Authentication Techniques

Annual progress report by

Manik Lal Das
(Roll No. 02429701)

Supervisor: Dr. Deepak B Phatak, KReSIT, IIT Bombay
Co-supervisor: Dr. Ashutosh Saxena, IDRBT Hyderabad

Kanwal Rekhi School of Information Technology
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076, India.
August 2005
Abstract

Authentication is a process that attests to the attributes of participants in information system and is intended to promote trust in communicating parties. The technological methods and specifications used for authentication are often based on cryptographic techniques. Based on the applications requirements, the authentication process can be classified into two categories: Data authentication and Entity authentication. In this report we discuss our recent developments in such authentication techniques.

The most widely used data authentication techniques are digital signature, proxy signature, multi-signature, etc. In this report, we focus on proxy and multi signatures. A proxy signature allows a designated entity, called a proxy signer, to sign the message on behalf of the original signer. In 1996, Mambo et al. introduced the concept of proxy signature. Since then, several new schemes and improvements have been proposed on proxy signatures; however, proxy revocation is not addressed in most of the schemes. Recently, bilinear pairings (in short pairings), namely the Weil-pairing and the Tate-pairing of algebraic curves have found interesting applications in identity (ID) based cryptosystem. The main advantage of ID-based cryptography is that it avoids public key certification burden. The public key of a user is his/her identity, e.g., e-mail, social security number, etc. It is observed that most of the works on ID-based cryptosystem using pairings suffer with the key escrow problem and require secure channel in the key issuance stage. We studied the existing proxy signature schemes observed that most of the schemes are defected by proxy revocation mechanism, which is a practical requirement. Using bilinear pairings, we present a proxy signature scheme that provides effective revocation mechanism, where the scheme does not require secure channel in the key issuance stage and avoid the key escrow problem. Subsequently, we propose a secure key issuance protocol and extend its scope to multi-signatures. In both, proxy and multi-signature schemes, we analyze the security and efficiency of the schemes.

Apart from the data authentication, we present three schemes for ‘remote systems authentication’ using smart cards, which plays a crucial role in entity authentication. The use of smart card restricts the participants from distribution of their login-IDs and avoids the scenario of many logged in users with the same login-ID. In our schemes, the remote system does not maintain any password or verifier table for the validation of a login request, in turn, the scheme is secure against stolen-verifier and insider attacks. In addition, the schemes provide a flexible password change option, where users can change their passwords at any time without any assistance from the remote system.
## Contents

1 Introduction .......................... 5  
   1.1 Data Authentication .................. 6  
      1.1.1 Proxy Signature .................. 6  
      1.1.2 Multi-signature ................. 7  
   1.2 Entity Authentication ............... 7  
   1.3 Elements of an Authentication System .... 8  
   1.4 Organization of the Report .......... 9  

2 Preliminaries ....................... 10  
   2.1 Discrete Logarithm Problem ........... 10  
      2.1.1 Bilinear Pairings ............... 11  
   2.2 Integer Factorization Problem ........ 13  

3 Proxy Signature ..................... 15  
   3.1 Introduction ....................... 15  
   3.2 Security Properties ................. 16  
   3.3 The Proposed Schemes ............... 17  
      3.3.1 Proxy Signature Scheme based on RSA .... 17  
      3.3.2 Proxy Signature Scheme using Bilinear Pairings .... 17  
         3.3.2.1 The Model .................. 18  
         3.3.2.2 The Scheme ................. 19  
         3.3.2.3 Analysis of the Scheme ........ 21  
   3.4 Conclusion ......................... 25  

4 Multi Signatures .................... 26  
   4.1 Introduction ....................... 26  
   4.2 The Proposed Schemes ............... 26  
      4.2.1 Multi-signature Scheme based on RSA .... 26  
         4.2.1.1 The Scheme [Das et al., 2004c] .... 27  
         4.2.1.2 Analysis .................. 29  
      4.2.2 Multi-signature Scheme using Bilinear Pairings .... 29  
         4.2.2.1 The Model .................. 30  
         4.2.2.2 Detailed Description .......... 31  
         4.2.2.3 The Scheme ................. 33
## CONTENTS

4.2.2.4 Security Analysis .............................................. 35
4.3 Conclusion .................................................................. 36

5 Remote Systems Authentication ........................................ 38
  5.1 Password-based Authentication in Remote Systems .............. 38
    5.1.1 Related Works .................................................... 39
    5.1.2 Mathematical Background ...................................... 40
    5.1.3 The Scheme ........................................................ 41
    5.1.4 Analysis of the Scheme ......................................... 43
      5.1.4.1 Security ....................................................... 43
      5.1.4.2 Efficiency .................................................... 44
    5.1.5 Conclusion ......................................................... 45
  5.2 Pairing-based Authentication in Remote Systems ............... 45
    5.2.1 Authentication in Mobile Communications .................... 45
      5.2.1.1 Security properties ......................................... 46
      5.2.1.2 Related Works .............................................. 46
      5.2.1.3 The Proposed Scheme [Das and Saxena, 2005] .......... 49
      5.2.1.4 Analysis ....................................................... 50
    5.2.2 Pairing-based Remote User Authentication .................... 52
      5.2.2.1 The Scheme .................................................... 52
      5.2.2.2 Analysis ....................................................... 53
    5.2.3 Conclusion ......................................................... 55

6 Conclusion ................................................................... 56

A Ten years research on proxy signatures: A Survey ..................... 72
  A.1 Introduction ........................................................... 72
  A.2 Preliminaries ........................................................... 74
    A.2.1 Discrete Logarithm Problem .................................... 74
    A.2.2 Bilinear Pairings .................................................. 75
    A.2.3 Integer Factorization Problem ................................... 77
  A.3 Security Properties of Proxy Signature ............................. 78
    A.3.1 Classification of Proxy Signature .............................. 79
      A.3.1.1 Proxy-Unprotected Notion .................................. 79
      A.3.1.2 Proxy-Protected Notion ...................................... 80
      A.3.1.3 Threshold Notion ............................................. 80
  A.4 Models of Proxy Signature ........................................... 80
    A.4.1 DLP-based and RSA-based Proxy Signature .................. 80
    A.4.2 Pairing-based Proxy Signature Scheme ........................ 81
  A.5 Review of Some Proxy Signature Schemes .......................... 82
    A.5.1 DLP-Based Proxy Signature Schemes .......................... 82
      A.5.1.1 Mambo, Usuda and Okamoto [Mambo et al., 1996] ...... 82
      A.5.1.2 Kim, Park and Won [Kim et al., 1997] .................... 83
      A.5.1.3 Zhang [Zhang, 1997a] ....................................... 85
List of Tables

3.1 Performance of pairing-based proxy signature schemes . . . . . . . . . . . 24
A.1 Computation Time : DLP-based Proxy Signatures . . . . . . . . . . . . 102
A.2 Computation Time : RSA-based Proxy Signatures . . . . . . . . . . . . 103
A.3 Computation Time : Pairing-based Proxy Signatures . . . . . . . . . . . 104
Chapter 1

Introduction

Authentication is a process through which one proves and verifies certain information. Sometimes one may want to verify the origin of a document, the identity of the sender, the time and date a document was sent and/or signed, and so on. The process can be divided into two categories, namely Data authentication and Entity authentication. Data authentication provides the genuineness of data as well as legitimacy of the sender. Whereas, entity authentication assures the genuineness of a person or a server. The most widely used data authentication techniques are digital signature, proxy signature, multi-signature, etc. The entity authentication techniques are surrounded by password authentication and biometric authentication.

Digital signature is a cryptographic means through which the authenticity, data integrity and sender’s non-repudiation can be verified. Typically, digital signature of a document is a piece of information encrypted by the signer’s secret key. Numerous researches have shown significant contributions to this field using various cryptographic primitives [Koblitz, 1994], [Menezes et al., 1996]. However, there are many practical environments where digital signatures do not possess specific requirements. For example, a manager of a company wants to go for a long trip. He would need a proxy agent, to whom he would delegate his signing capability, and thereafter the proxy agent would sign the documents on behalf of the manager. A proxy signature serves the purpose in such environments. It may so happen that the document requires multiple signatures. For instance, an approval in an organization needs signature from the head of the department, dean, director and chairman. More than one party’s signature is called multi-signature and it has enormous applications in electronic business.

Entity authentication verifies the identity of a communication party and validates his
request. The important factors of an entity authentication are “things you know: a password”, “things you have: a credit card” and “things you are: biometric”.

1.1 Data Authentication

Data authentication and user authentication are closely linked in the sense that it is often an individual that certifies the authenticity. In the case of a digital signature, there needs to be a way to associate an individual with his public key. The usual way of doing this is through the key certification authority, who verifies the identity of the individual through some independent means. In 1976, Diffie and Hellman [Diffie and Hellman, 1976] introduced a new direction in public key cryptosystem. Since then, many excellent schemes have been proposed using various cryptographic primitives. The most widely used schemes are RSA signature [Rivest et al., 1978], ElGamal signature [ElGamal, 1985], Schnorr signature [Schnorr, 1991], DSA [NIST, 2000], ECDSA [ANSI, 1999], etc. Though digital signature provides authentication; however, there are many real scenario where proxy agents or multiple signers need to provide data authentication, in turn, proxy signature and multi-signature have enormous applications in electronic communications.

1.1.1 Proxy Signature

Proxy signature is a digital signature, where an original signer delegates his signing capability to a proxy signer, and then the proxy signer signs messages on behalf of the original signer. In 1996, Mambo et al. [Mambo et al., 1996] introduced proxy signature scheme and classified proxy signatures on the basis of delegation, namely, full delegation, partial delegation and delegation by warrant. In full delegation, an original signer gives his secret key to a proxy signer and using it the proxy signer signs the document. The drawback of proxy signature with full delegation is that the absence of a distinguishability between original signer and proxy signer. In partial delegation, the original signer derives a proxy key from his secret key and hands it over to the proxy signer as a delegation of signing rights. In this case, the proxy signer can misuse the delegation of signing rights, because partial delegation does not restrict the proxy signer’s signing capability. The weakness of full and partial delegations are eliminated by partial delegation with warrant, where a warrant states the signers’
identity, delegation period and the qualification of the message on which the proxy signer can sign, etc.

1.1.2 Multi-signature

A multi-signature is a digital signature that allows multiple signers to generate a signature in a serial or parallel manner. In everyday life, many legal documents require signatures from more than one party, viz. contracts, decision making processes, petitions etc. In 1983, Itakura and Nakamura [Itakura and Nakamura, 1983] first introduced the notion of multi-signature. Since then, several schemes and improvements for multi signatures [Boyd, 1989], [Harn, 1994], [Camenisch and Stadler, 1997], [Horster et al., 1995], [Ohta and Okamoto, 1999], [Mitomi and Miyaji, 2001] have been proposed; however, the schemes lack with formal security model. A formal notion of security for multi-signature is proposed by Micali et al. [Micali et al., 2001]. Afterwards, research on the security model in multi-signature has been a crucial criterion for any newly proposed scheme.

1.2 Entity Authentication

Entity authentication is the process by which an agent in a system gains confidence about the identity of a communicating partner. More precisely, an entity authentication process is coupled with the distribution of a “session key” which the partners can later use for message confidentiality, integrity, or whatever else. There are basically three important factors for an entity authentication, namely “something you know : a password”, “something you have : a token” and “something you are : biometric”.

Something you know : a password

Passwords are simply secrets that are provided by the user upon request by a recipient. Passwords are generally stored on a server in an encrypted form so that a penetration of the file system does not reveal password lists. However, possible threats have been evolved on its way to guess the password that people are most likely use. People are advised to use strong passwords, i.e., a random bit strings having sufficient size complying the desirable security policy.

Something you have : a token

Token means a mag-stripe card, a smart card, or a password calculator. Token-based authentication is the modest technique, since it relies on a physical object that one
must have in order to log on the system. Unlike passwords, the owner can inform to the appropriate authority if the token has been stolen, and it’s hard for the owner to share the token with others and still be able to log on. The major weaknesses are higher costs and the risk of loss or hardware failure.

Something you are : biometric

Biometrics use personal features instead of facts to authenticate a person. In most cases, a well-designed biometric system simply accepts a reading from the person and correctly perform the authentication. The distinguishing characteristic is obviously portable, since it’s part of the owner’s body. Unlike passwords, the biometric data rarely matches perfectly. Instead, the authentication mechanism measures how close the latest reading matches the reading of a user record. However, the benefits are offset by several weaknesses, viz. the equipment is expensive to buy, install, and operate in comparison to other systems.

1.3 Elements of an Authentication System

Regardless of whether an authentication system is computer based or not, there are several elements usually present, and certain things usually take place. First of all, we require a particular person or group of people to be authenticated. Second, we need a distinguishing characteristic that differentiates particular person or group from others. Third, there is a proprietor who is responsible for the system being used. Fourth, we need an authentication mechanism to verify the distinguishing characteristic. Fifth, we grant some privileges when the authentication succeeds by using an access control mechanism, and the same mechanism denies the privileges if authentication fails.

An appropriate example is the password-based login operation. The person of interest is an individual allowed to use the computer. The system usually assigns the person a symbolic name or user identification code which we will call the login identity. The distinguishing characteristic for the user is his secret password. The process contains an authentication procedure that compares the typed-in password against the password established for the user; the procedure succeeds if the two match. The access control mechanism allows the user to access the system resources, and the system uses user name whenever it makes access control decisions on protected resources.
1.4 Organization of the Report

In the next Chapter we give the technical preliminaries. We discuss the different cryptographic security assumptions, some standard signature models and a few computational problems, which are strongly related to the forthcoming Chapters.

Chapter 3 discusses our proposed proxy signature scheme. We present a bilinear pairing based proxy signature scheme that avoids key escrow and eliminates secure channel requirements in the key issuance stage, which remain constraints in the existing bilinear pairing based schemes. As we have discussed the proposed RSA based proxy signature scheme in my previous progress report, we include that work in Appendix B. Further, we surveyed the various proxy signature schemes right from its introduction, and an extensive survey report is given in Appendix A.

Chapter 4 discusses about multi signatures. Firstly, we present a multi-signature scheme based on RSA that uses a re-blocking method for different modulus sizes. Secondly, we present a secure key issuing protocol in identity-based cryptosystem then extend the scope to multi-signature. Our scheme avoids the key escrow problem and eliminates secure channel requirement in the key issuance stage.

Chapter 5 discusses the remote systems authentication techniques. We present three schemes for remote systems authentication. In the first and second schemes, a user will have a registered smart card and by using that card the customer can authenticate himself as well as confirm that the communicating server is genuine. In the third scheme, the mobile device securely transacts with the merchants through service provider. The preliminary approach which was presented in my previous progress report, is also included in Appendix C.

Finally, we conclude the report with Chapter 6.
Chapter 2

Preliminaries

Desirable cryptographic schemes have evolved to meet the security requirements in practical situation. The underlying security assumptions primarily rely on either discrete logarithm problem or integer factorization problem [Menezes et al., 1996]. This section discusses the mathematical background of these problems and some computational problems, which directly or indirectly links to the various schemes throughout this report.

2.1 Discrete Logarithm Problem

The discrete logarithm is the inverse of discrete exponentiation in a finite cyclic group. Given a cyclic group \( G \) of order a large prime \( p \) with group operation \( x \) and a generator \( g \), exponentiation in \( G \) is defined by \( g^x = g \times g \times \ldots \times g \) \((x\text{-times})\). Suppose \( y = g^x \), then the discrete logarithm of \( y \) is \( x \) and is written as \( \log_g y = x \mod p \).

In 1976, Diffie and Hellman [Diffie and Hellman, 1976] described a key exchange protocol, where the underlying protocol was based discrete logarithm problem (DLP). Since then, several key exchange [Boyd and Mathuria, 2003], public key encryption [Canetti et al., 2003], [Menezes et al., 1996], and digital signatures [ElGamal, 1985], [Goldwasser et al., 1988], [Schnorr, 1991] have been proposed in which security assumptions rely on the hardness of the DLP. As many of the forthcoming schemes are based on Schnorr’s signature scheme, we briefly discuss the scheme as follows.

The Schnorr Signature Scheme[Schnorr, 1991] : The scheme has the following phases:

System Parameters \((SP_{DLP})\), KeyGen \((KG_{DLP})\), Sign \((S_{DLP})\) and Verify \((V_{DLP})\).

System Parameters \((SP_{DLP})\) : Inputs \(1^k\); and outputs \(params_{DLP}\). The \(params_{DLP}\) consists of primes \(p, q\) such that \(2^{k-1} \leq p < 2^k\) and \(q\) divides \(p - 1\), an element \(g \in \mathbb{Z}_p^*\) of or-
der $q$, and a hash function $h : \{0, 1\}^* \rightarrow \mathbb{Z}_q$. In other words, $\text{paramsDLP} \leftarrow \text{SP}_{DLP}(1^k)$.

**KeyGen ($KG_{DLP}$)**: The users agree on a group $G$ (multiplicative group of integers modulo $p$ for a prime $p$ (at least 1024 bits) with generator $g$ of prime order $q$ (160 bits) in which the discrete logarithm problem is hard. The user chooses a secret key $x$ such that $0 < x < q$. The public key is $y$ where $y = g^x \mod p$.

In other words, user public key $\leftarrow KG(paramsDLP, user parameter(s), user secret key)$.

**Sign ($S_{DLP}$)**: To sign a message $m$,

Choose a random $k$ such that $0 < k < q$ and compute $r = g^k \mod p$. Compute $e = H(m || r)$ and $\sigma = (k - xe) \mod q$. The signature is the pair $(e, \sigma)$.

In other words, $\sigma \leftarrow S_{DLP}(paramsDLP, user parameter(s), x, m)$.

**Verify ($V_{DLP}$)**: Let $r' = g^\sigma y^e \mod p$. Compute $e' = H(m || r')$. If $e' = e$ then the signature is valid.

In other words, $\text{Result} \leftarrow V_{DLP}(paramsDLP, y, m, \sigma)$, where $\text{Result} \in \{Accepts, Rejects\}$.

This signature scheme has been proven to be secure under existential forgery attacks [Pointcheval and Stern, 1996].

### 2.1.1 Bilinear Pairings

Pairings were first introduced to elliptic curve cryptography for destructive methods like the MOV reduction [Menezes et al., 1993]. With the help of the Weil pairing, the authors of [Menezes et al., 1993] showed a way to reduce the discrete logarithm problem on supersingular elliptic curves to the discrete logarithm problem of an extension of the underlying finite field. Later Frey and Ruck [Frey and Ruck, 1994] extended this attack to more general elliptic curves with the Tate pairing. However, the Weil pairing and Tate pairing can also be used as a constructive tool in cryptosystem [Boneh and Franklin, 2001], [Boneh et al., 2001], [Cocks, 2001], [Hess, 2002]. In the following, the properties of pairings are listed:

Suppose $G_1$ is an additive cyclic group generated by $P$, whose order is a prime $q$, and $G_2$ is a multiplicative cyclic group of the same order. A map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is called a bilinear mapping if it satisfies the following properties:

- Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} \forall P, Q \in G_1$ and $a, b \in \mathbb{Z}_q^*$;
- Non-degenerate: There exist $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$;
- Computable: There is an efficient algorithm to compute $\hat{e}(P, Q) \forall P, Q \in G_1$.

In general, $G_1$ is a group of points on an elliptic curve and $G_2$ is a multiplicative subgroup of a finite field.
CHAPTER 2. PRELIMINARIES

Computational Problems

Definition 1. Discrete Logarithm Problem (DLP): Given $Q, R \in G_1$, find an integer $x \in \mathbb{Z}_q^*$ such that $R = xQ$.

The MOV and FR reductions: Menezes et al. [Menezes et al., 1993] and Frey and Ruck [Frey and Ruck, 1994] show a reduction from the DLP in $G_1$ to the DLP in $G_2$. The reduction is: Given an instance $Q, R \in G_1$, where $Q$ is a point of order $q$, find $x \in \mathbb{Z}_q^*$ such that $R = xQ$.

Let $T$ be an element of $G_1$ such that $g = \hat{e}(T, Q)$ has order $q$, and let $h = \hat{e}(T, R)$. Using bilinear property of $\hat{e}$, we have $\hat{e}(T, R) = \hat{e}(T, Q)^x$. Thus, DLP in $G_1$ is no harder than the DLP in $G_2$.

Definition 2. Computational Diffie-Hellman Problem (CDHP): Given $(P, aP, bP)$ for $a, b \in \mathbb{Z}_q^*$, compute $abP$.

The advantage of any probabilistic polynomial-time algorithm $A$ in solving CDHP in $G_1$, is defined as $\text{Adv}_{A,G_1}^{\text{CDH}} = \text{Prob}[A(P, aP, bP, abP) = 1 : a, b \in \mathbb{Z}_q^*]$. For every probabilistic algorithm $A$, $\text{Adv}_{A,G_1}^{\text{CDH}}$ is negligible.

