.

Step 7: Smart card computes $p_1^{'} = s^{-1} \times p_1$, $p_2^{'} = s^{-1} \times p_2$, $p_3^{'} = s^{-1} \times p_3$, $p_4^{'} = s^{-1} \times p_4$, and sends ($p_1^{'}, p_2^{'}, p_3^{'}, p_4^{'}$) back to the verifier.

Step 8: The verifier checks whether $p_1^{'} = l_a$, $p_2^{'} = \hat{l}_a$, $p_3^{'} = \hat{l}_b$, and finally concludes: If $p_4^{'} = l_b$, then the signature is valid. If $p_4^{'} \neq l_b$, then the signature is invalid.

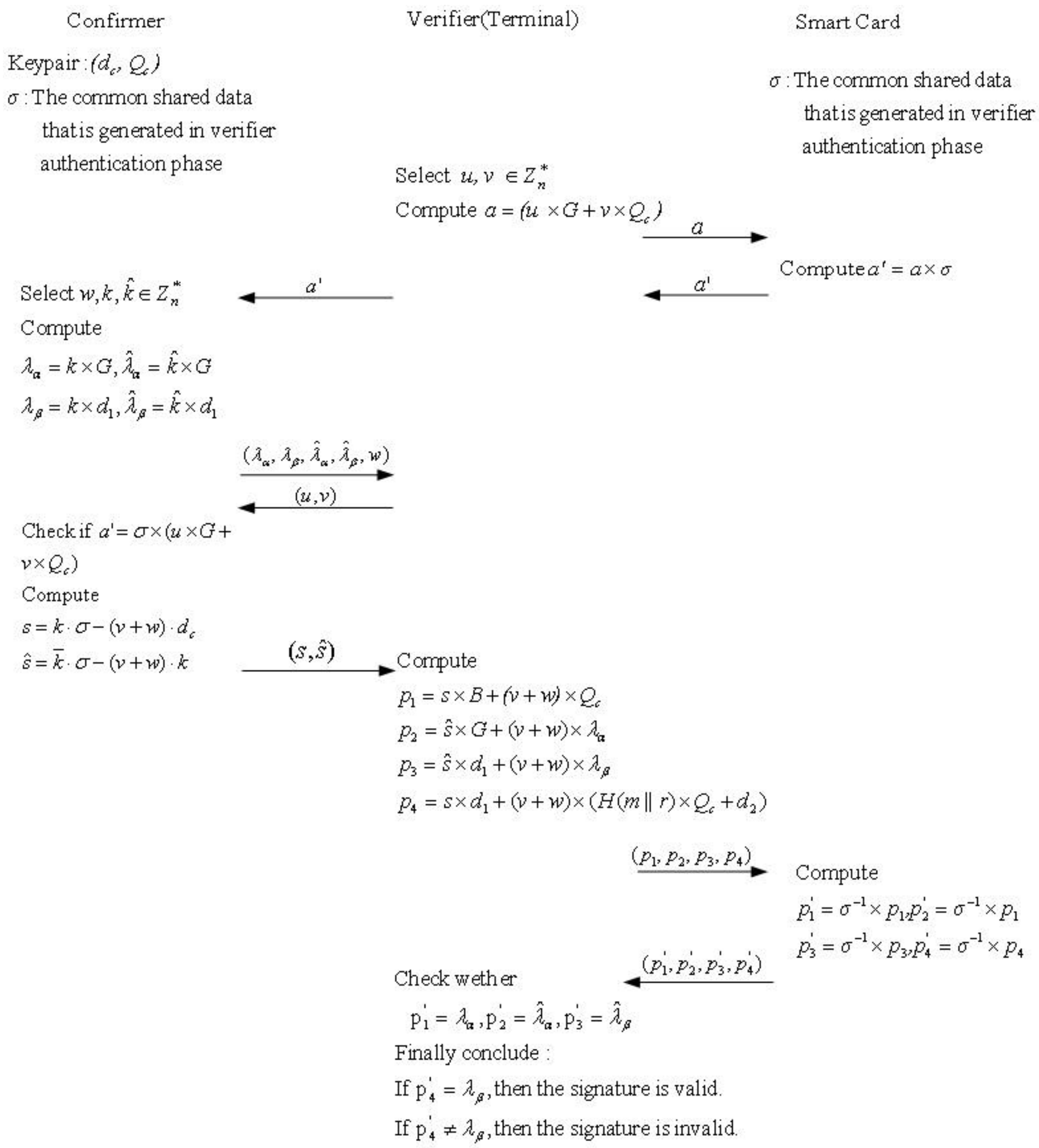

Confirmer
Verifier(Terminal)
Smart Card
Keypair : (d_c, Q_c)
σ : The common shared data that is generated in verifier authentication phase
σ : The common shared data that is generated in verifier authentication phase
Select $u, v \in Z_n^*$
Compute $a = (u \times G + v \times Q_c)$
a
Compute $a' = a \times \sigma$
a'
a'
Select $w, k, \hat{k} \in Z_n^*$
Compute
$\lambda_\alpha = k \times G, \hat{\lambda}_\alpha = \hat{k} \times G$
$\lambda_\beta = k \times d_1, \hat{\lambda}_\beta = \hat{k} \times d_1$
$(\lambda_\alpha, \lambda_\beta, \hat{\lambda}_\alpha, \hat{\lambda}_\beta, w)$
(u, v)
Check if $a' = \sigma \times (u \times G + v \times Q_c)$
Compute
$s = k \cdot \sigma - (v + w) \cdot d_c$
$\hat{s} = \bar{k} \cdot \sigma - (v + w) \cdot k$
$(s, \hat{s})$
Compute
$p_1 = s \times B + (v + w) \times Q_c$
$p_2 = \hat{s} \times G + (v + w) \times \lambda_\alpha$
$p_3 = \hat{s} \times d_1 + (v + w) \times \lambda_\beta$
$p_4 = s \times d_1 + (v + w) \times (H(m \| r) \times Q_c + d_2)$
(p_1, p_2, p_3, p_4)
Compute
$p_1' = \sigma^{-1} \times p_1, p_2' = \sigma^{-1} \times p_1$
$p_3' = \sigma^{-1} \times p_3, p_4' = \sigma^{-1} \times p_4$
(p_1', p_2', p_3', p_4')
Check wether
$p_1' = \lambda_\alpha, p_2' = \hat{\lambda}_\alpha, p_3' = \hat{\lambda}_\beta$
Finally conclude :
If $p_4' = \lambda_\beta$, then the signature is valid.
If $p_4' \neq \lambda_\beta$, then the signature is invalid.

Figure 17. Elliptic curve designated confirmer signature scheme --- verification phase

5. Complexity and Security Analysis

5.1. Complexity Analysis of the RSA-Based Undeniable Signature Scheme Using Smart Card