Definition 3. Decisional Diffie-Hellman Problem (DDHP): Given $(P, aP, bP, cP)$ for $a, b, c \in \mathbb{Z}_q^*$, determine whether $c \equiv ab \mod q$. This is same as MOV and FR reductions. The advantage of any probabilistic polynomial-time algorithm $A$ in solving DDHP in $G_1$ is defined as $\text{Adv}_{A,G_1}^{\text{DDH}} = [\text{Prob}[A(P, aP, bP, cP) = 1] - \text{Prob}[A(P, aP, bP, abP) = 1] : a, b \in \mathbb{Z}_q^*]$. For every probabilistic polynomial-time algorithm $A$, $\text{Adv}_{A,G_1}^{\text{DDH}}$ is negligible.

Definition 4. Gap Diffie-Hellman (GDH) group: A prime order group $G_1$ is a GDH group if there exists an efficient polynomial-time algorithm which solves the DDHP in $G_1$ and there is no probabilistic polynomial-time algorithm which solves the CDHP with non-negligible probability of success. The domains of bilinear pairings provide examples of GDH groups. The MOV reduction provides a method to solve DDH in $G_1$, whereas there is no known efficient algorithm for CDH in $G_1$.

Definition 5. Bilinear Diffie-Hellman Problem (BDHP): Given $(P, aP, bP, cP)$ for $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc}$.

Definition 6. Weak Diffie-Hellman Problem (WDHP): Given $(P, Q, aP)$ for $a \in \mathbb{Z}_q^*$, compute $aQ$.

A pairing-based signature scheme is as follows:

Hess’s Signature Scheme [Hess, 2002]: The signature scheme has the following phases:

System Parameters $(SP_{CDHP})$, Extract $(E_{CDHP})$, Sign $(S_{CDHP})$ and Verify $(V_{CDHP})$.

System Parameters $(SP_{CDHP})$: $\text{paramsCDHP} \leftarrow SP_{CDHP}(1^k)$
CHAPTER 2. PRELIMINARIES

It takes $1^k$ and master-key $s$ as inputs; outputs paramsCDHP. The paramsCDHP includes groups $G_1, G_2$ of order prime $p$; a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$; hash functions, and public key of PKG ($Q_{PKG} = sP$). The PKG keeps $s$ secret.

Extract ($E_{CDHP}$): It takes paramsCDHP, user public key $Q = H(ID)$ as inputs; outputs user secret key $S_{ID} = sQ$. In other words, $S_{ID} \leftarrow E_{CDHP}(\text{paramsCDHP}, Q_{ID})$.

Sign ($S_{CDHP}$): To sign a message $m$, the signer chooses an arbitrary $P_1 \in G_1$, picks a random $k \in \mathbb{Z}_q^*$ and computes

- $r = \hat{e}(P_1, P)^k$.
- $c = h(m, r)$.
- $\sigma = cS_{ID} + kP_1$.

The signature on $m$ is the tuple $(m, c, \sigma)$.

In other words, $\sigma \leftarrow S_{CDHP}(\text{paramsCDHP}, r, S_{ID}, m)$

Verify ($V_{CDHP}$): The signature $(m, c, \sigma)$ is verified by the following:

- Compute $r' = \hat{e}(\sigma, P) \cdot \hat{e}(H(ID), -Q_{PKG})^c$.
- Accept the signature if and only if $c = h(m, r)$.

In other words, Result $\leftarrow V_{CDHP}(\text{paramsCDHP}, Q_{ID}, m, \sigma)$

The Hess’s signature scheme is proven secure against existential forgery on adaptive chosen-message attack under the assumption that CDHP is hard [Hess, 2002].

2.2 Integer Factorization Problem

The integer factorization (also known as prime decomposition) problem is: Input a positive integer; output it as a product of prime numbers. The problem holds a strong security assumption in cryptography, complexity theory, and quantum computers. The following is the most widely used signature scheme based on integer factorization problem:

RSA signature Scheme [Rivest et al., 1978]: The RSA (Rivest, Shamir and Adleman) signature scheme is one of the first great advances in public key encryption and signature schemes. The signature scheme has following phases: System Parameters ($SP_{RSA}$), KeyGen ($KG_{RSA}$), Sign ($S_{RSA}$) and Verify ($V_{RSA}$). The algorithms are as follows:

System Parameters ($SP_{RSA}$): Inputs $1^k$, primes $p, q$; and outputs paramsRSA. The paramsRSA consists of a modulus $N = pq$ and a hash function $h : \{0, 1\}^* \rightarrow \mathbb{Z}_N$. In other words, paramsRSA $\leftarrow SP_{RSA}(1^k)$.

KeyGen ($KG_{RSA}$): Compute $\phi(N) = (p - 1)(q - 1)$. Choose an integer $1 < e < \phi(N)$...
which is co-prime to $\phi(N)$. Compute $d$ such that $de \equiv 1 \mod \phi(N)$. The public key consists of the modulus $N$ and the public exponent $e$. The secret key consists of the modulus $N$ and the secret exponent $d$, which must be kept secret. In other words, user secret key $\leftarrow KG_{RSA}(paramsRSA, \text{user public key})$.

**Sign ($S_{RSA}$):** To sign a message $m$,

Compute $\sigma = H(m)^d \mod N$. The signature of a message $m$ is the tuple $(\sigma, m)$.

In other words, $\sigma \leftarrow S_{RSA}(paramsRSA, d, m)$.

**Verify ($V_{RSA}$):** Compute $m' = \sigma^e \mod N$. The signature is valid if $m' = m$.

In other way, $\text{Result} \leftarrow V_{RSA}(paramsRSA, e, m, \sigma)$.

This signature scheme has been proven to be secure under the assumption of integer factorization problem is hard [Boneh, 1999].
Chapter 3

Proxy Signature

3.1 Introduction

Proxy signature is a digital signature where an original signer delegates his signing capability to a designated proxy signer, and then the proxy signer performs message signing on behalf of the original signer. For example, a manager of a company wants to go for a long trip. He would need a proxy agent, to whom he would delegate his signing capability, and thereafter the proxy agent would sign the documents on behalf of the manager. The delegation capability can be formed in three types, namely full delegation, partial delegation and delegation by warrant. In full delegation, an original signer gives his secret key to a proxy signer and using it the proxy signer signs document. The drawback of proxy signature with full delegation is that the absence of a distinguishability between the original signer and proxy signer. In partial delegation, the original signer derives a proxy key from his secret key and hands it over to the proxy signer as a delegation of signing rights. In this case, the proxy signer can misuse the delegation of signing rights, because partial delegation does not restrict the proxy signer’s signing capability. The weakness of full and partial delegations are eliminated by partial delegation with warrant, where a warrant explicitly states the signers’ identity, delegation period and the qualification of the message on which the proxy signer can sign, etc. Once proxy delegation power is given, the proxy revocation is also an important concern when the original signer secret key is compromised and/or any misuse of delegation of signing rights is noticed. It may so happen that the original signer wants to terminate the delegation of signing rights before the expiry (for instance, the manager of a company returns back from his trip before time that he was scheduled for).
In 1996, Mambo et al. [Mambo et al., 1996] first introduced the concept of proxy signature. Since then, numerous researches have shown an impressive list of proxy signature schemes using various cryptographic primitives [Koblitz, 1994], [Menezes et al., 1996]. We have studied an extensive research literature on proxy signatures and our survey report is included in Appendix A.

### 3.2 Security Properties

Desirable security properties of proxy signatures have evolved over this period. A widely accepted list of required properties at this juncture is given below.

- **Strong unforgeability**: A designated proxy signer can create a valid proxy signature on behalf of the original signer. But the original signer and other third parties cannot create a valid proxy signature.

- **Strong identifiability**: Anyone can determine the identity of corresponding proxy signer from the proxy signature.

- **Strong undeniability**: Once a proxy signer creates a valid proxy signature on behalf of the original signer, he cannot deny the signature creation.

- **Verifiability**: The verifier can be convinced of the signers’ agreement from the proxy signature.

- **Distinguishability**: Proxy signatures are distinguishable from the normal signatures by everyone.

- **Secrecy**: The original signer secret key cannot be derived from any information, such as the shares of the proxy key, proxy signatures, etc.

- **Prevention of misuse**: The proxy signer cannot use the proxy key for other purposes than it is made for. That is, he cannot sign message with the proxy key that have not been defined in the warrant. If he does so, he will be identified explicitly from the warrant.

According to the delegation capability, proxy signature can also be classified as proxy-unprotected and proxy-protected notions [Kim et al., 1997]. This differentiation is important for the practical applications, since it enables proxy signature schemes to avoid potential disputes between original signer and proxy signer.
3.3 The Proposed Schemes

3.3.1 Proxy Signature Scheme based on RSA

In 1999, Okamoto et al. [Okamoto et al., 1999] proposed a proxy signature scheme based on RSA. Later, Lee et al. [Lee et al., 2001b] showed that a secure mobile agent can be constructed using strong non-designated proxy signature. However, both the schemes [Okamoto et al., 1999] and [Lee et al., 2001b] are designed as proxy-unprotected notion, in turn, these schemes do not meet the strong unforgeability property and suffer from delegation capability misuse. In 2003, Shao [Shao, 2003] proposed another proxy signature scheme based on factoring. The schemes [Okamoto et al., 1999] and [Shao, 2003] require secure channel to deliver the proxy key. Further, these schemes do not provide any proxy revocation mechanism, thereby the original signer cannot revoke his delegated signing rights if he wants to do so, which is a practical requirement.

Considering the weaknesses in [Okamoto et al., 1999], [Lee et al., 2001b], [Shao, 2004], we proposed a proxy signature scheme [Das et al., 2004b] that provides effective proxy revocation. Our scheme does not require secure channel for delegation of signing capability delivery. We note that our scheme was presented in the last progress seminar and the same is included in Appendix B.

3.3.2 Proxy Signature Scheme using Bilinear Pairings

Bilinear pairings, namely the Weil pairing and the Tate pairing of algebraic curves have been found important applications in cryptography and allowed to construct identity(ID)\(^1\) [Shamir, 1984] based cryptographic scheme. In ID-based scheme, the public key is computed from the user ID. Thereby, ID-based cryptographic scheme has an advantage over the traditional public key based schemes [Rivest et al., 1978], [NIST, 2000] that the public key is linked with an entity of a user, in turn, avoids the public key certification burden. The practical ID-based schemes were found in 2001 by Cocks [Cocks, 2001] and by Boneh and Franklin [Boneh and Franklin, 2001], [Hess, 2002].

With the advancement of pairing-based cryptosystem, many proxy signature schemes [Chen et al., 2003a], [Zhang and Kim, 2003], [Xu et al., 2004], [Zhang et al., 2004] have

\(^1\)An identity (ID) is the unique identifier of a user, such as email-id, social security number, permanent account number, etc.
been proposed; however, proxy revocation mechanism has not been addressed in these schemes. Moreover, we observe that these schemes lack the inherent key escrow problem and require secure channel in the key issuance stage. The revocation of delegation of signing rights (i.e., proxy revocation) is an important concern in any proxy signature scheme. One may argue that the keys in pairing-based cryptography are short-term keys and revocation of these keys could be avoided. But, there are many situations where revocation of a compromised key or malpractices of proxy key requires immediate attention.

In this section, we propose a proxy signature signature scheme using bilinear pairings that provides effective proxy revocation. We use a binding-blinding technique to eliminate the secure channel requirements in the key issuance stage. The proposed scheme can avoid the key escrow problem and can withstand the necessary security properties.

3.3.2.1 The Model

The proposed model consists of

- a trusted party, called Private Key Generator (PKG), who issues secret key to the signer.

- an original signer, who gives delegation of signing rights to a proxy signer.

- a proxy signer, who signs the message on behalf of the original signer.

- a verifier, who verifies the proxy signature and decides to accept or reject.

- five phases: Setup, KeyGen, Proxy Key Generation, Proxy Signature Generation and Proxy Signature Verification.

The phases work as follows:

Setup : It takes as input a security parameter; and outputs system parameters $\text{paramsCDHP}$ and master-key of PKG. The $\text{paramsCDHP}$ includes $G_1$ (a cyclic additive group of prime order $q$) generated by $P$; $G_2$ (a cyclic multiplicative group of prime order $q$); a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$; hash functions $H_1 : \{0,1\}^* \times G_1 \times G_1 \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow G_1$, $h : \{0,1\}^* \times G_1 \times G_1 \rightarrow \mathbb{Z}_q^*$; and public key of PKG. The $\text{paramsCDHP}$ is made public and the master-key is kept secret.

KeyGen : takes signer’s chosen parameters and $\text{paramsCDHP}$ as inputs; outputs signer’s secret key. The phase is composed of a partial secret key issuance stage and
a secret key generation stage. The whole process involves a binding-blinding technique to avoid the key escrow problem and to eliminate the secure channel requirements. The binding-blinding technique is as follows:

- The signer chooses two secret blinding factors, calculates the binding parameters and sends them to the PKG in a blinded manner along with his identity.
- The PKG verifies signer’s identity and checks the validity of binding parameters. Upon successful validation, the PKG creates signer’s partial secret key. Then, the PKG sends partial secret key to the signer in a blinding manner over the public channel.

Partial secret key issuance: It receives signer’s chosen parameters for the issuance of partial secret key. Upon validating the signer’s identity, the PKG sends partial secret key to the signer in a blinding manner and publishes \(<\ registration\text{-}token,\ identity>\) pair in a public directory. We note that the PKG controls this public directory and checks it for new request for partial secret key issuance, and thereby, the registered-token replacement is not possible by the dishonest signer or adversary.

Secret key generation: Upon receiving the partial secret key, the signer unblinds it and computes secret key.

Proxy Key Generation: takes the original signer’s delegation of signing rights and proxy signer’s secret key as the inputs; and outputs the proxy key.

Proxy Signature Generation: inputs \(paramsCDHP\), proxy key and message \(m\); outputs the proxy signature of \(m\).

Proxy Signature Verification: takes as input the \(paramsCDHP\), public key of the signers, message \(m\) and signature of the message; outputs valid or invalid.

3.3.2.2 The Scheme

To avoid the original signer’s forgery on the proxy signature, the proxy-protected notion [Kim et al., 1997] is a secure approach. Our scheme is based on proxy-protected notion and uses the merits of partial delegation with warrant.

The participating entities and their roles in the scheme are defined as follows:
- Private Key Generator (PKG): A trusted authority who receives signer’s identity (ID) containing other parameters, checks validity of ID and issues partial secret key to the signer.
- Original Signer: An entity who gives delegation of signing rights to a proxy signer.
- **Proxy Signer:** An entity who signs the message on behalf of the original signer.
- **Verifier:** An entity who verifies the proxy signature and decides to accept or reject.

The scheme works as follows:

[ **Setup** ]

The setup parameters remain same as explained in the model.

[ **KeyGen** ]

Let $ID_\lambda$ be the ID of a user $U_\lambda$. The user gets his partial secret key from the PKG by the following steps:

- $U_\lambda$ computes his public key as $Pub_\lambda = H_2(ID_\lambda)$.

- $U_\lambda$ picks two secret blinding factors $a_\lambda, b_\lambda \in \mathbb{Z}_q^*$, computes $X_\lambda = a_\lambda Pub_\lambda$, $Y_\lambda = a_\lambda b_\lambda Pub_\lambda$, $Z_\lambda = b_\lambda P$ and $W_\lambda = a_\lambda b_\lambda P$, then sends $(X_\lambda, Y_\lambda, Z_\lambda, W_\lambda, ID_\lambda)$ to the PKG over a public channel.

- The PKG computes $Pub_\lambda = H_2(ID_\lambda)$ and verifies the validity of $ID_\lambda$ by whether $\hat{e}(Y_\lambda, P) = \hat{e}(X_\lambda, Z_\lambda) = \hat{e}(Pub_\lambda, W_\lambda)$.

- Once the $ID_\lambda$ is valid, the PKG computes signer’s partial secret key as $D_\lambda = sY_\lambda$ and computes a registration-token $Reg_\lambda = sZ_\lambda$. Then, PKG publishes $(Reg_\lambda, ID_\lambda)$ in a public directory and sends $D_\lambda$ to the signer over a public channel.

- On receiving $D_\lambda$, the signer checks whether $\hat{e}(D_\lambda, P) = \hat{e}(Y_\lambda, Pub_{PKG})$. If it is valid, the signer unblinds his secret key as $S_\lambda = a_\lambda^{-1} D_\lambda$.

**Original signer secret key:** Let $ID_o$ be the identity of an original signer. The original signer chooses blinding factors $a_o$ and $b_o$, runs the **KeyGen** phase and gets partial secret key $D_o$. After validating the partial secret key $D_o$, the original signer generates his secret key as $S_o = a_o^{-1} D_o$.

**Proxy signer secret key:** Let $ID_p$ be the identity of the proxy signer. The proxy signer chooses the blinding factors $a_p$ and $b_p$, runs the **KeyGen** phase and gets partial secret key $D_p$. After validating the partial secret key $D_p$, the proxy signer generate his secret key as $S_p = a_p^{-1} D_p$.

[ **Proxy Key Generation** ]

Both the original signer and proxy signer agree on a warrant $m_w$. The proxy key is created by the following steps:

- The original signer computes $U_o = S_o + b_o H_1(m_w, Pub_o, Pub_p)$ and $\psi_o = b_o P$. Then, he sends the tuple $(m_w, U_o, \psi_o, Pub_o)$ to the proxy signer as his delegation of signing.
CHAPTER 3. PROXY SIGNATURE

rights over a public channel.
- The proxy signer checks whether $\hat{e}(U_o, P) = \hat{e}(\psi_o, H_i(m_w, Pub_o, Pub_p)) \cdot \hat{e}(Pub_o, Reg_o)$.
If delegation of signing rights is valid, the proxy signer computes proxy key as $V_p = U_o + S_p + b_p H_i(m_w, Pub_o, Pub_p)$.

[ Proxy Signature Generation ]
The proxy signer does the following operations to generate a proxy signature on message $m$:
- Select a random $r \in Z_q^*$ and compute $R = rP$
- Compute $a = h(m, R, Pub_p)$ and $\psi_p = b_p P$
- Compute $V = (r + a)^{-1}V_p$.

The proxy signature of $m$ is the tuple $(m_w, m, R, V, \psi_o, \psi_p, Pub_o, Pub_p)$.

[ Proxy Signature Verification ]
The proxy signature $(m_w, m, R, \psi_o, \psi_p, Pub_o, Pub_p)$ is valid if and only if
$\hat{e}(R + h(m, R, Pub_p)P, V) = \hat{e}(\psi_o + \psi_p, H_i(m_w, Pub_o, Pub_p)) \cdot \hat{e}(Pub_o, Reg_o) \cdot \hat{e}(Pub_p, Reg_p)$.

3.3.2.3 Analysis of the Scheme

Correctness: $\hat{e}(R + h(m, R, Pub_p)P, V)$
$= \hat{e}((r + h(m, R, Pub_p))P, (r + a)^{-1}V_p)$
$= \hat{e}((r + a)P, (r + a)^{-1}V_p)$
$= \hat{e}(P, U_o + S_p + b_p H_i(m_w, Pub_o, Pub_p))$
$= \hat{e}(P, S_p + S_o + (b o + b_p) H_i(m_w, Pub_o, Pub_p))$
$= \hat{e}(P, S_p) \cdot \hat{e}(P, S_o) \cdot \hat{e}(P, (b o + b_p) H_i(m_w, Pub_o, Pub_p))$
$= \hat{e}(Pub_o, Reg_o) \cdot \hat{e}(Pub_p, Reg_p) \cdot \hat{e}((b o + b_p) P, H_i(m_w, Pub_o, Pub_p))$
$= \hat{e}(Pub_o, Reg_o) \cdot \hat{e}(Pub_p, Reg_p) \cdot \hat{e}(\psi_o + \psi_p, H_i(m_w, Pub_o, Pub_p))$.

Security Analysis: The most important notion of security of a signature scheme is the forgery, namely Universal forgery, Selective forgery and Existential forgery.
- Universal forgery: Finding an efficient algorithm that will form a valid signature on a message without the knowledge of the signer’s secret key.
- Selective forgery: Forge a signature for a particular chosen message, with or without interacting with the signature.
- Existential forgery: Forge signature for at least one message without the knowledge of the secret key. The adversary has no control over the message whose signature he
obtains, so it may be randomly chosen.
We show that our scheme is secure against existential forgery, in turn proves that the
scheme is secure against selective and universal forgeries.
In our scheme, the master-key \( s \) is known to the PKG and the secret key \( S_i \) (cor-
responds to \( ID \)) is known to signer only. The partial secret key \( D_i \) is known to both PKG
and signer. With respect to the above forgery definition, the possible forgery, called
existential forgery on adaptive chosen-message attacks in our scheme is analyzed by
the following game. The game comprises with a challenger \( C \) and an adversary \( A \).
The challenger \( C \) takes a security parameter \( k \) and runs the \texttt{Setup} phase. It returns
the resulting system parameters \texttt{paramsCDHP} to \( A \) and keeps master-key \( s \) with itself.
Queries : \( A \) issues adaptively the queries \( q_1, q_2, \ldots, q_m \) for the followings (in any or-
der):
\texttt{KeyGen} query on \( ID_i \) : \( C \) runs the \texttt{KeyGen} phase and generates secret key \( S_i \) corre-
sponding to \( ID_i \) and sends it to \( A \).
\texttt{Sign} query on \( (ID_i, M^*) \) : \( C \) runs the \texttt{KeyGen} phase to generate the secret key \( S_i \) corre-
sponding to \( ID_i \). Then, \( C \) signs the message \( M^* \) and returns the message-signature
pair \( (M^*, \sigma(M^*)) \) to \( A \).
Guess : \( A \) outputs a message-signature pair \( (M, \sigma(M)) \), where \( M \) is the message that
did not appear in the \texttt{signing query}.
Result : \( A \) wins if \( \sigma(M) \) is a valid signature on \( M \). The advantage of \( A \) against a
signature scheme is defined to be the probability that \( A \) produces a valid message-
signature pair in the game.
We say that our scheme is secure against existential forgery on adaptive chosen-
message attacks if no polynomially bounded adversary has non-negligible advantage
in this game.
\textbf{Theorem 1}. A signature scheme is said to be secure against existential forgery on adaptive
chosen-message attacks if no polynomially bounded adversary (in \( k \)) has non-negligible ad-
\texttt{vantage} (in \( k \)).
Apart from the security against existential forgery, our scheme can also withstand the
following possible attacks:
\textbf{Registration-token replacement} : The PKG creates registration-token \( (Reg_i) \) corre-
sponding to each registered user and publishes it along with user ID in a public di-
rectory, which is controlled by PKG only. If any request comes from user identity \( ID^* \)
for issuance of a partial secret key, the PKG first checks whether \( ID^* \) is in the public
directory. If it is found in the public directory, the PKG rejects the request, otherwise
executes the $\text{KeyGen}$ algorithm for $ID^*$. Thus, $\text{registration-token}$ replacement is not possible by any party (the PKG itself can replace the same, but we assume that the signer trusts PKG for not to do it).

Man-in-the-Middle attacks: As the communication channel of the key issuance phase in our scheme is a public channel, an attacker may try to calculate the secret key or blinding factors of a user by intercepting the binding parameters and partial secret key. Now the problem can be defined as: Given $\text{paramsCDHP}$, binding parameters $(a_i\text{Pub}_i, a_i b_i \text{Pub}_i, b_i P, a_i b_i P, ID_i)$ and partial secret key $D_i$ (i.e., $s a_i b_i \text{Pub}_i$); Compute secret key $S_i$ (i.e. $s b_i \text{Pub}_i$) or blinding factors $(a_i, b_i)$. The solutions of this problem are equivalent to solve the CDHP and WDHP, which are assumed to be computationally hard.

One partial secret key $\rightarrow$ Many secret keys: The scenario of generating more than one secret key from a partial secret key is not possible, as the secret key $S_i$ and the $\text{Reg}_i$ are linked by a secret blinding factor $b_i$. If a user generates another secret $S'_i$ from $S_i$ and signs a message by $S'_i$, then the verification of the signature fails because the change from $S_i$ to $S'_i$ is not reflected in $\text{Reg}_i$. Thereby, the signer cannot perform this type of attempt without being detected.

Strong Unforgeability: The proxy signature is created on using the original signer and proxy signer’s secret keys. The original signer cannot generate a valid proxy signature as he does not know proxy signer secret key. Any third party cannot create a valid proxy signature without having the knowledge of proxy signer’s secret key. Further, the PKG cannot frame neither the original signer nor proxy signer as extracting the blinding factors $a_i, b_i$ from the signer parameters is also assumed as hard as WDHP.

Identifiability, Undeniability and Distinguishability: The proxy signature on $m$ is the tuple $(m_w, m, R, V, \psi_o, \psi_p, \text{Pub}_o, \text{Pub}_p)$. The signature verification requires signers’ public key as well as their registration tokens. The public keys $\text{Pub}_o, \text{Pub}_p$ and warrant $m_w$ are the straightforward witnesses (i.e., identities) of the signatories. In addition, a verifier will come to know the agreement between original and proxy signers for the proxy signature from $m_w$. From the correctness of the proxy signature, it is clear that the proxy signer cannot deny his signature creation. Moreover, for any dispute in the court of law, the PKG can prove the identity of the signers’ by $<\text{registration-token, ID}>$ tuple because $\text{Reg}_o = s \psi_o$ and $\text{Reg}_p = s \psi_p$.

Prevention of misuse: The original signer signs $(m_w, \text{Pub}_o, \text{Pub}_p)$ to create his dele-
CHAPTER 3. PROXY SIGNATURE

The delegation of signing rights and the proxy signer signs the message with this delegation. The restriction of message and limitation of proxy is clearly defined in $m_w$ and signed delegation is made for the designated proxy signer only. In case of the proxy signer’s misuse, the responsibility of proxy signer is determined explicitly from the warrant $m_w$. The original signer’s misuse is also prevented because he cannot create a valid proxy signature against the name of the proxy signer.

Efficiency: In order to analyze efficiency, we compare our scheme with the existing pairings-based proxy signatures [Chen et al., 2003a], [Xu et al., 2004], [Zhang et al., 2004] in Table 3.1. The notations used in Table 3.1 are as follows:

- $P_o$: Time taken for a pairing operation (20 ms) [Barreto and Kim, 2001]
- $H$: Time taken for a Map-to-Point (5.53 ms) [Bertoni et al., 2005]

<table>
<thead>
<tr>
<th>Phases $\rightarrow$ Schemes ↓</th>
<th>Proxy Key Generation</th>
<th>Proxy Sign. Generation</th>
<th>Proxy Sign. Verification</th>
<th>Secure Channel</th>
<th>Key Escrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Chen et al., 2003a] $^2$</td>
<td>$3P_o + 2H$ (71.06)</td>
<td>$P_o + H$ (25.53)</td>
<td>$2P_o + 2H$ (51.06)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[Xu et al., 2004]</td>
<td>$3P_o + 3H$ (76.59)</td>
<td>$H$ (5.53)</td>
<td>$5P_o + 4H$ (122.12)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[Zhang et al., 2004]</td>
<td>$2P_o + 3H$ (56.59)</td>
<td>$-$</td>
<td>$2P_o$ (40)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>$3P_o + 3H$ (76.59)</td>
<td>$-$</td>
<td>$4P_o + H$ (85.53)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.1: Performance of pairing-based proxy signature schemes

We note that the computation time for a valid proxy signature falls into two parts. The first part consists of the time taken for the Setup, KeyGen and Proxy key generation phases, which is a one-time computation and remains same for the entire delegation period. The second part consists of the computation time for Proxy signature generation and Proxy signature verification phases, and these are required as and when a signature is generated or verified. The computation cost for the Setup phase remains same for all the schemes. Our scheme requires more computational cost for the KeyGen phase as the secret key is computed through a binding-blinding technique. However, this is a one-time computation and does not increase the cost for the signature and verification phases. Additionally, using the binding-blinding technique, our scheme eliminates secure channel in the key issuance.

$^2$The scheme is designed for multi-proxy notion, we consider one-proxy computational cost as other schemes are designed as one-proxy notion.
CHAPTER 3. PROXY SIGNATURE

stage and avoids the key escrow problem, which are remain constraints in the existing schemes.

It is clear from the Table 3.1 that our scheme takes less computation time for proxy signature generation compared to [Chen et al., 2003a], [Xu et al., 2004], but takes more computation time for verification compared to [Chen et al., 2003a], [Zhang et al., 2004].

Proxy Revocation: In our scheme, proxy revocation can be easily done by revoking the registration-token from the PKG’s public directory. If the original signer wants to revoke his delegation of signing rights, he sends a revoke-request tuple \((M_r, m_w, Rev, Pub_o, Pub_p, \psi_o)\) to the PKG and proxy signer, where \(Rev = S_o + b_oH_1(M_r, Pub_o, Pub_p)\) and \(M_r\) states the identity of the signer along with the reason for the proxy revocation. The PKG first checks the authenticity and validity of the revoke-request and if the request is valid then PKG revokes the tuple \((Reg_o, ID_o)\) and \((Reg_p, ID_p)\) from the public directory. We note that the proxy signer will not object if the PKG removes \((Reg_p, ID_p)\) without his consent (the original consent is with PKG), because if the delegation capability is no longer authorized, the delegated proxy signer is no longer required. The PKG validates the revoke-request by the following checks:

\[
\hat{e}(Rev, P) = \hat{e}(S_o + b_oH_1(M_r, Pub_o, Pub_p), P) \\
= \hat{e}(sb_oPub_o + b_oH_1(M_r, Pub_o, Pub_p), P) \\
= \hat{e}(Reg_o, Pub_o) \cdot \hat{e}(H_1(M_r, Pub_o, Pub_p), \psi_o)
\]

We note that the schemes [Chen et al., 2003a], [Xu et al., 2004], [Zhang et al., 2004] do not provide proxy revocation.

3.4 Conclusion

We proposed a proxy signature scheme using bilinear pairings which provides effective proxy revocation. It is observed that the existing pairing-based proxy signature schemes suffer from the key escrow problem and require secure channel in the key issuance stage. We use a binding-blinding technique to avoid the key escrow problem and to eliminate the secure channel requirements in the key issuance stage. We have shown that the proposed scheme is secure against existential forgery on adaptive chosen-message attacks and other possible attacks.
Chapter 4

Multi Signatures

4.1 Introduction

A multi-signature is a digital signature that allows multiple signers to generate a signature. In 1983, Itakura and Nakamura [Itakura and Nakamura, 1983] first introduced the notion of multi-signature. Several multi-signature schemes [Boyd, 1989], [Harn, 1994], [Horster et al., 1995], [Okamoto, 1988], [Ohta and Okamoto, 1999], [Camenisch and Stadler, 1997], [Mitomi and Miyaji, 2001] have been proposed; however, the schemes lack formal security model. A formal notion of security for multi-signature is proposed by Micali et al. [Micali et al., 2001]. Boldyreva [Boldyreva, 2003] generalized the security model.

4.2 The Proposed Schemes

We present two multi-signature schemes. The first scheme is based on RSA and the second one is on bilinear pairings.

4.2.1 Multi-signature Scheme based on RSA

In 1983, Itakura and Nakamura [Itakura and Nakamura, 1983] first suggested an algorithm in which they allowed a signer to have an RSA modulus with a different bit size. Thereafter, several multi-signature schemes and improvements have been proposed [Okamoto, 1988], [Ohata and Okamoto, 1991], [Horster et al., 1995], [Boyd, 1989], [Park et al., 1997], [Ohta and Okamoto, 1999]. The schemes [Itakura and Nakamura, 1983], [Okamoto, 1988] are not efficient because multi-signature generation and verification
CHAPTER 4. MULTI SIGNATURES

27
time increases as the number of signatories increases. Later, Ohata and Okamoto [Ohata and Okamoto, 1991] proposed a scheme to avoid the restriction of the signing order. But in their scheme, if the length of the partial multi-signature exceeds a predetermined threshold value, the extra bits exceeding the threshold value are appended to a message. So, the length of expanded message depends on the number of signers and the bit size of each signer’s RSA modulus. Micali et al. [Micali et al., 2001] proposed a security notion for multi-signature, where the subset of signers is known a priori, i.e., each signer has to know all participants of the current subgroup of signers, a description of which is hashed and signed along with a message.

We proposed a sequential multi-signature scheme [Das et al., 2004c] based on RSA that uses re-blocking method. The proposed scheme causes bit expansion in block size of a multi-signature but the bit expansion is not greater than the number of signers regardless of their RSA modulus.

4.2.1.1 The Scheme [Das et al., 2004c]

Re-blocking: Let \( N \) is an RSA modulus and \( e \) is a public key with \( gcd(e, \phi(N)) = 1 \), where \( \phi(N) \) is the Euler totient function. For a message \( M \), a minimum value of \( l \) has to be chosen such that \( 0 < M < 2^l N \). Then, \( \phi(2^l N) = 2^{l-1} \phi(N) \) and \( gcd(e, 2^{l-1} \phi(N)) = 1 \). Now, if \( ed \equiv 1 \mod 2^{l-1} \phi(N) \), then \( M^{ed} = M \mod 2^l N \). If \( C = M^e \mod 2^l N \) and \( ed' \equiv 1 \mod 2^{l-1} \), then \( C \mod 2^l = M^e \mod 2^l \). Also, \( M^{ed'} \mod 2^l = C^{ed} \mod 2^l \), which implies \( M \mod 2^l = C^{ed} \mod 2^l \).

Thus, this re-blocking method cannot be used for enciphering large block size messages. But this can be applied to multi signatures scheme, because the signing process involves only hashed message and few other parameters.

Multi-signature Generation: Let there are \( t \) signers sequentially generate a multi-signature and the final signer sends it to the verifier. When each signer \( U_i (i = 1, 2, \cdots t) \) joins to the signing process, each signer generates RSA key pair \((e_i, d_i)\), where \((e_i, N_i)\) is made public and \((d_i, N_i)\) is kept secret.

We use the re-blocking method to solve the moduli clashes problem and for this each signer computes \( l_i \) from \( N_i \) as follows:

\[
l_i = \begin{cases} 
1 & \text{if } i = 1 \\
2^{l_i-1} N_i < 2^{l_i-1} N_{i-1} < 2^l N_i & \text{otherwise.}
\end{cases}
\]
First signer signature generation: The first signer $U_1$ signs a message $M$ with the following steps:

- Compute $S_1 = h(M)^{d_1} \mod 2N_1$, where $h(\cdot)$ is a collision-resistant hash function.
- Send signature $(M, S_1)$ to the second signer $U_2$ over a public channel.

Signature verification by $U_2$: Upon receiving the signed string $(M, S_1)$, $U_2$ verifies it as follows:

- Compute $V_1 = (S_1)^{e_1} \mod 2N_1$.
- Compare $V_1$ and $h(M)$. If they are equal, the signature $(M, S_1)$ is valid. Otherwise, rejects the received signed string.

Signature generation by $U_2$: $U_2$ signs $M$ along with $S_1$ by the following steps:

- Compute $S_2 = (S_1 \oplus h(M))^{d_2} \mod 2^{l_2}N_2$.
- Send signature $(M, S_1, S_2)$ to $U_3$ over a public channel.

For $i = 3, \ldots, t - 1$,

Signature verification by $U_i$: $U_i$ receives the signed string from $U_{i-1}$ and verifies it as follows:

- Compute $V_{i-1} = (S_{i-1})^{e_{i-1}} \mod 2^{l_{i-1}}N_{i-1}$.
- Compute $S_x = V_{i-1} \oplus h(M)$. Accept the signature if $S_x$ and $S_{i-2}$ are equal. Otherwise, reject the received signed string.

Signature generation by $U_i$: $U_i$ signs $M$ along with $S_{i-1}$ by the following steps:

- Compute $S_i = (S_{i-1} \oplus h(M))^{d_i} \mod 2^{l_i}N_i$.
- Send signature $(M, S_{i-1}, S_i)$ to $U_{i+1}$ over a public channel.

Signature verification by $U_t$: Upon receiving the signed string $(M, S_{t-2}, S_{t-1})$, $U_t$ verifies it as follows:

- Compute $V_{t-1} = (S_{t-1})^{e_{t-1}} \mod 2^{l_{t-1}}N_{t-1}$.
- Compute $S_f = V_{t-1} \oplus h(M)$. Accept the received signing string if $S_f$ and $S_{t-2}$ are equal. Otherwise, reject the received signed string.
Signature generation by \( U_t : \) \( U_t \) signs \( M \) along with \( S_{t-1} \) by the following steps:

- Compute \( S_t = (S_{t-1} \oplus h(M))^{d_t} \mod 2^{l_t}N_t \).
- Send multi-signature \((M, S_t)\) to the verifier.

**Multi-signature Verification** : The verifier verifies the multi-signature \((M, S_t)\) as follows:

- Compute
  \[
  V = \left( \cdots ((S_t)^e \mod 2^{h_t}N_t) \oplus h(M)\right)^{e_t-1} \cdots \oplus h(M) \right)^{e_1} \mod 2N_1
  \]
- Compare \( V \) and \( h(M) \). If they are equal, the verifier accepts \( S_t \) as a valid multi-signature of \( M \).

We note that the size of the multi-signature does not grow linearly as the number of signers increases. Moreover, the verification complexity remains constant and is independent of the number of signers.

### 4.2.1.2 Analysis

The security of the scheme is relying on the hardness of large integer factorization. The signature of \( U_t \) is computed as \( S_t = (S_{t-1} \oplus h(M))^{e_t} \mod 2^{l_t}N_t \). The size of the signature is \(|l_t| + |N_t|\). Though the signature \((M, S_{t-1}, S_t)\) travels over the public channel, the adversary cannot get the most significant \(|N_t|\) bits of \( S_t \) from \( S_{t-1} \). Thus, the adversary cannot guess or calculate \( d_t \) from the partial multi-signature until and unless he has the knowledge of \( d_t \).

### 4.2.2 Multi-signature Scheme using Bilinear Pairings

In traditional public key cryptosystems, the public key of an entity is a random bit string picked from a given set that leads to a problem of how the public key is linked with the entity. To solve this problem, public key certification is widely used. However, the certification process needs infrastructure cost and requires a burdensome computation effort to manage the certificates. In 1984, Shamir [Shamir, 1984] first introduced the concept of identity (ID) based cryptography in which the public key of the user is his ID or derived from ID. ID-based cryptosystems are advantageous over the traditional public key cryptosystems, as key distribution and revocation are
simplified. In ID-based cryptosystem, a verifier can verify a signature just by using the signer’s ID. The properties of bilinear pairings [Boneh and Franklin, 2001] allowed to construct ID-based cryptosystem, where public key certification can be avoided. However, ID-based cryptosystems suffer from the drawback of key escrow problem and require secure channel for the key issuance. To overcome these drawbacks, numerous researches in [Boneh and Franklin, 2001], [Chen et al., 2002], [Lee et al., 2004] shown approaches using multiple authorities. Boneh and Franklin [Boneh and Franklin, 2001] and Chen et al. [Chen et al., 2002] considered multiple Key Generation Centers (KGCs) for users’ secret key issuance. However, in their models, multiple KGCs need to check the user’s ID independently, which is quite a burden. Moreover, their models require a secure channel between the user and the KGCs. Recently, Lee et al. [Lee et al., 2004] proposed a key issuing protocol to remove the key escrow in ID-based cryptosystem. In their approach, the system contains one KGC and multiple Key Privacy Authorities (KPAs). The KGC checks the user’s ID and issues blinded partial secret key to the user. Then the multiple KPAs sequentially provide key privacy service to the user’s secret key by issuing their signature in a blinded manner. It is observed that Lee et al.’s protocol is inefficient [Ganeshitti et al., 2005]. This encourages us to work on a secure issuing protocol in ID-based cryptosystem. Here we propose a secure key issuing protocol in ID-based cryptosystem and extend the idea to ID-based multi-signature. The protocol does not require secure channel and avoids key escrow problem. We use multiple key privacy authorities (KPAs) at the same trusted level so that the users’ key privacy is distributed into multiple authorities to eliminate key escrow problem. To avoid secure channel for the key issuance, we use a simple binding-blinding technique.

4.2.2.1 The Model

The proposed model has the following entities:
Key Generator Center (KGC) and Key Privacy Authorities (KPAs) : They are trusted parties, who receives signer’s ID, checks the validity of signer’s ID and issues a partial secret key to the signer in a blinded manner.
Signer : An entity who signs the message with his secret key.
Verifier: An entity who verifies the signature.

The model consists of eight phases : System Setup ($SS$), System Key Setup($KS$), User Registration($UR$), Partial Secret Key Request($PR$), Key Privacy Request($PR$), Secret Key Extract($KE$), Sign($S$) and Verify($V$).
CHAPTER 4. MULTI SIGNATURES

System Setup ($SS$): inputs a security parameter $1^\lambda$; and outputs public system parameters $paramsCDHP$, i.e., $paramsCDHP \leftarrow SS(1^\lambda)$.

System Key Setup ($KS$): inputs KGC and KPAs secret keys; and outputs system public key.

User Registration ($UR$): inputs user identity $ID$; and outputs a registration key $r_{ID} \in \mathbb{Z}_q^*$, i.e., $r_{ID} \leftarrow UR(ID)$.

Partial Secret Key Request ($KR$): inputs user identity $ID$ and blinded public key $D_{ID}$; and outputs a blinded partial secret key $Q'_{0ID}$. That is, $Q'_{0ID} \leftarrow KR(ID, D_{ID})$.

Key Privacy Request ($PR$): inputs $ID$ and $Q'_{0ID}$; outputs $Q_{iID}$ for $i = 1, 2, \ldots, n$. That is, $Q_{iID} \leftarrow PR(ID, Q'_{0ID})$.

Secret Key Extract ($KE$): inputs partial secret keys, user blinding factor; and outputs user secret key. That is, $S_{ID} \leftarrow KE(Q'_{1ID}, Q'_{2ID}, \ldots, Q'_{nID}, r_{ID})$.