For simplification, we let t be the size of the vector (described in Section 3.2.1.) and r be an integer, $r < (t-1)$.

In the confirmation protocol, the smart card does two hashing, $(t+2r+1)$ multiplications, one division and one inverse operation. According the Euclid's extended algorithm, the inverse operation needs about $(0.843 * \ln(n) + 1.47)$ divisions.

About the communication cost, the protocol needs to send $(2t+6) * \ln(n)$ integers. In the signatures request protocol, it just needs one multiplication and two hashing. From the above analysis, we can see that only very low computation is required for the smart card.

5.2. Security Analysis of the RSA-Based Undeniable Signature Scheme Using Smart Card

In the confirmation protocol, the attacker just can get ID_{card}, R, V_R' and B'. If the attacker wants to know the valid signature, he needs to compute Δ_M with B'. The difficulty is the same as the factorization problem. If the attacker wants to compute the valid signature, it is computational infeasible. Besides, it is also computationally infeasible for the attacker to fake the signature with V_R' and B'.

Yvo and Moti proposed some attacks on the Chaum's undeniable signature protocol. The attacks are based on the assumption that the attacker must have the signer's signature. Such assumption is improper on our protocol. In our protocol, the signature is protected on the smart card. Because the smart card has the features tamper detection and zeroization, when it is damaged by others it will destroy all the confidential data on it.

5.3. Security Analysis of Elliptic Curve Undeniable Signatures Scheme Using Smart Cards

Security of the designated confirmer signature: We have introduced the four security requirements in Chapter 2.4. The proposed designated confirmer signature scheme is based on Michels and Stadler's scheme and the security requirements of the original scheme have been discussed. However, in the proposed scheme, what we do are to add the authentication scheme in the front of the verification scheme and to add a secret parameter shared between the confirmer and the smart card. These changes won't influence the security discussed in the original paper.

Mutual authentication: The confirmer and the smart card exchange their certificates and check the validity of the certificates to each other in the beginning of the scheme. However, the certificate only proves that the public key is correct and owned by the user recorded in the certificate. The both sides must prove that they did have the private keys corresponding to the public key in the certificates. In fact, after decrypting *e'* to gets R' and σ and confirming that σ is equal to $H(R \| R')$ at Step 4 in the verifier authentication phase, the confirmer could believe that the smart card have the private key because the smart card could get R by decrypting $e = E_{x_2}(R)$ using the authentication key which is computed by its own private key and confirmer's public key.

Replaying attack: The common secret parameter σ is similar to the nonce used in general authentication scheme. At Step 5 in the signature verification phase, if the verifier doesn't execute the authentication scheme using the smart card and doesn't obtain the correct secret parameter σ, the confirmer will terminate the verification communication.

Verify the signature without smart card: Because the private key never leaves the smart card and the smart card is a special device with character of tamper-proof, the verifier is impossible to pass the authentication in the beginning of the verification scheme. The verifier still can't verify the signature without smart card in the final step even though he passes the authentication using the smart card. This is because that the verifier needs the smart card to help him compute the final result.

6. Conclusion

In this thesis we propose two practical protocols. The first scheme is based on RSA and we utilize the server-aided computation to let smart card make RSA computation with assistance of terminal or server. In this scheme, all signatures will be protected by smart card, and therefore no one could attack the system via the signature. In the second proposed scheme, we design a designated confirmer signature based on Elliptic Curve Cryptosystem with mutual authentication. In the protocol only the verifiers with legitimate smart cards could verify the signature. Besides, we design a system on the smart card environment according to the first protocol. This system can be used as mechanism for the copyright protection of software. We design the signatures request protocol to securely upload another signature on the smart card. Through this protocol, the smart card can dynamically maintain the signature and be securely uploaded another signature on it.

Reference

[1] A. Fujioka, T. Okamoto, K. Ohta, "Interactive Bi-Proof Systems and Undeniable Signature Schemes," In Advances in Cryptology - proceedings of Eurocrypt'91, Lecture Notes in Computer Science, Springer-Verlag, 1992, pp. 243-256.

[2] A. Menezes: "Elliptic Curve Public Key Cryptosystems," Kluwer Academic Publishers, 1993.

[3] B. Schneier, "A Primer on Authentication and Digital Signatures," Computer Security Journal, vol.10, no. 2, 1994, pp. 38-40.

[4] C. Boyd and E. Foo, "Off-line Fair Payment Protocols Using Convertible Signature," In Advances in Cryptology - proceedings of Asiacrypt' 98, , Springer-Verlag, 1998 pp. 271-285.

[5] C.H. Lin and C.C. Chang, "Server-Aided Computation Protocol for RSA Enciphering Algorithm," International Journal of Computer Mathematics Vol. 53, 1994, pp. 149-155.

[6] C. H. Lin, C. T. Wang and C. C. Chang, "A Group-oriented (t, n) Undeniable Signature Scheme without Trusted Center," Proceedings of the 1996 1st Australasian Conference on Information Security and Privacy, ACISP'96, 1996, pp. 266-274.

[7] C. P. Schnorr, "Efficient Signature Generation for Smart Cards,"

Journal of Cryptology, Vol.4, 1991, pp.161-174.

[8] D. Chaum and H. Van Antwerpen, "Undeniable Signatures," Advances in Cryptology- Crypt' 89, August 22-24, 1989, pp.212-216.

[9] D. Chaum, "Zero-Knowledge Undeniable Signature," Advances in Cryptology- Eurocrypt' 90, Springer Verlag, 1990, pp. 458-464.

[10] D. Chaum, "Some Weakness of Weaknesses of Undeniable Signatures," Leture Notes in Computer Science 547, Advances in Cryptology-Eurocrypt' 91, Springer Verlag, 1992, pp.554-556.

[11] D. Chaum, "Designated Confirmer Signatures," Advances in Cryptology- Eurocrypt'94, 1994, pp.86-91.

[12] D. Johnson, A. Menezes, "The Elliptic Curve Digital Signature Algorithm," Technical Report COOR 99-34, Dept. of C&O, University of Waterloo, Canada, available at: http://www.cacr.math.uwaterloo.ca

[13] Ivan Damgård, Torben P. Pedersen, "New Convertible Undeniable Signature Schemes," Advances Cryptology: Workshop on the Theory and Application of Cryptographic Techniques, EUROCRYPT '96 (Zaragoza, Spain, May 12-16, 1996). LNCS; Springer-Verlag, (1070): 372-386, 1996.