Sign ($S$): inputs $paramsCDHP$, user parameter(s), secret key $S_{ID}$ of the user and message $m$; and outputs signature of $m$. That is, $\sigma \leftarrow S(paramsCDHP, user parameter(s), S_{ID}, m)$.

Verify ($V$): inputs the $paramsCDHP$, public key $Q_{ID}$ of the user and signature of the message; and outputs accept or reject. That is, $Result \leftarrow V(paramsCDHP, Q_{ID}, \sigma)$, where $Result \in \{\text{Accepts, Rejects}\}$.

4.2.2.2 Detailed Description

System Setup ($SS$): Suppose $G_1$ is an additive group of prime order $q$, $G_2$ is a multiplicative group of the same order, $P$ is a generator of $G_1$, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map, $H : \{0, 1\}^* \rightarrow G_1$ and $h : \{0, 1\}^* \times G_2 \rightarrow \mathbb{Z}_q^*$ are two hash functions.

System Key Setup ($KS$): The KGC chooses a secret key $s_0 \in \mathbb{Z}_q^*$ and computes public key as $P_0 \leftarrow s_0P$. For $i = 1, 2, \ldots, n$ the KPAs generate their key pairs as follows. The KPA $i$ chooses its secret key $s_i \in \mathbb{Z}_q^*$ and computes public key $P_i \leftarrow s_iP$.

Then KPA $i$ computes its share $Y'_i \leftarrow s_iP_0$ and sends it to the KGC. The KGC collects the shares from all the KPAs and computes the system public key as

$$Y = \sum_{i=1}^{n} Y'_i = s_0(s_1 + s_2 + \cdots + s_n)P.$$ 

The correctness of the system public key is verified by checking the equality...
CHAPTER 4. MULTI SIGNATURES

\[ \hat{e}(Y, P) = \hat{e}(\sum_{i=1}^{n} P_i, P_0). \]

User Registration (UR) : A user, with an identity ID, who wants to obtain the secret key has to first register with KGC in off-line. The user submits his ID and supported credentials. After verification of user credentials, the KGC randomly generates \( r_{ID} \in \mathbb{Z}_q^* \), and stores \( R_{ID} = r_{ID}^{-1} P \) in a Registration Database Table (RDT) with corresponding ID, which is accessible to KPAs also. Then, the user obtains registration key \( r_{ID} \) from KGC.

Partial Secret Key Request (KR) : A user with identity ID computes \( Q_{ID} = H(ID) \) and \( D_{ID} = r_{ID} Q_{ID} \). He requests the KGC to issue a partial secret key by sending ID and \( D_{ID} \) over a public channel. Then the KGC issues a blinded partial secret key as follows:

- Compute the public key of the user as \( Q_{ID} \leftarrow H(ID) \).
- Validate \((ID, D_{ID})\) by checking the equality \( \hat{e}(D_{ID}, R_{ID}) = \hat{e}(Q_{ID}, P) \).
- Compute a blinded partial secret key as \( Q'_{0ID} \leftarrow s_0 D_{ID} \).
- Send \( Q'_{0ID} \) to the user over a public channel.

Here, the registration key \( r_{ID} \) is used as blinding factor and it eliminates the need of secure channel between the user and the KGC. The user verifies the issued blinded partial secret key by checking the equality \( \hat{e}(Q'_{0ID}, P) = \hat{e}(D_{ID}, P_0) \).

Key Privacy Request (PR) : The user requests KPA\(_i\) for \( i = 1, 2, \ldots, n \), to provide key privacy service by sending ID and \( Q'_{0ID} \). The KPA\(_i\) does the following:

- Check the equality \( \hat{e}(Q'_{0ID}, R_{ID}) = \hat{e}(Q_{ID}, P_0) \) to validate \( Q'_{0ID} \).
- Compute \( Q'_{iID} \leftarrow s_i Q'_{0ID} \).
- Send \( Q'_{iID} \) to the user over a public channel.

We note that Lee et al.’s [Lee et al., 2004] scheme needs user ID verification by multiple KGCs, whereas our protocol requires only one KGC to verify the ID and the KPAs to verify only \( Q'_{0ID} \). Moreover, in [Lee et al., 2004], the key securing process is sequential as KPA\(_i\) has to verify the component issued by KPA\(_{i-1}\), and thus the order need to be prefixed. Whereas, in our protocol this process is carried out in parallel and hence all the KPAs verify only one parameter \( Q'_{0ID} \) issued by the KGC.
Secret Key Extract (KE) : The user combines all the partial secret key components received from the KPAs. Then he unblinds $S'_{ID}$ and gets his secret key $S_{ID}$ as follows.

$$S'_{ID} \leftarrow \sum_{i=1}^{n} Q'_{iID} \quad S_{ID} \leftarrow r_{ID}^{-1} S'_{ID} \leftarrow s_0(s_1 + s_2 + ... + s_n)Q_{ID}$$

The user can verify the correctness of his secret key whether $\hat{e}(S_{ID}, P) = \hat{e}(Y, Q_{ID})$.

One may note that the computational overhead in our protocol is significantly less than that Lee et al.’s scheme [Lee et al., 2004], as we use only one blinding factor throughout the protocol. The KGC can remove the entry of $R_{ID}$ from RDT to avoid the increase of RDT volume.

Sign (S) : To sign a message $m$, the signing algorithm $S$ performs the following operations:

- Pick a random integer $k \in \mathbb{Z}_q^*$.
- Compute $r \leftarrow \hat{e}(P, P)^k$ and $c = h(m, r)$.
- Compute $\sigma \leftarrow cS_{ID} + kP$.

The signature on $m$ is the tuple $(\sigma, c, m) \leftarrow S(\text{params}, (k, r), S_{ID}, m)$.

Verify (V) : To verify the signature $(\sigma, c, m)$ for message $m$, the verification phase $V$ performs the following operations:

- Compute $r' \leftarrow \hat{e}(\sigma, P) \cdot \hat{e}(cQ_{ID}, -Y)$.
- Accept the signature if and only if $c = h(m, r')$.

4.2.2.3 The Scheme

Let $\mathcal{U} = \{U_1, U_2, \cdots, U_t\}$ be a group of $t$ signers. The signer $U_i$ gets his public-secret key pair $(Q_{ID_i}, S_{ID_i})$ by running the key issuing protocol given in the previous section.

Serial Multi-signature Scheme: In this case, the first signer signs a message and the second signer signs on the first signer’s signature along with the message. The signature process is completed when the last signer’s signature is done.

Multi-signature Generation : Let $m$ be the message to be signed by the signers as per
the sequence $U_1 \rightarrow U_2 \rightarrow U_3 \rightarrow \cdots \rightarrow U_{n-1} \rightarrow U_n$.

The first signer generates his signature as $(\sigma_1, c_1, m) \leftarrow S(paramsCDHP, (k_1, R_1), S_{ID_1}, m)$, and sends $(\sigma_1, c_1, m)$ to the second signer.

For $i = 2, 3, \cdots, t - 1$,

Signature generation by $U_i : U_i$ extracts $R_{i-1}$ in $(\sigma'_{i-1}, c_{[1,i-1]}, m)$ verification and validates the signature on checking whether $c_{i-1} = h(m, R'_{i-1})$. For his signature generation, computes $R'_i = R_i \cdot R'_{i-1}$ and $(\sigma_i, c_i, m) \leftarrow S(paramsCDHP, (k_i, R'_i), S_{ID_i}, m)$. After that, he computes $\sigma'_i = \sigma_i + \sigma'_{i-1}$. Then, $U_i$ sends $(\sigma'_i, c_{[1,i]}, m)$ to $U_{i+1}$. We note that $U_i$ cannot generate his signature without verifying $U_{i-1}$ signature, because $R'_i \leftarrow R_i \cdot R'_{i-1}$ and $R'_{i-1}$ has to be extracted from $(\sigma'_{i-1}, c_{[1,i-1]}, m)$.

Signature verification by $U_t : U_t$ accepts the signature $(\sigma'_{t-1}, c_{[1,t-1]}, m)$ received from $U_{t-1}$ by checking whether $Accepts \leftarrow V(paramsCDHP, \sigma'_{t-1}, m)$.

Signature generation by $U_t : U_t$ extracts $R'_{t-1}$ in $(\sigma'_{t-1}, c_{[1,t-1]}, m)$ verification and validates the signature on checking whether $c_{t-1} = h(m, R'_{t-1})$. For his signature generation, computes $R'_t = R_t \cdot R'_{t-1}$ and $(\sigma_t, c_t, m) \leftarrow S(paramsCDHP, (k_t, R'_t), S_{ID_t}, m)$. Then, he computes $\sigma'_t = \sigma'_{t-1} + \sigma_t$ and sends the multi-signature $(\sigma'_t, c_{[1,t]}, m)$ to the verifier.

Multi-signature Verification : The verifier verifies the signature $(\sigma'_t, c_{[1,t]}, m)$ by checking whether

$$R'_t = \hat{e}(\sigma'_t, P) \cdot \hat{e}\left(\sum_{i=1}^{t} c_i Q_{ID_i}, -Y\right)$$

and accepts the multi-signature if and only if $c_t = h(m, R'_t)$.

**Parallel Multi-signature Scheme** : The parallel multi-signature allows multiple signers to sign the same message concurrently. Each of the co-signers of the group can independently sends his signature to a designated clerk, who will combine all the individual signatures and send multi-signature to the verifier. We note that the clerk can be any one of the co-signers.

Multi-signature Generation : For $i = 1, 2, \cdots, t$,

- Select an integer $k_i \in Z_{q'}^*$ computes $R_i = \hat{e}(P, P)^{k_i}$ and broadcasts $R_i$ to the remaining $(t - 1)$ signers.
- Compute $R \leftarrow \prod_{i=1}^{t} R_i$, $c \leftarrow h(m, R)$ and $c_i \leftarrow h(m, R_i)$.

Now, $U_i$ generates signature as $(\sigma_i, c_i, m) \leftarrow S(\text{paramsCDHP}, (k_i, R), S_{ID_i}, m)$.

The clerk verifies each individual signature by checking whether

$Accepts \leftarrow V(\text{paramsCDHP}, \sigma_i, m)$. Once all individual signatures are valid, the clerk computes $\sigma' \leftarrow \sum_{i=1}^{t} \sigma_i$ and sends the multi-signature $(\sigma', c, m)$ to the verifier.

Multi-signature Verification: The verifier verifies the signature $(\sigma', c, m)$ by checking whether

$$R = \hat{e}(\sigma', P) \cdot \hat{e} \left( \sum_{i=1}^{t} Q_{ID_i}, -Y \right)^c$$

and accepts the multi-signature if and only if $c = h(m, R)$.

### 4.2.2.4 Security Analysis

The security models [Micali et al., 2001], [Boldyreva, 2003] for multi signatures have considered the signature forgery notion. In the following, we first define the adversary for our scheme and then present the security analysis. The adversary for our multi-signature schemes (MS) is defined with the following capabilities.

**MS-Adversary:** Given $\text{paramsCDHP}$, an adversary $A_{MS}$ can ask hash and signing queries, and can execute the following $\forall j \in [1, l]$ with given $l$.

- Select a message $m_j$, and the signer’s identity $ID_j$.
- Generate a valid partial multi-signature $(\sigma_{j-1}, c)$ by colluding with $U_k$, where $k \neq j$.
- Send $(\sigma_{j-1}, c)$ to $U_j$.
- Get a valid partial multi-signature $(\sigma_j, c)$ from the $U_j$.

We say that $A_{MS}$ is successful if after $l$ iterations, he can compute a multi-signature for a message $m$ such that $m \neq m_j$ and at least one ID of the signers is not in $ID_j \forall j \in [1, l]$.

An adversary $A$ learns the system parameters $\text{paramsCDHP}$ and identity $ID_h$ of the single honest signer. $A$ extracts secret keys corresponding to the rest ($t - 1$) signers’ IDs. Now, $A$ runs multi-signature generation with the honest player on behalf
of \((t - 1)\) corrupted signers on the chosen message \(m'\). The advantage of adversary \(\text{Adv}(A)\) is defined as the probability of \(A\) to output the valid message-set-signature triple \((m', U, \sigma)\), such that \(U_h \in U, \forall(m', U, \sigma) = \text{Accepts}\) and \(U_h\) did not participate in the multi-signature generation on the input message \(m'\).

We say that a multi-signature scheme \(S\) is secure against existential forgery under chosen message attack if there exists a polynomial-time adversary \(A\) with non-negligible advantage \(\text{Adv}(A)\).

In Theorem 1 of Hess’s [Hess, 2002] ID-based signature scheme is secure against existential forgery under adaptive chosen message attack in the random oracle model [Bellare and Rogaway, 1993].

**Theorem 2.** The proposed multi-signatures are secure against existential forgery under chosen message attack in the random oracle model.

*Proof:* Let \(A_{MS}\) be a polynomial-time adversary for our multi-signatures. Let \(A_{HS}\) be a polynomial-time adversary for Hess’s scheme [Hess, 2002]. Using Theorem 1 in [Hess, 2002], we prove that our multi-signatures are secure against existential forgery under chosen message attack in the random oracle model.

The idea behind this proof is that if \(A_{MS}\) manages to frame an honest signer by constructing a valid multi-signature on an arbitrary message without interacting with the honest signer, then \(A_{HS}\) can forge a previously unsigned message. \(A_{HS}\) can query the hash and signing oracles with an identity \(ID\) for any arbitrary message using the above MS-Adversary model. Whenever \(A_{MS}\) wants to get a valid multi-signature to frame an honest signer, it sends the respective signing query to \(A_{HS}\). In the case of serial multi-signature scheme, \(A_{HS}\) forwards the signing query given by \(A_{MS}\) to its signing oracle with the identity and the partial multi-signature (generated by \(S \setminus S_i\)) as input. In the case of parallel multi-signature scheme, \(A_{MS}\) sends the message and identity of the honest user to the \(A_{HS}\), which it forwards to its signing oracles. \(A_{HS}\) returns the respective reply of signing oracle to \(A_{MS}\). It is easy to see that \(A_{MS}\) is successful in its attempt if the reply from \(A_{HS}\) is a valid signature. But, generating a valid signature by \(A_{HS}\) is a contradiction to the result of the Theorem 1 in [Hess, 2002]. Hence the proof.

### 4.3 Conclusion

We have proposed two multi-signature schemes: first one is based on RSA and the second one is on bilinear pairings.
The RSA-based multi-signature scheme uses a re-blocking method that causes bit expansion in the block size of a multi-signature but the bit length of the expansion is not greater than the number of signers. The complexity of the verification process remains constant and is independent of the number of signers involved in the signing process. In addition, the multi-signature provides highest level authenticity as each signer not only signs the received string but also signing the message itself.

Thereafter, using bilinear pairing we proposed a secure key issuing protocol in ID-based cryptosystem and extended it to multi-signature scheme that avoids secure channel for the key issuance. In order to avoid the key escrow, the scheme distributes the privacy of user secret key into multiple trusted authorities. At any point of time if any one of the KPA remains honest, our scheme is secure against the key escrow problem. We have shown that our multi-signature schemes are secure against existential forgery on adaptive chosen message attack in the random oracle model.
Chapter 5

Remote Systems Authentication

Remote systems authentication is an important concern in various electronic commerce applications. In order to get systems resource accessibility, the login request need to be authenticated by the remote system, and subsequently the subscriber or customer needs to assure that the communicating entity is genuine. This section presents our works on remote systems authentication.

5.1 Password-based Authentication in Remote Systems

Password authentication is a convenient method to verify the legality of a user to prevent any kind of possible malicious depredations. In 1981, Lamport [Lamport, 1981] first introduced password-based remote user authentication scheme. Since then, a significant amount of research effort engaged continuously to monitor the merits and de-merits of newly proposed password based authentication techniques. Taking computational cost into consideration, remote systems authentication techniques can be classified into two broad categories, viz. public key based [IEEE P1363.2 D12, 2003] and one-way hash function based schemes. Public key based technique needs high computational cost and relies on some security assumptions e.g., discrete logarithm problem or integer factorization problem [Menezes et al., 1996]. However, the high computation cost does not permit the protocol runs in a small handheld devices (e.g., smart card). In contrast to public key based authentication, hash function based authentication technique is efficient and viable for implementation in smart card. The use of smart card not only makes the protocol secure but also prevents users from distribution of their login-IDs, which effectively prohibits the scenario of many logged in users with the same login-ID. This generally happens in digital libraries and similar
systems, where a subscriber can share his login-ID and password with others.

After Lamport’s scheme [Lamport, 1981], several hash-based remote user authentication schemes using smart card have been proposed and some of them are reviewed by us in the next section. However, most of the schemes provide only one-way authentication, that is, only remote server can check the authenticity of a user. The user cannot check whether he is communicating with the correct server. It is a vital gap where a potential adversary can spoof the server and get valuable information from users. This motivates us to construct an authentication mechanism that provides both user and server authentications. We present a scheme in which the user is assigned a smart card that stores some secret parameters during the registration. The scheme is based on hash function and needs a few arithmetic operations like elliptic curve point addition and multiplication [Hankerson et al., 2004]. The scheme provides mutual authentication between user and server and offers a user friendly password change option without server cooperation.

5.1.1 Related Works

In 1981, Lamport [Lamport, 1981] first proposed a well-known hash-based password authentication scheme. However, Lamport’s scheme suffers high hash overhead and the necessity for password resetting problems decreases its suitability for practical use. In addition, the scheme is vulnerable to the replay attack. Haller [Haller, 1994] proposed a modified version of Lamport’s scheme, but the modified version is also vulnerable to the replay attack. Shimizu [Shimizu, 1990] proposed a one-time password authentication scheme on eliminating the weaknesses of [Lamport, 1981] and [Haller, 1994]. The one-time characteristic is gained by using two variable random numbers that are changed in every authentication. The user has to either memorize two variable random numbers or carry with some sort of portable storage token, e.g., smart card. Later, Shimizu et al. [Shimizu et al., 1998] proposed a token-free one-time password authentication scheme, where user neither memorizes any random number nor carries a portable storage token. Instead, a random number is stored in the server for authenticating the user. If the server receives the correct reply corresponding to the sent random number, then the server authenticates the user and refreshes the stored random number. However, the scheme in [Shimizu et al., 1998] suffers from the man-in-the-middle attack. The above schemes are based on hash-based weak password authentication and cannot resist the password guessing attack [Gong et al., 1993]. In 2000, Sandirigama et al.[Sandirigama et al., 2000] proposed a
hash-based strong-password authentication scheme, but it is vulnerable to the replay attack and the denial-of-service attack noticed in [Ku and Chen, 2001], [Lin et al., 2001]. Additionally, Lin et al. [Lin et al., 2001] proposed a refined scheme and claimed that the scheme can resist stolen-verifier, replay, and denial-of-service attacks. However, Chen and Ku [Chen and Ku, 2002], Tsuji and Shimizu [Tsuji and Shimizu, 2003] noticed the weakness of [Lin et al., 2001]. Peyravian and Zunic [Peyravian and Zunic, 2000] proposed a hash-based password authentication scheme that requires low computational efforts. Later, Hwang and Yeh [Hwang and Yeh, 2002] showed that Peyravian-Zunic’s scheme is vulnerable to password guessing, server spoofing, and stolen-verifier attacks. In addition, Hwang and Yeh [Hwang and Yeh, 2002] proposed an improved scheme using the public key cryptosystem. The use of public-key cryptosystem for password authentication [IEEE P1363.2 D12, 2003] violates the merits of Peyravian and Zunic’s scheme. Later, Das et al. [Das et al., 2004a] proposed hash-based scheme involving two secret parameters, but their construction was weak and insecure.

As hash-based remote user authentication scheme generally suffers with off-line dictionary attacks, people realize that public-key based authentication technique could solve the weaknesses of hash-based authentication techniques. Many remote user authentication system scheme [Chang and Liao, 1994], [Jablon, 1996], [Das et al., 2004d] [Saxena et al., 2004], [Scott, 2004] based on public key, have been proposed in literature, but they require high computation cost. Thus, taking advantages of simple hash-based authentication approach and a few arithmetic operations would be an ideal mechanism to have a scheme that can fit in a small handheld device like smart card.

5.1.2 Mathematical Background

An elliptic curve [Hankerson et al., 2004], defined modulo a prime \( p \), is the set of solutions \((x, y)\) to an equation of the form denoted as \( E(F_p) : y^2 = x^3 + ax + b \mod p \), where \( a \) and \( b \in F_p \) satisfied \( 4a^3 + 27b^2 \neq 0 \mod p \). If \((x, y)\) satisfies the above equation then \( P = (x, y) \) is a point on the elliptic curve.

**Map-to-Point**: An important requirement of our scheme is that the login-ID has to be mapped onto a point on \( E(F_p) \). The recent results [Biham and Chen, 2004], [Wang et al., 2004b] regarding the security of hash functions, prompted us to use Map-to-Point operation though it is an expensive operation than the ordinary hash operation. First, the string has to be converted into an integer and then a mapping is
required from that integer onto a point on the curve. To convert the login-ID into an integer requires the use of a hash function. To map the resulting integer onto a point on the curve, one can use the algorithm by Koblitz [Koblitz, 1994].

**Point Addition**: Let points \( P(x_1, y_1), Q(x_2, y_2) \in E(F_p) \), where \( P \neq \pm Q \). The point \( R(x_3, y_3) = P + Q \in E(F_p) \) is calculated as
\[
x_3 = \left( \frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2, \text{ and } y_3 = \left( \frac{y_2 - y_1}{x_2 - x_1} \right)(x_1 - x_3) - y_1.
\]

**Point Doubling**: Let points \( P(x_1, y_1), Q(x_2, y_2) \in E(F_p) \), where \( P \neq -P \). The point \( R(x_3, y_3) = 2P \in E(F_p) \) is calculated as
\[
x_3 = \left( \frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1, \text{ and } y_3 = \left( \frac{3x_1^2 + a}{2y_1} \right)(x_1 - x_3) - y_1, \text{ where } a \text{ is the coefficient of } x \text{ in } E(F_p).
\]

**Scalar Multiplication of a Point**: The scalar \( n \) multiplication of a curve point \( P \) is typically defined as \( n \)-times addition of \( P \), i.e., \( nP = P + P + \cdots + P \) \((n\text{-times})\). There are many techniques for faster computation of scalar multiplication of a curve point [Brier and Joye, 2003], [Dimitrov et al., 2005].

## 5.1.3 The Scheme

The entities of the proposed scheme are the user, user’s smart card and the remote server. For simplicity we will be referring the remote server as RS throughout the article. The scheme mainly consists of three phases: the registration phase, the authentication phase and the password change phase. The registration phase is a one-time operation, whereas authentication and password change phases are executed as and when required. Initially, RS chooses a secret key \( s \) and publishes a Map-to-Point function \( H : \{0, 1\}^* \rightarrow E(F_p) \).

**Registration Phase**: This phase is executed when a new user wants to register with RS. A new user \( U_i \) registers with RS by the following steps:

R1. \( U_i \) submits his identity \( ID_i \) and password \( PW_i \) to RS.

R2. RS computes \( REG_{ID_i} = s \cdot H(ID_i) + H(PW_i) \).

R3. RS personalizes a smart card with \( ID_i, REG_{ID_i}, H(PW_i), H(\cdot) \) and sends smart card to \( U_i \) in a secure manner.

**Authentication Phase**: This phase is executed every time when a user wants to login with RS. The phase is further divided into user authentication and RS authentication.
phases. In user authentication phase, user sends a login request to RS and RS validates
the request. In RS authentication phase, RS sends a response to the user and the user
validates the response.

**User Authentication** : The user \( U_i \) attaches the smart card to a terminal and enters his
\( ID_i \) and \( PW_i \) which are verified by the smart card. If they are valid, the smart card
performs the following:

UA1. Compute a dynamic coupon \( DID_i = REG_{ID_i} - H(PW_i) \).

UA2. Compute \( R_i = H(DID_i \| t) \), where \( t \) is current date and time of user system.

UA3. Send the login message \( (ID_i, R_i, t) \) to RS over a public channel.

Validation of User Authenticity : Let RS receives the login request at time \( t^* (\geq t) \). The
RS performs the following steps to validate the login message \( (ID_i, R_i, t) \):

UV1. Verify the validity of the time interval between \( t^* \) and \( t \). If \( (t^* - t) \leq \Delta t \), RS
proceeds to step (UV2), where \( \Delta t \) denotes the expected valid time interval for
transmission delay. Otherwise, reject the login message.

UV2. Compute \( DID'_i = s \cdot H(ID_i) \).

UV3. Compute \( R'_i = H(DID'_i \| t) \). If \( R'_i = R_i \), the authenticity of the login request
holds, otherwise reject the request.

**RS Authentication** : If the user authenticity is passed correctly, RS proves its genuine-
ness by the following steps:

SA1. Compute \( X_i = H(DID'_i \| t \| t') \), where \( t' \) is current date and time of RS.

SA2. Send \( (X_i, t', ID_i) \) to the user over public channel.

Validation of RS Authenticity : Let the user receives the response at time \( t'' (\geq t') \).

SV1. Verify the validity of the time interval between \( t'' \) and \( t' \). If \( (t'' - t') \leq \Delta t' \), the
user proceeds to step (SV2), where \( \Delta t' \) denotes the expected valid time interval
for transmission delay. Otherwise, terminate the operation.

SV2. Compute \( X'_i = H(DID_i \| t \| t') \). If \( X'_i = X_i \), RS is authentic and the user starts to
access the system’s resources.
**Password Change Phase:** This phase is invoked whenever a user $U_i$ wants to change his password. By invoking this phase, $U_i$ can easily change his password without taking any assistance from RS. The phase works as follows:

P1. $U_i$ attaches the smart card to a terminal and enters $ID_i$ and $PW_i$. The entered $ID_i$ and $PW_i$ are verified by the smart card. If they are valid, proceeds to step (P2); otherwise, terminates the operation.

P2. $U_i$ is prompted to submit a new password $PW_i^*$. 

P3. The smart card computes $REG_{ID_i}^* = REG_{ID_i} - H(PW_i) + H(PW_i^*)$.

P4. The password has been changed now with the new password $PW_i^*$ and the smart card replaces the old $REG_{ID_i}, PW_i$ by new $REG_{ID_i}^*$ and $PW_i^*$, respectively.

5.1.4 **Analysis of the Scheme**

5.1.4.1 **Security**

We show that the proposed scheme resists the following attacks:

*Resistance to Replay Attack:* Suppose an adversary wants to replay the login request with date and time $t_{\text{new}}$. The replay attack cannot work because it fails the step (UV1) and step (SV1) in the authentication phase for the time interval $(t_{\text{new}} - t)$ and $(t_{\text{new}} - t')$, respectively.

*Resistance to User impersonation Attack:* In our scheme, a valid login message consists of $ID_i, R_i$ and $t_i$ where $R_i = H(DID_i || t)$ and $DID_i = s \cdot H(ID_i)$. The adversary cannot get any clue or useful information from $R_i$ because the security is based on the Map-to-Point function. Further, the adversary can not construct a valid $DID_i$ without the knowledge of server’s secret key $s$. Even a smart card is stolen, the party cannot login to RS without knowing the password of card owner.

*Resistance to RS impersonation attack:* RS impersonation is only possible if the secret key $s$ can be extracted from the stored $REG_{ID_i}$. Another possibility is that replication of a registered smart card. In the former case, the adversary (including valid user) can not get $REG_{ID_i}$ because after validating user password, all the computations are done by the smart card internally. In the later case, replication of a smart card is quite difficult as per present literature and though it happens by some experiments
[Kocher et al., 1999], the execution cost is much higher than the cost of the intended parameter(s).

**Resistance to Insider Attack**: In real life, a user may use a common password to access several systems for his convenience. If the user login message is only password-based and RS maintains passwords table to validate the login message, an insider of RS can impersonate user login by stealing password and subsequently accessing to other systems. In our scheme, RS is not maintaining any password table for validation of user login request, so the scheme is protected from the stolen-verifier attacks. Although, the user submits his password to RS during the registration process, he can change his password by invoking the password change phase at a later time, thereby, the scheme can resist the stolen-verifier and insider attack.

**Resistance to Denial-of-service Attack**: In the password change phase, the denial-of-service attack may occur when RS updates the new verifier or password for the next login without checking the validation of the received message. This makes RS to reject all subsequent login requests of a legal user. Therefore, it is intuitive to check the received new request before updating the verifier or password. In our scheme, when a user logs into RS or to change his password, the smart card checks the validity of the card owner password before processing any instructions. If a smart card is stolen from the valid user, the dishonest party cannot login to RS or to change the password without knowing the valid user password. Thus, denial-of-service attack is not possible in our scheme.

### 5.1.4.2 Efficiency

The authentication phase of the proposed scheme has two stages of computation. In the first stage, the user sends the login request which involves only one elliptic curve point addition and two Map-to-Point operations. The verification of login request requires one elliptic curve point scalar multiplication and one Map-to-Point operation. In the second stage, the server responses with only one Map-to-Point operation, for which the verification requires only one Map-to-Point operation. Thus, the total cost of a successful login requires one elliptic curve point scalar multiplication, one elliptic curve point addition and five Map-to-Point operations. We note that the elliptic curve point multiplication is executed at RS that removes the burden of high computation requirements in a smart card. This makes our scheme very much practical in comparison to the existing schemes [Chang and Liao, 1994], [Jablon, 1996], [Scott, 2004].
Further, our scheme provides both the user and server authentication before gaining the access to the server resources. By this, the spoofing attack is also avoided, which is very crucial, in particular, for the financial transactions.

5.1.5 Conclusion

We proposed a two-way authentication scheme for remote systems using smart card. The proposed scheme provides the following features:
- both user and RS authenticates each other before exchanging the actual data;
- the use of smart card prevents the scenario of many logged in users with the same login-ID;
- users can choose and change their preferred passwords freely without any assistance from RS;
- RS does not maintain any verifier or password table for the verification of login request;

5.2 Pairing-based Authentication in Remote Systems

5.2.1 Authentication in Mobile Communications

With the advent of wireless technology, personal communication systems (PCS) are becoming an increased use of wireless devices. The technology enables portable computers and electronic devices to be equipped with wireless interfaces, allowing networked communication while being on the move. In one hand, it offers a new paradigm of computing, where users carrying portable devices, have access to a shared infrastructure independent of their geographical location. On the other hand, PCS and mobile devices establish strong business case, such as electronic cheque payment, credit and debit applications, secure billing, etc. In these applications, security over the wireless interface is still a major concern in which end-to-end authorization is a primary requirement. Specifically, when financial transactions rely on the mobile devices, a data collection agent at the network Authentication Center (AC) might need to assure the confidentiality, authenticity, integrity of the data and non-repudiation of the sender.

Here, we present a secure protocol for mobile communications. We use bilinear pair-
ings properties [Boneh and Franklin, 2001] to establish a secure data exchange between mobile devices and AC.

5.2.1.1 Security properties

The basic security requirements of an authentication protocol in wireless communications [Horn and Preneel, 1998] are as follows:

*Mutual authentication of mobile device and Authentication Center (AC)*: It is necessary that both the mobile device and AC have to authenticate each other before the actual data transfer takes place.

*Agreement between mobile device and AC on a secret session key with mutual implicit key authentication*: Both mobile device and AC must establish a session key to be used for the data transfer.

*Mutual key confirmation*: This is required to ensure that the other entity possesses the same session key.

*Mutual key control*: The mutual key control refers to the ability of one party to force the choice of a specific session key value.

*Non-repudiation of mobile device*: This ensures that the mobile device can provide a verifiable commitment to purchase items from a value added service provider by use of digital signatures.

*Confidentiality of user identities*: It is desirable to keep both the movements and the mobile users identity as confidential.

5.2.1.2 Related Works

In this section, we discuss few existing protocols based on public key based authentication in wireless communications. The notations used in this section are as follows:
A: Mobile device

$A$: A’s secret key

$y_A$: A’s public key

$r_A, r_B$: Random numbers

$A_{\text{Cert}}$: A’s public key certificate

$h(\cdot)$: One-way hash function

$\sigma_A(M)$: A’s signature on message $M$

$\text{chd}$: Charging data

$B$: Authentication Center (AC)

$x_B$: B’s secret key

$y_B$: B’s public key

$g$: A generator

$B_{\text{Cert}}$: B’s public key certificate

$\{M\}_k$: Message $M$ is encrypted with key $k$

$S \rightarrow R$: $m$ means $S$ sends $m$ to $R$ via public channel

$\text{pay}$: Payment data

We note that the following computations take place either modulo $p$ (a prime of size at least 1024 bits) or modulo $N$ (product of two primes of size at least 1024 bits).

**Yacobi and Shmuely’s protocol**[Yacobi and Shmuely, 1989]: Suppose the public-secret key pair of $A$ is $(y_A, x_A)$, and for $B$ the public-secret key pair is $(y_B, x_B)$, where $y_A = g^{-x_A}$ and $y_B = g^{-x_B}$. The protocol works as follows:

1. $B \rightarrow A: x_B + r_B$.

2. $A \rightarrow B: x_A + r_A$.

The session key is $K_{AB} = g^{r_A r_B}$. This is calculated by $A$ as $K_{AB} = (g^{x_B + r_B y_B})^{r_A}$ and by $B$ as $K_{AB} = (g^{x_A + r_A y_A})^{r_B}$.

Weakness: The security of this protocol is based on the discrete logarithm problem. However, the protocol is insecure against masquerade attack [Horn et al., 2002] i.e., an attacker who obtains a previously used key between $A$ and $B$ to masquerade as $B$.

**Park’s protocol**[Park, 1997]: The public-secret key pair of $A$ is $(y_A, x_A)$, where $y_A = g^{-x_A}$. The public-secret key pair of $B$ is $(y_B, x_B)$, where $y_B = g^{-x_B}$. The protocol works as follows:

1. $B \rightarrow A: g^{x_B + r_B}$.

2. $A \rightarrow B: x_A + r_A$.

Weakness: The session key is $K_{AB} = g^{r_A r_B}$. This is calculated in the same way as in [Yacobi and Shmuely, 1989], but with less computational cost at $A$ since $A$ has already received $g^{x_B + r_B}$. Horn et al. [Horn et al., 2002] noticed the weakness on this protocol, where an attacker who obtains a previously used key between $A$ and $B$ can masquerade as $B$. The condition for success of the attack is that the attacker $C$ obtains a session
key $K_{AB}$ from a previously used key, for which the protocol messages have been previously recorded. Thus, $C$ has the values $K_{AB} = g^{r_A r_B}$, $x_A + r_A$ and $g^{x_B + r_B}$. Now, $C$ starts a new run of the protocol with the entity $A$ in which the message sent by $B$ is a replay of the message $x_B + r_B$ sent by $B$ as in the previous protocol run. Entity $A$ will send a new message $x_A + r_A'$, but by subtracting the recorded message $x_A + r_A$ from this, $C$ obtains $r_A' - r_A$. It is now straightforward to check that the attacker is able to find the new session key as $K_{AB}' = (g^{x_B + r_B} y_B)^{r_A' - r_A} K_{AB} = g^{r_A' r_B}$.

**MSR+DH Protocol** [Beller et al., 1993]: This protocol incorporates both symmetric key and public key techniques. In this protocol, each of the subscribers and the AC chooses secret key and publishes its corresponding public key such that each subscriber-AC pair is bound by a secret key $K_s$, which can only be computed by these two entities. Each entity also obtains a certificate from a certification authority which ensures the authenticity of the public key. For each round of authentication, the AC broadcasts own identity $ID_B$, public key $y_B$, and certificate $B_{Cert}$. After verifying the validity of $B_{Cert}$, the subscriber chooses a random number $r_A$, and sends $\{ID_A, A_{Cert}, y_A\}^{r_A}$ and $r_A^2 \mod N$ to AC. The session key, $K_c$, is then computed as $K_c = \{K_s\}^{r_A}$. To assure that both entities obtain the same $K_c$, another round of message is exchanged. The protocol works as follows:

2. $A \rightarrow B : r_A^2 \mod N, \{ID_A, A_{Cert}, y_A\}^{r_A}$.
3. $A \leftrightarrow B :$ Exchange known message using $K_c$.

Weakness: The protocol requires mobile device to do immense amount computation. Additionally, each subscriber public key needs to be certified by a trusted certifying authority, where the certificate management is also a cumbersome process. Moreover, if one $K_c$ is compromised, the adversary would be able to impersonate the subscriber to make fraudulent calls by replaying the associated cipher text of $r_A$.

**The ASPeCT protocol** [Horn and Preneel, 1998], [Horn et al., 2002], [ASPeCT]: The public-secret key pair of $A$ is $(y_A, x_A)$, where $y_A = g^{-x_A}$. The public-secret key pair of $B$ is $(y_B, x_B)$, where $y_B = g^{-x_B}$. The protocol works as follows:

1. $A \rightarrow B : g^{r_A}$
2. $B \rightarrow A : r_B, h_2(K_{AB}, r_B, B), chd, T_B, B_{Cert}$
3. $A \rightarrow B : \{\sigma_A h_3(g^{r_A}, g^{r_B}, r_B, B, chd, T_B, pay), A_{Cert, pay}\}^{K_{AB}}$. 
The session key is calculated as $K_{AB} = h_1(r_B, (y_A^r)^{x_A})$ and by $B$ as $K_{AB} = h_1(r_B, (g^r)^{x_B})$. Here, $h_1, h_2, h_3$ are one-way hash functions.

Weakness: The main weakness of this protocol is that the identification of $A$ happens as late as message 3. This might be a consequence of user ID confidentiality to ensure user anonymity. However, the first message is the proper place for $A$’s identity. An attacker can simply cut off the third message, disabling $B$ from receiving the message. Of course, with this the attacker neither derives the session key nor gains anything, except the service provision to frustrate its rival value-added service providers.

5.2.1.3 The Proposed Scheme [Das and Saxena, 2005]

We use the bilinear pairings properties and a blinding-binding technique to provide secure end-to-end authorization and senders’ non-repudiation. The protocol consists of a setup phase and an authentication phase.

Setup Phase: Suppose $G_1$ is an additive cyclic group generated by $P$, whose order is a prime $q$, and $G_2$ be a multiplicative cyclic group of the same order. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing mapping, $H : \{0, 1\}^* \rightarrow G_1$ and $h(.)$ are two hash functions.

The AS selects a master-key $s$ and computes public key as $P_{ubAS} = sP$. Then, AS publishes the system parameters $params = (G_1, G_2, \hat{e}, q, P, P_{ubAS}, H, h)$ and keeps $s$ secret.

Let $ID_A$ be the identity$^3$ of a mobile device $A$. We note that during personalization of mobile device $A$, the device provider stores two secret parameters $a_A, b_A \in \mathbb{Z}_q^*$ and hash function $H$ inside the mobile device (or inside subscriber identification module) memory, where $\mathbb{Z}_q^*$ is the set of integers modulo $q$. Once the mobile device joins to the network AC (say $B$), the secret key of the mobile device is computed by the following steps:

1. $A$ computes own public key $P_{ubA} = H(ID_A)$ and then computes $X_A = a_A P_{ubA}, Y_A = a_A b_A P_{ubA}, Z_A = b_A P$ and $W_A = a_A b_A P$.

2. $A \rightarrow B : (X_A, Y_A, Z_A, W_A, P_{ubA})$.

3. $B$ verifies the validity of received parameters by whether $\hat{e}(Y_A, P) = \hat{e}(X_A, Z_A) = \hat{e}(P_{ubA}, W_A)$.

$^3$The identity (ID) of a mobile device could be the unique handset number assigned by the manufacturer.
4. Once the received parameters are valid, $B$ computes $D_A = sY_A$ and $\lambda_A = sZ_A$.

5. $B \rightarrow A: (D_A, \lambda_A)$.

6. On receiving $(D_A, \lambda_A)$, the $A$ checks whether $\hat{e}(D_A, P) = \hat{e}(Y_A, Pub_{AS})$. If it is valid, $A$ computes own secret key as $S_A = a^{-1}_A D_A$. Then, $A$ stores $(W_A, \beta_A, S_A)$ in its memory, where $\beta_A = a_A \lambda_A$.

**Authentication Phase:** This phase is executed every time when a mobile device communicates to the network and wants to execute some financial transactions. For example, the mobile device wants to make a payment for the due bill. The protocol works as follows:

1. $A \rightarrow B: (r_A \oplus \beta_A, W_A, Pub_A)$.

2. $B$ computes $\beta^*_A$ as $sW_A$ and then extracts $r_A$ by $r_A \oplus \lambda_A \oplus \lambda^*_A$.

3. $B \rightarrow A : r_B, h(ID_B \parallel K_{AB} \parallel r_A \parallel r_B \parallel T_B \parallel chd)$.

4. $A \rightarrow B : \{\sigma_A, pay, Pub_A\}_{K_{AB}}$. The AC verifies the signed message as $\hat{e}(\sigma_A, P) = \hat{e}(Pub_A, \lambda_A) \cdot \hat{e}(K_{AB} H(pay \oplus K_{AB} \oplus ID_B \oplus r_B \oplus T_B \oplus chd), P)$. If it holds, AC accepts the message; otherwise, rejects it.

The signing key of the mobile device is calculated as $\sigma_A = S_A + K_{AB} H(pay \oplus K_{AB} \oplus ID_B \oplus r_B \oplus T_B \oplus chd)$, where $K_{AB} = h(r_A || r_B || T_B)$ is the session key.