[14] J. Boyar, D. Chaum, I. Damgard, T. Pedersen, "Convertible

Undeniable Signature," Lecture Notes in Computer Science 537, Advances in Cryptology: Proc. Crypto ' 90, Springer Verlag, 1991, pp.189-205.

[15] L. Chen, "Efficient Fair Exchange with Verifiable Confirmation of Signatures," In Advances in Cryptology - proceedings of Asiacrypt' 98, Springer-Verlag, 1998, pp.286-299.

[16] L. Harn and S. Yang, "Group Oriented Undeniable Signature Schemes Without the Assistance of a Mutually Trusted Party," Advance in Cryptology-Of Auscrypt' 92, Dec. 1992.

[17] M. Jakobsson, "Blackmailing Using Undeniable Signatures," Lecture Notes in Computer Science 950, Advances in Cryptology-Eurocrypt' 94, Springer Verlag, 1995, pp. 425-427

[18] M. Jakobsson, K.Sako, and R. Impagliazzo, "Designated Verifier Proofs and Their Application," In Advances in Cryptology: Proc. EUROCRYPT ' 96, LNCS 1070, Springer Verlag, 1996, pp. 143-154.

[19] M. Michels, "Breaking and Repairing a Convertible Undeniable Signature Scheme," In Proceedings of the 1996 ACM Conference on Computer and Communications Security, 1996, pp. 148-152.

[20] M. Michels and M. Stadler, "Efficient Convertible Undeniable Signature Schemes," Proc. 4th Annual Workshop on Selected Areas in Cryptography (SAC' 97), 1997, pp. 231-243.

[21] M. Michels and M. Stadler, "Generic Constructions for Secure and Efficient Confirmer Signature Schemes," In Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 406-421.

[22] N. Koblitz, "Elliptic Curve Cryptosystems: " Mathematics of Computation, 48 (1987), pp. 203-209.

[23] N. Koblitz: "A Course in Number Theory and Cryptography," New York, NY: Springer-Verlag, Second edition, 1994.

[24] N. Y. Lee and T. Hwang, "Group-oriented Undeniable Signature Schemes without a Trusted Center," 1998 International Computer Symposium (ICS'98), Tainan, Taiwan, R. O. C., Dec. 17-19, 1998, pp. 165-170.

[25] N. Y. Lee and T. Hwang, "Group-oriented Undeniable Signature Schemes with a Trusted Center," Computer Communications, vol. 22, no. 8, 1999, pp. 730-734.

[26] R.Gennaro, H.Krawczyk and T.Rabin, "RSA-Based Undeniable Signatures," Preliminary version in proceedings of CRYPTO'97, Springer-Verlag, LNCS 1294, pp.132-149.

[27] R. Gennaro, H. Krawczyk and T. Rabin, "Undeniable Certificates," Electronic Letters, vol. 35, no. 20, Sep. 1999, pp. 1723-1724.

[28] S. Kawamura and A. Shimbo, "Performance Analysis of Server-Aided Secret Computation Protocols for the RSA Cryptosystem," The Transactions of The Institute of Electronics, Information and Communication Engineers IEICE, Vol. E73, NO. 7, 1990, pp.1073-1080.

[29] S. Vanstone: "Responses to NIST' s Proposal," Communications of the ACM, 35, July 1992, pp. 50-52.

[30] T. Matsumoto, K. Kato and H. Imai, "Speeding up Secret Computations with Insecure Auxiliary Devices," Crypto ' 88, Lecture Note on Computer Sciences 403, Springer-Verlag, Berlin 1990, pp.497-506.

[31] T. Okamoto, "Designated Confirmer Signatures and Public-key Encryption Are Equivalent," In Advances in Cryptology - Crypto'94, Lecture Notes in Computer Science (LNCS) 839, Springer-Verlag, 1994, pp. 61-74.

[32] V. Miller: "Uses of Elliptic Curves in Cryptography," In H. C. Williams, editor, Advances in Cryptology-Crypto' 85, Proceedings, Lecture Notes in Compute Science, No. 218, NY: Springer-Verlag, 1985, pp.417-426.

[33] Y. Desmedt, M. Yung: "Weaknesses of Undeniable Signature Schemes," Lecture Notes in Computer Science547, Advances in

Cryptology-Eurocrypt' 91, Springer Verlag, 1992, pp.205-220.