### 5.2.1.4 Analysis

**Correctness:** Let the mobile devices sends a signed message $M$, where $M = (pay \oplus K_{AB} \oplus ID_B \oplus r_B \oplus T_B \oplus chd)$. The correctness of the protocol can be judged by the following:

$$\hat{e}(\sigma_A, P) = \hat{e}(S_A + K_{AB} H(M), P)$$
$$= \hat{e}(sb_A Pub_A + K_{AB} H(M), P)$$
$$= \hat{e}(Pub_A, \lambda_A) \cdot \hat{e}(K_{AB} H(M), P)$$

**Security:**

Confidentiality of user data: The business data exchanged between $A$ and $B$ is encrypted with the session key $K_{AB}$, where $K_{AB} = h(r_A || r_B || T_B)$. The user $A$ exchanges securely the parameter $r_A$ to $B$, thus no one can have $r_A$ to construct a valid $K_{AB}$. Until and unless an adversary has $K_{AB}$, he cannot read or manipulate the encrypted data.
Recovering secret key from the public parameters: The user A’s secret key is derived from a partial secret issued by the AC. Before issuance of partial secret key $D_A$, an adversary could intercept the public parameters $X_A, Y_A, Z_A$ and $W_A$, but from these he never comes to know the secret parameters $a_A$ and $b_A$. As other party does not know $a_A$, he cannot unbind $D_A$ to form a valid secret key $S_A$. The security of blinding and unblinding relying on the CDHP which is still a hard problem.

Mutual key control: A mutual key control refers to in ability of one party to force the choice of a specific session key value. In our protocol, the session key $K_{AB}$ is computed as $K_{AB} = h(r_A, r_B, T_B)$, so a party cannot be biased to a specific session key of its own choice.

Sender’s Undeniability: The signature of a sender A is verified by his public key, which is being computed from his identity i.e., $Pub_A = H(ID_A)$. Thus, the sender cannot deny his signed transaction.

User anonymity: When a user is roaming in wireless systems, it is desirable to protect the relevant information about him. Assuring the anonymity of a mobile user prevents unintended parties from associating with the messages to/from him or with the sessions in which he participates. Using the blinding-binding technique in our protocol, the anonymity of mobile device’s identity is maintained.

Performance: One may argue that the performance of our protocol is poor compared to the existing protocols. In this context, we note that during the activation or registration, our protocol needs more computational cost to compute the mobile device’s secret key by its own. This computation is a one-time cost and it is not occurred when the actual transaction (any value-added service) taken in place. However, the authentication protocol is as fast as ASPeCT with higher security environments. Moreover, in our protocol, the base-station authentication server does not need to manage any users’ certificates, which is a cumbersome process in the existing protocols.
CHAPTER 5. REMOTE SYSTEMS AUTHENTICATION

5.2.2 Pairing-based Remote User Authentication

5.2.2.1 The Scheme

The entities in the proposed scheme are the user, user’s smart card and the remote system (RS). The scheme consists of mainly three phases - the setup phase, the registration phase and the authentication phase.

Setup Phase: Suppose $G_1$ is an additive cyclic group of order prime $q$, and $G_2$ is a multiplicative cyclic group of the same order. Let $P$ be the generator of $G_1$, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing mapping and $H : \{0, 1\}^* \rightarrow G_1$ is a cryptographic hash function. RS selects a master-key $s$ and computes the public key as $\text{Pub}_{RS} = sP$. Then, RS publishes the system parameters $\text{paramsCDHP} = (G_1, G_2, \hat{e}, q, P, \text{Pub}_{RS}, H)$ and keeps $s$ secret.

Registration Phase: This phase is invoked by the following steps when a new user wants to register with the RS.

R1. Suppose a new user $U_i$ wants to register the RS.

R2. $U_i$ submits his identity $ID_i$ and password $PW_i$ to the RS.

R3. The RS computes $\text{Reg}_{ID_i} = sH(ID_i) + H(PW_i)$.

R4. The RS personalizes a smart card with the parameters $ID_i, \text{Reg}_{ID_i}, H$ and sends smart card to $U_i$ over a secure channel.

Authentication Phase: This phase is executed every time when a user logs into the system. The phase is further divided into the login and verification phases. In the login phase, user sends a login request to the RS. The login request comprises with a dynamic coupon, called DID, which is dependent on the user’s password and RS’s secret key. The RS allows the user to access the system after successful verification of the login request.

[ Login Phase ]

The user $U_i$ attaches the smart card to a terminal and keys $ID_i$ and $PW_i$. If $ID_i$ is identical to the one that is stored in the smart card, the smart card performs the following:

L1. Compute $DID_i = T \cdot \text{Reg}_{ID_i}$, where $T$ is a time stamp.

L2. Compute $V_i = T \cdot H(PW_i)$.

L3. Send the login message $< ID_i, DID_i, V_i, T >$ to the RS over the public channel.
[Verification Phase]

Let the RS receives the login message $< ID_i, DID_i, V_i, T>$ at time $T^* (\geq T)$. The remote system performs the following steps to verify $< ID_i, DID_i, V_i, T>$:

V1. Verify the validity of the time interval between $T^*$ and $T$. If $(T^* - T) \leq \Delta T$, the RS proceeds to step (V2), where $\Delta T$ denotes the expected valid time interval for transmission delay. Otherwise, reject the login message.

V2. Check whether $\hat{e}(DID_i - V_i, P) = \hat{e}(H(ID_i), Pub_{RS})^T$. If it holds, RS accepts the login message; otherwise, rejects it.

**Password Change Phase**: This phase is invoked whenever a user $U_i$ wants to change his password. By invoking this phase, $U_i$ can easily change his password without taking any assistance from the RS. The phase works as follows:

P1. $U_i$ attaches the smart card to a terminal and keys $ID_i$ and $PW_i$. If $ID_i$ is identical to the one that is stored in the smart card, proceeds to step (P2); otherwise, terminates the operation.

P2. $U_i$ submits a new password $PW_i^*$.

P3. The smart card computes

$$Reg_{ID_i}^* = Reg_{ID_i} - H(ID_i) + H(PW_i^*) = s \cdot H(ID_i) + H(PW_i^*) - T \cdot H(PW_i).$$

P4. The password has been changed now with the new password $PW_i^*$ and the smart card replaced $Reg_{ID_i}$ value by $Reg_{ID_i}^*$ value.

5.2.2.2 Analysis

**Correctness**: $\hat{e}(DID_i - V_i, P)$

\[= \hat{e}(T \cdot Reg_{ID_i} - V_i, P)\]
\[= \hat{e}(T \cdot s \cdot H(ID_i) + T \cdot H(PW_i) - T \cdot H(PW_i), P)\]
\[= \hat{e}(s \cdot H(ID_i), P)^T\]
\[= \hat{e}(H(ID_i), Pub_{RS})^T\]

**Performance**: The computation time of our login phase involves two elliptic curve point multiplication and the verification phase needs two pairing operation, one point addition and one hash to point. Typically, a remote user authentication using ElGamal’s signature [ElGamal, 1985] needs four discrete log operation, one scalar multiplication and one hash computation for login phase, whereas two discrete log operation,
one scalar multiplication, one hash computation and one inverse operation requires for verification phase. Thus, in contrast to the remote user authentication using El-Gamal’s signature, our scheme needs more computation for verification of the login message. As the verification process is done at the remote server with large computation system, the computation cost of the verification phase is not a significant overhead. The computation cost at the remote user’s system (e.g., smart card) is a crucial concern and our scheme takes less time. Moreover, our scheme provides a flexible password change option to the users and prevents the scenario of Many logged in users with the Same login ID.

**Security :** In the following, we show that the proposed scheme resists the replay attacks, forgery attacks and insider attacks.

*Replay attack :* The replay attack cannot work because it fails the step (V1) of the verification phase for the time interval ($T_{\text{new}} - T$), where $T_{\text{new}}$ denotes adversary’s computed time stamp.

*Forgery Attack :* It is observed from our scheme that a valid user login message consists of $ID_i$, $DID_i$, $V_i$ and $T$. The $ID_i$ is comprises with $T \cdot RegID_i$, where $RegID_i = s \cdot H(ID_i) + H(PW_i)$. This $RegID_i$ is stored in $U_i$’s smart card during the registration process by the RS. To extract this $RegID_i$ from the smart card is extremely difficult. An adversary cannot construct a valid $RegID_i$ without the knowledge of RS’s secret key $s$ and user’s password. A registered user is also prevented from this forgery attacks as he does not know $RegID_i$. If the adversary intercepts a valid login message, he cannot resend it later as $T$ will be different in the next time and it fails the step (V1) of the verification phase. Moreover, any unauthorized user cannot login to the RS with a valid registered smart card because he does not know the password of a legitimate user of the card owner.

*Insider Attack :* In practice, the user uses a common password to access several servers for his convenience. If the user’s login message is password-based and the remote system maintains verifier or password table for login message, an insider of a remote system could impersonate user’s login on stealing password and gets access of the other systems. In our scheme, the user’s login message is based on the user’s password as well as a RS’s secret key. In our scheme, the RS does not maintain any verifier or password table through which an insider gets the user password and impersonate a valid login. Although, the user submits his password to the RS during the registration process, he can change his password on invoking the password change phase,
thereby, the scheme resists the insider attack.

5.2.3 Conclusion

Using the properties of bilinear pairings, we proposed two secure protocols for remote systems authentication. The mobile communications protocol provides an end-to-end authorization and meets other security properties. The second scheme has all the merits of our previously proposed scheme in Section 5.1. The security of both the protocols is based on the hardness of CDHP.
Chapter 6

Conclusion

The report emphasizes Data authentication and Entity authentication techniques. Digital signatures play a central role for data authentication, data integrity and sender’s undeniability. There are many practical scenario, where digital signature cannot fulfill the applications requirement but proxy signature, multi-signature meets that requirement. We have studied proxy signature, multi-signature, remote systems authentication, and proposed a few schemes for both data and entity authentication. In the previous progress report, we mentioned that the security backbone of our work was primarily based on integer factorization problem. In the present work, we have extended the security assumption to DLP and CDHP.

The proxy signature plays a crucial role where the primary signer is not available, and in his absence, the designated proxy signer performs the primary signer’s role. We have critically reviewed the existing proxy signature schemes from different security assumptions and we analyzed their security and performance in a survey report, which is discussed in Appendix A. It is observed that many times, a paper typically breaks a previous scheme and proposes a new one, which someone breaks later and, in turn, proposes a new one, and so on. Most of such work, though quite important and useful, essentially provides an incremental advance to the same basic theme. With the advancement of ID-based cryptosystems using bilinear pairing, a few pairing-based proxy signatures have been proposed; however, most of the schemes lack inherent key escrow problem and require secure channel for proxy delivery. In addition, the schemes suffer from the proxy revocation. We proposed a pairing-based proxy signature scheme that provides effective revocation mechanism. The proposed scheme avoids the key escrow problem and does not require secure channel in the key issuance stage. Further, we extended the theme to multi-signature for both sequential
and parallel architecture. In the multi-signature protocol, the signer key privacy is distributed to multiple key privacy authority to avoid the key escrow problem. The security of both proxy and multi signatures is based on CDHP and we have shown that our schemes are secure against adaptive chosen-message attack under the random oracle model.

In the entity authentication chapter, we proposed three schemes for remote systems authentication. In the first and second schemes, a user will have a registered smart card and by using that card the customer can authenticate himself as well as confirm that the communicating server is genuine. The most attractive features of our schemes are:

- It prevents the scenario of many logged in users with the same login-ID.
- No password or verifier table to validate the login request.
- It provides a user friendly password change option to the users.
- It withstands replay, guessing, stolen-verifier, impersonation and insider attacks.

In the third scheme, the mobile device securely transacts with the merchants through service provider. The authentication center at the service provider authenticates to the mobile device before the transaction takes place.
Bibliography


Appendix A

Ten years research on proxy signatures: A Survey

A.1 Introduction

Digital signatures are widely used in security mechanisms in applications such as, to check the integrity of a message, to check the authenticity of the origin, and to protect from dishonest repudiation. There are many practical environments where digital signatures do not serve applications requirement. For example, a professor is required to attend two conferences concurrently. In this case, he would take a proxy person who can attend one conference on behalf of him. Proxy signature serves the purpose where the professor (original signer) delegates his signing capability to a student (proxy signer), and the student executes the work on behalf of the professor.

In 1996, Mambo et al. [Mambo et al., 1996] introduced proxy signature scheme and classified proxy signature on the basis of delegation, namely full delegation, partial delegation and delegation by warrant. In full delegation, an original signer gives his secret key to a proxy signer and the proxy signer signs document using original signer’s secret key. The drawback of proxy signature with full delegation is that the absence of a distinguishability between original signer and proxy signer. In partial delegation, the original signer derives a proxy key from his secret key and hands it over to the proxy signer as a delegation of signing rights. In this case, the proxy signer can misuse the delegation of signing rights, because partial delegation does not restrict the proxy signer’s signing capability. The weaknesses of full and partial delegations are eliminated by partial delegation with warrant. A warrant explicitly states the signers’ identity, delegation period and the qualification of the message on which the proxy
signer can sign, etc. Another important requirement of a proxy signature is that the revocation of signing rights (i.e., proxy revocation). The proxy revocation is essential for the situation where original signer key is compromised or any misuse of the delegation of signing rights is noticed. It may so happen that the original signer wants to terminate his delegation power before its expiry.

Mambo et al.’s [Mambo et al., 1996] scheme allows unlimited delegation, i.e., the proxy signer can sign any message because the original signer delegation provides unlimited signing capability to the proxy signer. The unlimited signing capability allows proxy signer to misuse the delegation capability.

In 1997, Kim et al. [Kim et al., 1997] revisited the proxy signatures and restricted the proxy signer using partial delegation with warrant. Subsequently, Zhang [Zhang, 1997a], [Zhang, 1997b] proposed threshold and non-repudiable proxy signature schemes. However, Ghodosi and Pieprzyk [Ghodosi and Pieprzyk, 1999] analyzed the shortcomings of [Zhang, 1997a] and the same is noticed by Lee et al. [Lee et al., 1998]. Petersen and Horster [Petersen and Horster, 1997] proposed another notion, called self-certified keys under different trust levels and used them for delegation of signing rights and delegated signatures and proxy signatures. The work in [Lee et al., 2001a] and [Lee et al., 2003] showed that Pertersen-Horster’s scheme is insecure.

In 1998, Blaze et al. [Blaze et al., 1998] proposed the protocols for atomic proxy cryptography.

In 1999, Okamoto et al. [Okamoto et al., 1999], for the first time, proposed proxy signature based on RSA signature scheme [Rivest et al., 1978]. However, Okamoto et al.’s scheme was designed as proxy unprotected notion. In the same year, Sun proposed two schemes [Sun, 1999], [Sun et al., 1999] on threshold proxy signatures. Then, Lee and Kim [Lee and Kim, 1999] proposed a strong proxy signature scheme. Later, Viswanathan et al. [Viswanathan et al., 1999] proposed a signature scheme for controlled environments.


In 2001, Romao and da Silva [Romao and da Silva, 2001] proposed secure mobile agent with proxy certificates. Lee et al. [Lee et al., 2001a], [Lee et al., 2001b] proposed two proxy signature schemes and highlighted their applications. However, Wang et al. [Wang et al., 2003] noticed that Lee et al.’s scheme [Lee et al., 2001b] is not secure.
Park and Lee [Park and Lee, 2001] another scheme for mobile communications. In 2002, Shum and Wei [Shum and Wei, 2002] proposed a proxy signature scheme with proxy signer privacy protection, but the scheme’s insecurity was noticed in [Sun, and Hsieh, 2003].

Year 2003 saw a very impressive list of publications demonstrating a vigourous interest in the proxy signature study. In this year, a number of new schemes and improvements have been proposed [Tsai et al., 2003], [Hwang and Chen, 2003], [Li et al., 2003], [Boldyreva et al., 2003], [Herranz and Saez, 2003a], [Herranz and Saez, 2003b], [Lee et al., 2003], [Wang et al., 2003], [Chen et al., 2003a], [Chen et al., 2003b], [Shao, 2003], [Lv et al., 2003], [Hwang et al., 2003] [Ivan and Dodis, 2003], [Lal and Awasthi, 2003], [Hsu et al., 2003], [Das et al., 2003a], [Das et al., 2003b], [Lin et al., 2003], [Zhang et al., 2003], [Zhang and Kim, 2003], [Ai-Ibrahim and Cerny, 2003]. However, most of the schemes observed the insecurity of previously proposed schemes and proposed an improved one, which was subsequently broken by others.

In 2004, we have some other schemes [Herranz and Saez, 2004], [Das et al., 2004b], [Wang, 2004], [Chen et al., 2004], [Shao, 2004], [Wang et al., 2004a], [Malkin et al., 2004], [Huang and Wang, 2004], [Tan and Liu, 2004a], [Tan and Liu, 2004b], [Xu et al., 2004], [Zhang et al., 2004] with different security assumptions.

In 2005, Lee and Lee [Lee and Lee, 2005] further addressed the security weaknesses of [Shum and Wei, 2002].

In this survey, we categorize several proxy signature schemes into different constructions based on their security assumptions. We analyze their security notions and computation complexities. In section A.2, we discuss mathematical background for general readership. Section A.3 discusses the security properties of proxy signature. We give proxy signature constructions in section A.4. Section A.5 reviews some of the schemes in detail. Finally, we give overall remarks in section A.6.

### A.2 Preliminaries

#### A.2.1 Discrete Logarithm Problem

The discrete logarithm is the inverse of discrete exponentiation in a finite cyclic group. Given a cyclic group $G$ of order a large prime $p$ with group operation $x$ and a generator $g$, exponentiation in $G$ is defined by $g^x = g \times g \times \ldots \times g$ ($x$-times). Suppose $y = g^x$, then the discrete logarithm of $y$ is $x$ and is written as $\log_g y = x \mod p$. 
In 1976, Diffie and Hellman [Diffie and Hellman, 1976] described a method of exchanging cryptographic keys in which the underlying protocol was based on discrete logarithm problem (DLP). Since then, several key exchange [Boyd and Mathuria, 2003], public key encryption [Canetti et al., 2003], [Menezes et al., 1996], and digital signature protocols [ElGamal, 1985], [Goldwasser et al., 1988], [Schnorr, 1991], [NIST, 2000], [ANSI, 1999] have been proposed in which security assumptions rely on the hardness of the DLP. As many of the forthcoming proxy signature schemes are based on Schnorr’s signature scheme, we briefly discuss the scheme as follows:

**The Schnorr Signature Scheme**[Schnorr, 1991] : The scheme has the following phases:

- **System Parameters** \( (SP_{DLP}) \): Inputs \( 1^k \); and outputs \( \text{params}_{DLP} \). The \( \text{params}_{DLP} \) consists of primes \( p, q \) such that \( 2^{k-1} \leq p < 2^k \) and \( q \) divides \( p-1 \), an element \( g \in \mathbb{Z}_p^* \) of order \( q \), and a hash function \( h : \{0, 1\}^* \rightarrow \mathbb{Z}_q \). In other words, \( \text{params}_{DLP} \leftarrow SP_{DLP}(1^k) \).
- **KeyGen** \( (KG_{DLP}) \): The users agree on a group \( G \) (multiplicative group of integers modulo \( p \) for some prime \( p \) (at least 1024 bits)) with generator \( g \) of prime order \( q \) (160 bits) in which the discrete logarithm problem is hard. The user chooses a secret key \( x \) such that \( 0 < x < q \). The public key is \( y \) where \( y = g^x \mod p \).
- **Sign** \( (S_{DLP}) \): To sign a message \( m \), Choose a random \( k \) such that \( 0 < k < q \) and compute \( r = g^k \mod p \). Compute \( e = H(m||r) \) and \( \sigma = (k - xe) \mod q \). The signature is the pair \( (e, \sigma) \).
- **Verify** \( (V_{DLP}) \): Compute \( r' = g^\sigma y^e \mod p \) and \( e' = H(m||r') \). If \( e' = e \) then the signature is valid. In other words, \( \text{Result} \leftarrow V_{DLP}(\text{params}_{DLP}, y, m, \sigma) \), where \( \text{Result} \in \{\text{Accepts}, \text{Rejects}\} \).

This signature scheme has been proven to be secure under existential forgery attacks [Pointcheval and Stern, 1996].

### A.2.2 Bilinear Pairings

Bilinear pairings were first introduced to elliptic curve cryptography for destructive methods like the MOV reduction [Menezes et al., 1993]. With the help of the Weil pairing, the authors of [Menezes et al., 1993] showed a way to reduce the discrete logarithm problem on supersingular elliptic curves to the discrete logarithm problem of an extension of the underlying finite field. Later, Frey and Ruck [Frey and Ruck, 1994]
extended this attack to general elliptic curves with the Tate pairing. However, the
Weil pairing and Tate pairing can also be used as a constructive tool for cryptography.
The properties of bilinear pairings allowed to construct identity based cryptosystem
[Boneh and Franklin, 2001], [Boneh et al., 2001], [Cocks, 2001], [Hess, 2002]. In the fol-
lowing, the properties of bilinear pairings are listed.
Suppose $G_1$ is an additive cyclic group generated by $P$, whose order is a prime $q$, and $G_2$ is a multiplicative cyclic group of the same order. A map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is called a bilinear mapping if it satisfies the following properties:
- Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} \forall P, Q \in G_1$ and $a, b \in \mathbb{Z}_q^*$;
- Non-degenerate: There exist $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$;
- Computable: There is an efficient algorithm to compute $\hat{e}(P, Q) \forall P, Q \in G_1$.
In general, $G_1$ is a group of points on an elliptic curve and $G_2$ is a multiplicative sub-
group of a finite field.

**Computational Problems**

Definition 1. Discrete Logarithm Problem (DLP): Given $Q, R \in G_1$, find an integer $x \in \mathbb{Z}_q^*$ such that $R = xQ$.

The MOV and FR reductions: Menezes et al. [Menezes et al., 1993] and Frey and Ruck [Frey and Ruck, 1994] show a reduction from the DLP in $G_1$ to the DLP in $G_2$.

The reduction is: Given an instance $Q, R \in G_1$, where $Q$ is a point of order $q$, find $x \in \mathbb{Z}_q^*$ such that $R = xQ$.

Let $T$ is an element of $G_1$ such that $g = \hat{e}(T, Q)$ has order $q$, and let $h = \hat{e}(T, R)$. Using bilinear property of $\hat{e}$, we have $\hat{e}(T, R) = \hat{e}(T, Q)^x$. Thus, DLP in $G_1$ is no harder than the DLP in $G_2$.

Definition 2. Computational Diffie-Hellman Problem (CDHP): Given $(P, aP, bP)$ for $a, b \in \mathbb{Z}_q^*$, compute $abP$.

The advantage of any probabilistic polynomial-time algorithm $A$ in solving CDHP in $G_1$, is defined as $\text{Adv}_{A,G_1}^{CDH} = \text{Prob}[A(P, aP, bP) = abP : a, b \in \mathbb{Z}_q^*]$. For every probabilistic algorithm $A$, $\text{Adv}_{A,G_1}^{CDH}$ is negligible.

Definition 3. Decisional Diffie-Hellman Problem (DDHP): Given $(P, aP, bP, cP)$ for $a, b, c \in \mathbb{Z}_q^*$, determine whether $c \equiv ab \mod q$. This is same as MOV and FR reductions.

The advantage of any probabilistic polynomial-time algorithm $A$ in solving DDHP in $G_1$ is defined as $\text{Adv}_{A,G_1}^{DDH} = \text{Prob}[A(P, aP, bP, cP) = 1] – \text{Prob}[A(P, aP, bP, abP) = 1] : a, b \in \mathbb{Z}_q^*$. For every probabilistic polynomial-time algorithm $A$, $\text{Adv}_{A,G_1}^{DDH}$ is negligible.

Definition 4. Gap Diffie-Hellman (GDH) group: A prime order group $G_1$ is a GDH group if there exists an efficient polynomial-time algorithm which solves the DDHP
APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

in $G_1$ and there is no probabilistic polynomial-time algorithm which solves the CDHP with non-negligible probability of success. The domains of bilinear pairings provide examples of GDH groups. The MOV reduction provides a method to solve DDH in $G_1$, whereas there is no known efficient algorithm for CDH in $G_1$.

Definition 5. Bilinear Diffie-Hellman Problem (BDHP): Given $(P, aP, bP, cP)$ for $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc}$.

Definition 6. Weak Diffie-Hellman Problem (WDHP): Given $(P, Q, aP)$ for $a \in \mathbb{Z}_q^*$, compute $aQ$.

The Hess’s Signature Scheme [Hess, 2002]: The signature scheme has the following phases: System Parameters ($\mathcal{SP}_{CDHP}$), Extract ($\mathcal{E}_{CDHP}$), Sign ($\mathcal{S}_{CDHP}$) and Verify ($\mathcal{V}_{CDHP}$).

System Parameters ($\mathcal{SP}_{CDHP}$): It takes $1^k$ and master-key $s$ as inputs; outputs $\mathit{params}_{CDHP}$. The $\mathit{params}_{CDHP}$ includes groups $G_1, G_2$ of order prime $p$; a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$; hash functions and public key of PKG ($Q_{PKG} = sP$). The PKG keeps $s$ secret. In other words, $\mathit{params}_{CDHP} \leftarrow \mathcal{SP}_{CDHP}(1^k)$

Extract ($\mathcal{E}_{CDHP}$): It takes $\mathit{params}_{CDHP}$, user public key $Q = H(ID)$ as inputs; outputs user secret key $S_{ID} = sQ$. In other words, $S_{ID} \leftarrow \mathcal{E}_{CDHP}(\mathit{params}_{CDHP}, Q_{ID})$.

Sign ($\mathcal{S}_{CDHP}$): To sign a message $m$, the signer chooses an arbitrary $P_1 \in G_1$, picks a random $k \in \mathbb{Z}_q^*$ and computes

- $r = \hat{e}(P_1, P)^k$.
- $c = h(m, r)$.
- $\sigma = cS_{ID} + kP_1$.

The signature of $m$ is the tuple $(m, c, \sigma)$.

In other words, $\sigma \leftarrow \mathcal{S}_{CDHP}(\mathit{params}_{CDHP}, r, S_{ID}, m)$

Verify ($\mathcal{V}_{CDHP}$): The signature $(m, c, \sigma)$ is verified by the following checking:

- Compute $r' = \hat{e}(\sigma, P) \cdot \hat{e}(H(ID), -Q_{PKG})^c$.
- Accept the signature if and only if $c = h(m, r)$.

In other words, $\text{Result} \leftarrow \mathcal{V}_{CDHP}(\mathit{params}_{CDHP}, Q_{ID}, m, \sigma)$

The Hess’s signature scheme is proven secure against existential forgery on adaptive chosen-message attack under the assumption that CDHP is hard [Hess, 2002].

A.2.3 Integer Factorization Problem

The integer factorization (also known as prime decomposition) problem is: Input a positive integer; output it as a product of prime numbers. The problem holds a strong se-
security assumption in cryptography, complexity theory, and quantum computers. The following is the most widely used signature scheme based on integer factorization problem.

The RSA signature Scheme [Rivest et al., 1978]: The RSA (Rivest, Shamir and Adleman) signature scheme is one of the first great advances in public key encryption and signature schemes.

The signature scheme has following phases: System Parameters($SP_{RSA}$), KeyGen ($KG_{RSA}$), Sign ($S_{RSA}$) and Verify ($V_{RSA}$). The phases are as follows:

System Parameters($SP_{RSA}$) : Inputs $1^k$, primes $p$, $q$; and outputs $params_{RSA}$. The $params_{RSA}$ consists of $N = pq$ and a hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_N$. In other words, $params_{RSA} \leftarrow SP_{RSA}(1^k)$.

KeyGen ($KG_{RSA}$) : Choose two large prime numbers $p$ and $q$ randomly and independently of each other. Choose an integer $e$ such that $1 < e < \phi(N)$ which is co-prime to $\phi(N)$, where $\phi(N) = (p - 1)(q - 1)$. Compute $d$ such that $de = 1 \mod \phi(N)$. The public key consists of the modulus $N$ and the public exponent $e$. The secret key consists of the modulus $N$ and the secret exponent $d$, which must be kept secret. In other words, user secret key $\leftarrow KG_{RSA}(params_{RSA},$ user public key).

Sign ($S_{RSA}$) : To sign a message $m$,

Compute $\sigma = H(m)^d \mod N$. The signature of a message $m$ is the tuple $(m, \sigma)$.

In other words, $\sigma \leftarrow S_{RSA}(params_{RSA}, d, m)$.

Verify ($V_{RSA}$) : Compute $m' = \sigma^e \mod N$. The signature is valid if $m' = m$.

In other way, Result$\leftarrow V_{RSA}(params_{RSA}, e, m, \sigma)$.

This signature scheme has been proven to be secure under the assumption of integer factorization problem is hard [Boneh, 1999].

A.3 Security Properties of Proxy Signature

Desirable security properties of proxy signatures have evolved over this period. A widely accepted list of required properties at this juncture is given below:

- Strong unforgeability: A designated proxy signer can create a valid proxy signature on behalf of the original signer. But the original signer and other third parties cannot create a valid proxy signature.

- Strong identifiability: Anyone can determine the identity of corresponding proxy signer from the proxy signature.
- Strong undeniability: Once a proxy signer creates a valid proxy signature on behalf of the original signer, he cannot deny the signature creation.

- Verifiability: The verifier can be convinced of the signers’ agreement from the proxy signature.

- Distinguishability: Proxy signatures are distinguishable from the normal signatures by everyone.

- Secrecy: The original signer secret key cannot be derived from any information, such as the shares of the proxy key, proxy signatures, etc.

- Prevention of misuse: The proxy signer cannot use the proxy key for other purposes than it is made for. That is, he cannot sign message with the proxy key that have not been defined in the warrant. If he does so, he will be identified explicitly from the warrant.

A.3.1 Classification of Proxy Signature

According to the nature of delegation capability, proxy signature can be classified as proxy-unprotected, proxy-protected and threshold notions. This differentiation is important in practical applications, since it enables proxy signature schemes to avoid potential disputes between the original signer and proxy signer.

A.3.1.1 Proxy-Unprotected Notion

This is the scenario when an original signer gives his signing rights (full delegation with warrant) to a proxy signer. The original signer sends a signed warrant to the proxy signer, who then uses this information to generate proxy signatures by executing a standard signature scheme. When a proxy signature is sent, the recipient checks its validity according to the corresponding standard signature verification process. As the proxy signer does not append his secret key on top of the received delegation, a dishonest original signer can sign the message and later claim that the signature was created by the proxy signer. This type of proxy signature primarily lacks strong unforgeability property.
A.3.1.2 Proxy-Protected Notion

This is the scenario when the proxy signer uses his secret key to safeguard him from the original signer’s forgery. In this case, the original signer sends a signed warrant to the proxy signer, who then uses it to construct a proxy key by appending his secret key. With the proxy key, the proxy signer can generate proxy signatures by executing a standard signature scheme. When a proxy signature is sent, the recipient first computes the proxy public key from some public information, and then checks its validity according to the corresponding standard signature verification process. By this technique, neither the original signer frames proxy signer nor the proxy signer frames original signer.

A.3.1.3 Threshold Notion

In a threshold proxy signature, the proxy key is shared by a group of $n$ proxy signers. In order to produce a valid proxy signature on a given message $m$, individual proxy signer produce his partial signature on that message, and then combines them into a full proxy signature on message $m$.

In a $(t, n)$ threshold proxy signature scheme, the original signer delegates his signing capability to a proxy group of $n$ members. Any $t$ or more proxy signers of the group can cooperatively issue a proxy signature on behalf of the original signer, but $(t - 1)$ or less proxy signers cannot forge a signature.

A.4 Models of Proxy Signature

A.4.1 DLP-based and RSA-based Proxy Signature

It consists of the following phases:\footnote{The phase $A$ for DLP is $A_{DLP}$ whereas for RSA it is $A_{RSA}$. The $params$ means $paramsDLP$ or $paramsRSA$ for the respective algorithm.}: System Parameters($SP$), KeyGen($KG$), Signing rights Generation($OS$), Signing rights Verification($OV$), Proxy Key Generation($PKG$), Proxy Signature Generation($PS$) and Proxy Signature Verification($PV$).

The entities involved in the model are:
- a trusted party who certifies the public key.
- an original signer, who delegates his signing capability to a proxy signer.
- a proxy signer, who signs the message on behalf of the original signer.
- a verifier, who verifies the proxy signature and decides to accept or reject.

System Parameters($\mathcal{SP}$): $\text{params} \leftarrow \mathcal{SP}(1^k)$.

KeyGen($\mathcal{KG}$):
- Original signer public key $y_o \leftarrow \mathcal{KG}(\text{params}, \text{secret key } x_o)$.
- Proxy signer public key $y_p \leftarrow \mathcal{KG}(\text{params}, \text{secret key } x_p)$.

Signing rights Generation($\mathcal{OS}$): $\sigma_o \leftarrow \mathcal{OS}(\text{params}, x_o, \omega)$
It takes $\text{params}, x_o, \omega$ as inputs; outputs signature on $\omega$.

Signing rights Verification($\mathcal{OV}$): It takes $\text{params}, y_o, \omega, \sigma_o$ as inputs; outputs accept or reject.
That is $\text{Result} \leftarrow \mathcal{OV}(\text{params}, y_o, \omega, \sigma_o)$; $\text{Result} \in \{\text{Accept, Reject}\}$

Proxy Key Generation($\mathcal{PKG}$): $\rho_p \leftarrow \mathcal{PKG}(\text{params}, \sigma_o, x_p, \text{random number})$
It takes $\text{params}, \sigma_o, x_p$ and random number as inputs; outputs proxy key $\rho_p$. Typically, the proxy signer uses simple arithmetic operation to form a proxy key.

Proxy Signature Generation($\mathcal{PS}$): $\sigma_p \leftarrow \mathcal{PS}(\text{params}, \rho_p, m)$
It takes $\text{params}, \rho_p$ and message $m$ as inputs; outputs signature on $m$.

Proxy Signature Verification($\mathcal{PV}$): $\text{Result} \leftarrow \mathcal{PV}(\text{params}, y_o, y_p, m, \sigma_p)$
It takes $\text{params}, y_o, y_p, m$ and $\sigma_p$ as inputs; outputs accept or reject.

A.4.2 Pairing-based Proxy Signature Scheme

It consists of the following phases\footnote{The phase $\mathcal{A}$ for pairing-based notion is $\mathcal{A}_{\text{CDHP}}$ and $\text{params}$ stands for $\text{params}_{\text{CDHP}}$.}: System Parameters($\mathcal{SP}$), Extract($\mathcal{E}$), Signing rights Generation($\mathcal{OS}$), Signing rights Verification($\mathcal{OV}$), Proxy Key Generation($\mathcal{PKG}$), Proxy Signature Generation($\mathcal{PS}$) and Proxy Signature Verification($\mathcal{PV}$).

The entities involved in the model are:
- a trusted party, say the Private Key Generator (PKG).
- an original signer, who delegates his signing capability to a proxy signer.
- a proxy signer, who signs the message on behalf of the original signer.
- a verifier, who verifies the proxy signature and decides to accept or reject.

System Parameters($\mathcal{SP}$): $\text{params} \leftarrow \mathcal{SP}(1^k)$.

Extract ($\mathcal{E}$):
- Original signer secret key $x_o \leftarrow \mathcal{E}(\text{params}, \text{public key } Q_o)$, where $Q_o = H(\text{ID}_o)$. 
Proxy signer secret key $x_p \leftarrow E(\text{params}, \text{public key } Q_p)$, where $Q_p = H(ID_p)$.

### Signing rights Generation (OS):

$$\sigma_o \leftarrow OS(\text{params}, x_o, \omega)$$

It takes $\text{params}$, $x_o$ and a warrant $\omega$ as inputs; outputs signature on $\omega$.

### Signing rights Verification (OV):

$$\text{Result} \leftarrow OV(\text{params}, y_o, \omega, \sigma_o)$$

It takes $\text{params}$, $y_o$, $\omega$ and $\sigma_o$ as inputs; outputs accept or reject.

### Proxy Key Generation (PKG):

$$\rho_p \leftarrow PKG(\text{params}, \sigma_o, x_p, \text{random number})$$

It takes $\text{params}$, $\sigma_o$, $x_p$ and random number as inputs; outputs proxy key $\rho_p$. Typically, the proxy signer use simple arithmetic operation to form a proxy key.

### Proxy Signature Generation (PS):

$$\sigma_p \leftarrow PS(\text{params}, \rho_p, m)$$

It takes $\text{params}$, proxy key $\rho_p$ and message $m$ as inputs; outputs signature on $m$.

### Proxy Signature Verification (PV):

$$\text{Result} \leftarrow PV(\text{params}, y_o, y_p, m, \sigma_p)$$

It takes $\text{params}$, $y_o$, $y_p$, $m$ and $\sigma_p$ as inputs; outputs accept or reject.

## A.5 Review of Some Proxy Signature Schemes

### Conventions and Notation for DLP based schemes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>A large prime</td>
</tr>
<tr>
<td>$\mathbb{Z}_p$</td>
<td>Set of integers modulo $p$</td>
</tr>
<tr>
<td>$\mathbb{Z}_p^*$</td>
<td>Multiplicative group of $\mathbb{Z}_p$</td>
</tr>
<tr>
<td>$g$</td>
<td>Generator of order $p - 1$ in $\mathbb{Z}_p^*$</td>
</tr>
<tr>
<td>Alice</td>
<td>Original signer</td>
</tr>
<tr>
<td>Bob</td>
<td>Proxy signer</td>
</tr>
<tr>
<td>$x_o, x_p$</td>
<td>Secret key of Alice and Bob, respectively</td>
</tr>
<tr>
<td>$y_o$</td>
<td>Public key of Alice, where $y_o \leftarrow g^{x_o} \mod p$</td>
</tr>
<tr>
<td>$y_p$</td>
<td>Public key of Bob, where $y_p \leftarrow g^{x_p} \mod p$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>A warrant</td>
</tr>
<tr>
<td>$h(.)$</td>
<td>A collision-resistant one-way hash function.</td>
</tr>
</tbody>
</table>

### A.5.1 DLP-Based Proxy Signature Schemes

#### A.5.1.1 Mambo, Usuda and Okamoto [Mambo et al., 1996]

Introduced the concept of proxy signatures. They classified the proxy signature on the basis of the degree of delegation, namely, full delegation, partial delegation and delegation by warrant. In the scheme both proxy unprotected and proxy protected
notions are envisaged with their respective security properties. As we are more focused on proxy-protected scheme, here we give proxy-protected notion.

**Assumption:** DLP is hard.

**Protocol:**
Alice chooses a secret key $x_o$ and generates public key as $y_o \leftarrow KG_{DLP}(paramsDLP, x_o)$.
Bob chooses a secret key $x_p$ and generates public key as $y_p \leftarrow KG_{DLP}(paramsDLP, x_p)$.

**Signing rights Generation:** Alice chooses a random number $k_o \in \mathbb{Z}_{p-1}^*$ and computes $r_o = g^{k_o} \mod p$. Then, Alice computes $\sigma_o \leftarrow OS(paramsDLP, (k_o, r_o), x_o, deleg)$. 

**Signing rights Verification:** Bob accepts $\sigma_o$ if $Accepts \leftarrow OV(paramsDLP, r_o, y_o, deleg, \sigma_o)$.

**Proxy Key Generation:** Bob computes proxy key as $\rho_p \leftarrow PG(paramsDLP, \sigma_o, x_p, y_p)$.

**Proxy Signature Generation:** The proxy signature on message $m$ is computed as $\sigma_p \leftarrow PS(paramsDLP, \rho_p, m)$.

**Proxy Signature Verification:** The verifier accepts the proxy signature if and only if $Accepts \leftarrow PV(paramsDLP, y_o, m, \sigma_p)$.

**Security:** The underlying security of the scheme is based on the hardness of discrete logarithms problem. However, the scheme has two weaknesses. Firstly, Unlimited delegation, i.e., Bob can sign any message on behalf of Alice because Alice has delegated unlimited signing rights to Bob. Secondly, Proxy transfer, i.e., If Bob transfers Alice’s delegation power to any other party $B$ then $B$ also can sign any message on behalf of Alice. In other words, the scheme does not satisfy the prevention of misuse security property.

A.5.1.2 Kim, Park and Won [Kim et al., 1997]

Proposed proxy signature for partial delegation with warrant and proxy signature for threshold delegation.

**Assumption:** DLP is hard.

**Protocol for Partial Delegation with Warrant:** The proxy signatures for partial delegation with warrant combines the benefit of [Mambo et al., 1996] partial delegation and Neuman’s [Neuman, 1993] delegation by warrant. The scheme works as follows: Alice chooses a secret key $x_o$ and generates public key as $y_o \leftarrow KG_{DLP}(paramsDLP, x_o)$. 

APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

Bob chooses a secret key \( x_p \) and generates public key as \( y_p \leftarrow KG_{DLP}(\text{paramsDLP}, x_p) \).

Signigng rights Generation: Alice chooses a random number \( k_o \in \mathbb{Z}_{p-1}^* \) and computes \( r_o = g^{f_o} \mod p \). Then Alice computes \( \sigma_o \leftarrow OS(\text{paramsDLP}, (k_o, r_o), x_o, \omega) \).

Signigng rights Verification: Bob accepts \( \sigma_o \) if and only if \( Accepts \leftarrow OV(\text{paramsDLP}, r_o, y_o, \omega, \sigma_o) \).

Proxy Key Generation: Bob computes proxy key as \( \rho_p \leftarrow PG(\text{paramsDLP}, \sigma_o, r_o, x_p, \omega) \).

Proxy Signature Generation: The proxy signature on message \( m \) is computed as \( \sigma_p = PG(\text{paramsDLP}, \rho_p, m) \).

Proxy Key Generation: The proxy signature on message \( m \) is computed as

\[
\sigma_p = PG(\text{paramsDLP}, \rho_p, m).
\]

Protocol for Threshold Delegation: In the threshold delegation, Alice sends her delegation to a proxy group so that to sign a message the proxy signer’s power is shared. Alice chooses a secret key \( x_o \) and generates public key \( y_o \leftarrow KG_{DLP}(\text{paramsDLP}, x_o) \).