[34] International Organization for Standards, http://www.iso.ch/

[35] PC/SC Workgroup, http://www.pcscworkgroup.com/

[36] RSA Security Inc., http://www.rsasecurity.com/

[37] Open Card consortium, http://www.opencard.org/

[38] Java Card Forum, http://www.javacardforum.org/

[39] MAOSCO Ltd , http://www.multos.com/

[40] Mondex web site, http://www.mondex.com/

[41] CEPSCO, http://www.cepsco.com/

[42] EMVCO, http://www.emvco.com

Appendix A: Elliptic-Curve Undeniable Signature

A.1. Elliptic-curve Undeniable Signatures Based on Chaum’ s

The first scheme is based on Chaum’ s. It is described as follows:

Key Generation Phase: The user Alice follows these steps:

Step 1: Select an elliptic curve E defined over Z_p.

Step 2: Select a base point $B \in E(Z_p)$ of order n which is prime.

Step 3: Select a random number d as her private key, where $d \in [1, n-1]$.

Step 4: Computes $Q = d$ ´ B as her public key. (“$\times$” indicates the multiplication of a number and an elliptic curve point)

Commitment Phase:

If Alice wants to sign a message m, she computes $r = dm \bmod n$, $Z = r$´$Q = (x_1, y_1)$ as his undeniable signature and sends Z and m to the verifier Bob.

Verification Phase:

Step 1: Bob selects two random integers $a, b \in [1, n-1]$, and calculates $W = a \times Z + b \times Q = (x_2, y_2)$, then sends W to Alice.

Step 2: After receiving *W*, Alice calculates $R = (d^{-1} \bmod n) \times W$, where d^{-1} is the multiplicative inverse of d. The result R is sent back to Bob.

Step 3: The verifier computes

$$s = am \bmod n$$

$$R' = s \times Q + b \times B$$

If $R' = R$, then the message m is authentic.

Lemma 6. In verification phase, if $R' = R$, then the verifier can authenticate the message m and the signature Z.

Proof:

$$R = (d^{-1} \bmod n) \times W$$

$$= (d^{-1} \bmod n) \times (a \times Z + b \times Q)$$

$$= (d^{-1} \bmod n) \times ((adm \bmod n) \times Q + (bd \bmod n) \times B))$$

$$= (am \bmod n) \times Q + b \times B$$

$$= s \times Q + b \times B$$

$$= R'$$

A.2. Elliptic-curve Group Undeniable Signatures

Suppose that there are k users in a group, and every member of the group has a private key. The users are connected in an order of U_1,

U_2 , .., U_k as in Harn and Yang's scheme. The three phases are as following:

Key Generation Phase:

Step 1: Step 1: Select an elliptic curve E defined over Z_p.

Step 2: Step 2: Select a base point $B \in E(Z_p)$ of order n which is prime.

Step 3: Step 3: Each member in the group selects a number d_i , where $d_i \in [1, n-1]$, $1 \quad i \quad k$.

Step 4: Step 4: The group public key is computed as Q = Q_n = $d_n \times (Q_{n-1})$ = $(d_1 d_2 \cdots d_k \bmod n)$ ´B

Commitment Phase:

All members of the group compute $t = d_1 d_2 \cdots d_k \bmod n$, Z = Z_n = $d_n \times (Z_{n-1})$ = $(tm \bmod n) \times Q = (x_1, y_1)$ as their group undeniable signature and send Z and m to the verifier Bob.

Verification Phase:

Step 1: Bob selects two integers a,b$\in$[1,n-1], calculates $W = a \times Z + b \times Q = (x_2, y_2)$, then send W to the signing group.

Step 2: After receiving R, the group members calculate

$t' = t^{-1} = d_1^{-1} d_2^{-1} .. d_k^{-1} \bmod n$, $R = t' \times W$, where d_{i-1} is the multiplicative inverse of d_i. The result R is sent back to the verifier.

Step 3: The verifier computes

$$R' = (am \bmod n) \times Q + b \times B$$

If $R' = R$, then the message m is authentic.

Lemma 7. In the verification phase, if $R' = R$, then the verifier can authenticate the message m and the signature Z.

Proof:

$$R = (t^{-1} \bmod n) \times W$$

$$= (t^{-1} \bmod n) \times (a \times Z + b \times Q)$$

$$= (t^{-1} \bmod n) \times (a \times ((tm \bmod n) \times Q) + (b \times ((t \bmod n) \times B))$$

$$= (am \bmod n) \times Q + b \times B$$