Each proxy signer acts as a dealer with a random secret \( u_i \), chooses a random polynomial such that \( f(x) = u + a_1 x + \cdots + a_{t-1} x^{t-1} \mod p - 1 \). The proxy group keys are generated as follows:

Public keys: \( g^u \mod p, g^{a_1} \mod p, \ldots, g^{a_{t-1}} \mod p \).

Secret keys: \( x_{p,i} = u + a_1 i + \cdots + a_{t-1} i^{t-1}; i = 1, 2, \ldots, t \).

Signigng rights Generation: Alice chooses a random number \( k_o \in \mathbb{Z}_{p-1}^* \) and computes \( r_o = g^{f_o} \mod p \). Then Alice computes \( \sigma_o \leftarrow OS(\text{paramsDLP}, (k_o, r_o), x_o, \omega) \).

Proxy Sharing: To share \( \sigma_o \) in a threshold manner with threshold \( t \), Alice chooses random \( b_j \in \mathbb{Z}_{p-1}; j = 1, 2, \cdots, t-1 \), and publishes the values \( B_j = g^{b_j}, j = 1, 2, \cdots, t-1 \). Then, she computes the proxy share \( \sigma_i \) as \( \sigma_i = f'(i) = \sigma_o + b_1 i + \cdots + b_{t-1} i^{t-1} \).

Signigng rights Verification: Each proxy signer accepts \( \sigma_i \) if and only if

\[
g^{\sigma_i} = y_o^{h(\omega,r_o)} r_o \cdot \prod_{j=1}^{t-1} B_j^{(i^j)} \mod p.
\]

Proxy Key Generation: Each proxy signer computes proxy key as

\[
\rho_{p,i} = PG(\text{paramsDLP}, \sigma_i, r_o, x_{p,i}, \omega).
\]

Proxy Signature Generation: To sign a message \( M \), each proxy signer computes \( v = h(l, M) \), where \( l = g^r \mod p \) (\( r \) is secret to the proxy signer). Then, the proxy signer computes \( \lambda_i = s_i + \sigma_i v \mod p - 1 \) and reveals \( \lambda_i \), where \( s_i = f(i) = r + a_1 i + \cdots + a_{t-1} i^{t-1} \).

On validating \( \lambda_i \), each proxy signer computes \( \sigma \) satisfying \( \sigma = r + \sigma_a v = f(0) + f'(0) v \mod p - 1 \) by applying Lagrange formula to \( \{\lambda_i\} \). The proxy signature on \( M \) is the tuple \( (\omega, r_o, M, \sigma, v) \).

Proxy Signature Verification: The verifier checks whether

\[
v' = g^\sigma \cdot (y_o \cdot y_p)^{h(\omega,r_o) r_o} \cdot v
\]
mod \( p \), and then whether \( v = h(v', M) \).

**Security:** To the best of our knowledge the proposed first scheme (i.e., partial delegation with warrant) is still unbroken. However, the intuition in this paper that using warrant does not require proxy revocation is not correct. There are many situations where proxy revocation is a must although warrant explicitly states the validity and restricts the message signing. Further, Sun et al. [Sun et al., 1999] showed the insecurity of threshold delegation.

### A.5.1.3 Zhang [Zhang, 1997a]

Proposed threshold and nonrepudiable proxy signature schemes, where both the original signer and the proxy signer can not falsely deny their signature.

**Assumption:** DLP is hard.

**Protocol:**

Alice chooses a secret key \( x_o \) and generates public key as \( y_o \leftarrow K_G_{DLP}(\text{paramsDLP}, x_o) \).

Assume that there is a group of \( n \) proxy signers \( p_i, i = 1, \cdots, n \).

**Proxy Generation:** Alice picks \( k_o \in \mathbb{Z}_{p-1} \), computes \( R = g^{k_o} \mod p \), and broadcasts \( R \). The proxy signer randomly selects \( \alpha_i \in \mathbb{Z}_{p-1} \), computes \( y_{p_i} = g^{\alpha_i} R \mod p \), checks whether \( y_{p_i} \in \mathbb{Z}_{p}^* \) and if it holds (otherwise recomputes another \( y_{p_i} \)), broadcasts \( y_{p_i} \).

Alice computes \( \hat{R} = \prod_{i=1}^{n} y_{p_i} \), and \( \hat{s} = n^{-1} \hat{R} x_o + k \mod p - 1 \) and broadcasts \( \hat{s} \). Then, each proxy signer computes \( \hat{R} = \prod_{i=1}^{n} y_{p_i}, \sigma_{p_i} = \hat{s} + \alpha_i \mod p - 1 \), and checks if the equality holds: \( g^{\hat{s}} = y_{n}^{-1} \hat{R}^{R} \mod p \). If it holds, proxy signer accepts \( \sigma_{p_i} \) as a valid proxy share from Alice.

The threshold proxy signature and verification are done as in the schemes [Harn, 1994], [Gennaro et al., 1996].

**Security:** It is computationally infeasible for Alice and for other parties to produce a proxy signature because finding the proxy shares is equivalent to solving the DLP. However, Lee et al.’s [Lee et al., 1998] pointed out some weaknesses in Zhang’s threshold proxy signatures [Zhang, 1997a]. Later, some additional attacks are commented in [Ghodosi and Pieprzyk, 1999].
A.5.1.4 Petersen and Horster [Petersen and Horster, 1997]

Proposed a protocol for self-certified keys issuance under different trust level and used them for delegation of signing rights and delegated signatures, proxy signatures.

**Assumption:** DLP is hard.

**Protocol:**

Alice chooses a secret key $x_o$ and generates public key as $y_o \leftarrow KG_{DLP}(params_{DLP}, x_o)$.

Bob chooses a secret key $x_p$ and generates public key as $y_p \leftarrow KG_{DLP}(params_{DLP}, x_p)$.

**Signing rights Generation:** Alice chooses a random number $k_o \in \mathbb{Z}_{p-1}^*$ and computes $r_o = g^{k_o} \mod p$. Then Alice computes $\sigma_o \leftarrow OS(params_{DLP}, (k_o, r_o), x_o, ProxyID)$.

**Signing rights Verification:** Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow OV(params_{DLP}, r_o, y_o, ProxyID, \sigma_o)$.

**Proxy Key Generation:** Bob computes proxy key as $\rho_p \leftarrow PG(params_{DLP}, \sigma_o, x_p)$.

**Proxy Signature Generation:** The proxy signature on message $m$ is computed as $\sigma_p \leftarrow PS(params_{DLP}, r_o, ProxyID, \rho_p, m)$.

**Proxy Signature Verification:** The verifier accepts the proxy signature if and only if $Accepts \leftarrow PV(params_{DLP}, r_o, ProxyID, y_o, \omega, m, \sigma_p)$.

**Security:** The security of the key issuance protocol is based on the hardness of the DLP. However, the scheme has three weaknesses. Firstly, (Proxy signer’s deniability), i.e., the proxy signer can deny his signature creation later because a proxy signature does not contain any authentic information of proxy signer. Secondly, (Proxy signer’s misuse), i.e., the proxy signer gets a proxy key pair $(x_p, y_p)$ from original signer. He can deny his signature by showing that the proxy signature is created by the original signer with the name of him. Thirdly, (Original signer’s misuse), i.e, the original signer sends the delegation of signing rights to a proxy signer without any agreement or warrant. In this case, the original signer can argue that the proxy signature is not valid for the message. Further, the schemes in [Lee et al., 2001a], [Lee et al., 2003] showed that Petersen and Horster’s scheme is insecure.

A.5.1.5 Sun, Lee and Hwang [Sun et al., 1999]

Sun et al. [Sun et al., 1999] showed some weaknesses of Zhang’s [Zhang, 1997a] and Kim et al.’s [Kim et al., 1997] threshold proxy signature schemes. Subsequently, Sun et al. proposed a $(t, n)$ threshold proxy signature scheme based on Zhang’s scheme.
Assumption: DLP is hard.

Protocol:
Alice chooses a secret key $x_o$ and generates public key as $y_o \leftarrow KG_{DLP}(\text{paramsDLP}, x_o)$.
Assume that there is a group of $n$ proxy signers $p_i, i = 1, \ldots, n$.
Proxy Generation: Alice picks $k_o \in \mathbb{Z}_{p-1}$, computes $R = g^{k_o} \mod p$, and broadcasts $R$. Then, each proxy signer randomly selects $\alpha_i \in \mathbb{Z}_{p-1}$, computes $y_{pi} = g^{\alpha_i}R \mod p$.
Alice computes $\hat{R} = \prod_{i=1}^{n} y_{pi} \mod p$, and $\hat{s} = n^{-1}h(\hat{R}, PGID)x_o + k \mod p - 1$ and broadcasts $\hat{s}$, where $PGID$ is the proxy group identity that records the proxy status, the event mark of the proxy share generation $(t, n)$, the expiration time of the delegation of signing power, the identities of original signer and proxy signers. After validating $\hat{s}$, each proxy signer performs a $(t, n)$ verifiable threshold secret sharing scheme [Pedersen, 1991], and acts as a dealer to distribute proxy sub-shares to other $n - 1$ proxy signers for generating their valid proxy shares. Each proxy signer $p_t$ selects an $(t - 1)$-degree polynomial $f_i(x) = s_i + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1} \mod p - 1$, where $s_i = \hat{s} + \alpha_i + x_ih(\hat{R}, PGID) \mod p - 1$. Then $p_t$ sends the proxy sub-share $f_i(j)$ to proxy signer $p_j$ (for $1 \leq j \leq n$ and $j \neq i$) via a secure channel.

After validating all $f_j(i)$, $p_t$ computes $x'_t = \sum_{j=1}^{n} f_j(i) \mod p - 1$ as his proxy share.

Let $f(x) = \sum_{j=1}^{n} f_j(x) \mod p - 1$. This proxy share can be written as $x'_t = f(i)$ and will be used for generating proxy signatures. The shared secret key is regarded as $f(0) = \hat{n}s + \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} x_ih(\hat{R}, PGID) = \sum_{i=1}^{n} (\alpha_i + k_o) + \sum_{i=0}^{n} x_ih(\hat{R}, PGID) \mod p - 1$.

Proxy Signature Generation: Each participant proxy signer $p_t$ performs a $(t, t)$ verifiable secret sharing scheme by randomly choosing a $(t - 1)$-degree polynomial $f'_i(x) = \sum_{j=0}^{t-1} a'_{i,j}x^j \mod p - 1$ and broadcasts $c'_{i,j} = g^{a'_{i,j}} \mod p$ for $j = 0, 1, \ldots, t - 1$. Then $p_t$ computes $f'_i(j)$ and sends it to $p_j$ via a secure channel for $1 \leq j \leq n$ and $j \neq i$. Moreover, each participant proxy signer $p_t$ can get $x'' = f(i) = \sum_{j=1}^{t} f'_j(i) \mod p - 1$, where $f'(x) = \sum_{j=1}^{t} f_j(x) \mod p - 1$ and $Y = \prod_{k=1}^{t} c'_{k,0} \mod p$. Finally, each $p_t$ computes and broadcasts $T_i = x'_i \cdot m + x''_i \cdot Y \mod p - 1$.

After validation $T_i$, each $p_t$ computes $T = f(0) \cdot m + f'(0) \cdot Y \mod p - 1$ by applying Lagrange’s interpolating polynomial, where $m$ is the message. The proxy signature
on message $m$ is the tuple $(\hat{R}, PGID, Y, T)$.

Proxy Signature Verification: The verifier accepts the proxy signature if and only if

$$g^{T} = ((y_{o} \prod_{i=1}^{n} y_{i})^{h(\hat{R}, PGID)} \hat{R}^{h(m)} Y)^{Y} \mod p.$$  

**Security:** Hsu et al. [Hsu et al., 2003] and Shao [Shao, 2004] noticed that Sun et al.’s scheme is not secure. It lacks coalition attack.

### A.5.1.6 Lee, Kim and Kim [Lee et al., 2001b]

Proposed a scheme, where a mobile agent is constructed using non-designated proxy signature which represents both the original signer’s (customer) and the proxy signer’s (remote server) signatures. The authors of the paper provide Schnorr-based and RSA-based constructions of secure mobile agent. Here, we give Schnorr-based construction, the RSA-based construction is given in the next section.

**Assumption:** DLP is hard.

**Protocol:**

Alice chooses a secret key $x_{o}$ and generates public key as $y_{o} \leftarrow KG_{DLP}(paramsDLP, x_{o})$.

Bob chooses a secret key $x_{p}$ and generates public key as $y_{p} \leftarrow KG_{DLP}(paramsDLP, x_{p})$.

**Signing rights Generation:** Alice chooses a random number $k_{o} \in \mathbb{Z}_{p-1}^{*}$ and computes $r_{o} = g^{k_{o}} \mod p$. Then Alice computes $\sigma_{o} \leftarrow OS(paramsDLP, (k_{o}, r_{o}), x_{o}, \omega)$.

**Signing rights Verification:** Bob accepts $\sigma_{o}$ if and only if

$$Accepts \leftarrow OV(paramsDLP, r_{o}, y_{o}, \omega, \sigma_{o}).$$

**Proxy Key Generation:** Bob computes proxy key as $\rho_{p} \leftarrow PG(paramsDLP, \sigma_{o}, x_{p}, \omega)$.

**Proxy Signature Generation:** The proxy signature on message $m$ is computed as

$$\sigma_{p} \leftarrow PS(paramsDLP, \rho_{p}, m).$$

**Proxy Signature Verification:** The verifier accepts the proxy signature if and only if

$$Accepts \leftarrow PV(paramsDLP, r_{o}, y_{o}, \omega, m, \sigma_{p}).$$

**Security:** The scheme is insecure [Wang et al., 2003].

### A.5.1.7 Boldyreva, Palacio and Warinschi [Boldyreva et al., 2003]

Presented a formal definition and security notion for proxy signature, i.e., the existential unforgeability against adaptive chosen-message attacks [Goldwasser et al., 1988].

At the same time, they proposed a provable secure scheme, called triple Schnorr proxy signature scheme, which is a modified version of the scheme [Kim et al., 1997].
Assumption: DLP is hard.

Protocol:
Alice chooses a secret key $x_o$ and generates public key as $y_o \leftarrow KG_{DLP}(paramsDLP, x_o)$.
Bob chooses a secret key $x_p$ and generates public key as $y_p \leftarrow KG_{DLP}(paramsDLP, x_p)$.

Signing rights Generation: Alice chooses a random number $k_o \in \mathbb{Z}_{p-1}^*$ and computes
$$r_o = g^{k_o} \mod p.$$ Then, Alice computes $\sigma_o \leftarrow OS(paramsDLP, (k_o, r_o), y_o, y_p, x_o, \omega)$.

Signing rights Verification: Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow OV(paramsDLP, r_o, y_o, y_p, \omega, \sigma_o)$.

Proxy Key Generation: Bob computes proxy key as $\rho_p \leftarrow PG(paramsDLP, y_o, y_p, r_o, \sigma_o, x_p, \omega)$.

Proxy Signature Generation: The proxy signature on message $m$ is computed as
$$\sigma_p \leftarrow PS(paramsDLP, y_o, y_p, \omega, \rho_p, m).$$

Proxy Signature Verification: The verifier accepts the proxy signature if and only if $Accepts \leftarrow PV(paramsDLP, r_o, y_o, y_p, \omega, m, \sigma_p)$.

Proxy Identification: The proxy identification algorithm is ascertained by $ID((\omega, y_o, y'_p, \sigma)) = y'_p$.

Security: The scheme is proven secure against existential forgery on adaptively chosen message attacks under the random oracle model [Bellare and Rogaway, 1993].

A.5.1.8 Li, Tzeng and Hwang [Li et al., 2003]

Proposed a generalized version of $(t_1/n_1 - t_2/n_2)$ proxy signature. The $(t_1/n_1 - t_2/n_2)$ proxy signature scheme allows the original group of original signers to delegate their signing capability to a designated proxy group. The proxy group of proxy signers can cooperatively generate the proxy signature on behalf of the original group. Any verifier can verify the proxy signature on the message with the knowledge of the identities of the actual original signers and the actual proxy signers.

Assumption: DLP is hard.

Protocol:
Let the scheme consists of $n_1$ original signers and $n_2$ proxy signers.
For $i = 1, 2, \cdots, n_1$, the original signer chooses a secret key $x_{o_i}$ and generates public key as $y_{o,i} \leftarrow KG_{DLP}(paramsDLP, x_{o_i})$.
For $j = 1, 2, \cdots, n_2$, the proxy signer chooses a secret key $x_{p_j}$ and generates public key $y_{p_j} \leftarrow KG_{DLP}(paramsDLP, x_{p_j})$.
SIGNING RIGHTS GENERATION: For \( i = 1, 2, \cdots, t_1 (< n_1) \), the original signer chooses a random number \( k_o_i \in \mathbb{Z}_{p-1}^* \) and computes \( r_o_i = g^{k_o_i} \mod p \). Then, the original signer computes \( \sigma_o_i = \text{OS}((p\text{ar}msDLP, (k_o_i, r_o_i, x_o_i, \omega)) \). A designated clerk (any one of the original signers) verifies the individual proxy shares as whether

\[
\text{Accepts} \leftarrow \text{OV}(\text{paramsDLP}, r_o_i, y_o_i, \omega, \sigma_o_i).
\]

If it holds, the clerk combines the individual proxy shares as \( \sigma_o = \sum_{i=1}^{t_1} \sigma_o_i \mod p - 1 \). The final proxy share is the tuple \((\omega, K, \sigma_o)\), where \( K = \prod_{i=1}^{t_1} r_o_i \).

SIGNING RIGHTS VERIFICATION: Bob accepts \( \sigma_o \) if and only if

\[
\text{Accepts} \leftarrow \text{OV}(\text{paramsDLP}, K, y_o, \omega, \sigma_o).
\]

PROXY SIGNATURE GENERATION: Each proxy signer selects a random integer \( k_p_j \in \mathbb{Z}_{p-1}^* \), computes \( r_p_j = g^{k_p_j} \mod p \), and broadcasts \( r_p_j \). Then, each proxy signer computes \( R = \prod_{j=1}^{t_2} r_p_j \mod p \) and \( \sigma_p_j = k_p_j R + (\sigma_o \cdot x_o_i y_o_i) h((m, R, \text{ProxyID})) \mod p - 1 \), where \( t_2 \) is the threshold value of the proxy signers group.

A designated clerk verifies the individual proxy signature as whether

\[
g^{\sigma_p_j} = g^{(K \prod_{i=1}^{t_1} y_o_i h(\omega, K)} t_2 - 1 y_p_j) h((m, R, \text{ProxyID})) \mod p.
\]

If it does, the clerk combines the individual proxy signature of \( m \) as \( \sigma = \sum_{j=1}^{t_2} \sigma_p_j \mod p - 1 \). The proxy signature of \( m \) is \((\omega, K, m, R, \sigma)\).

PROXY SIGNATURE VERIFICATION: A verifier checks the validity of the proxy signature on the message \( m \) whether \( g^\sigma = R^R (K \prod_{i=1}^{t_1} y_o_i h(\omega, K) \prod_{j=1}^{t_2} y_p_j) h((m, R, \text{ProxyID})) \mod p \).

SECURITY: The scheme is assumed to be secure against the hardness of DLP.

A.5.1.9 Wang and Pieprzyk [Wang and Pieprzyk, 2003]

Proposed one-time proxy signature schemes using one-way hash function and oblivious transfer, respectively with different security properties.

ASSUMPTION: Hash function is collision-resistant.

PROTOCOL:

Proxy Generation: Alice randomly chooses an \( n \times t \) array \( A = (s_{ij})_{n \times t} \) as the private key. Each row holds \( t \) secret keys of an instance of the \((t, k)\) one-time signature. The public key is \( V = (v_{ij})_{n \times t} \), where \( v_{ij} = h(s_{ij}); \) here, \( h \) is the one-way hash function.

Then, Alice and Bob execute an Oblivious Transfer [Mu et al., 2002] \( m \)-out-
\( n \) protocol. At the end of the protocol, Bob learns one row (say, \( i^{th} \)) from \( A \), that is \((s_{i1}, \cdots, s_{it})\), as her private key. Alice has no information about the index \( i \).

Bob applies \( h \) to \((s_{i1}, \cdots, s_{it})\) and compares the results with the \( i^{th} \) row of public array.
APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

V. If the check fails, Bob exits the scheme and complains to Alice.

Proxy Signature Generation: To sign a message $M$, Bob uses $(s_{i1}, \cdots, s_{it})$ as her private key of the one-time signature. Bob first computes $S(M) = \{j_1, \cdots, j_k\} \subseteq \{1, \cdots, t\}$ and then reveals the signature $\sigma = \{(s_{ij_1}, \cdots, s_{ij_k}), i\}$.

Proxy Signature Verification: To verify the signature $\{(s_{ij_1}, \cdots, s_{ij_k}), i\}$ on a message $M$, the verifier interprets $M$ as an integer between 0 and $2^b - 1$ and computes $\{j_1, \cdots, j_k\}$ as the $M$th $k$-element subset of $\{j_1, \cdots, j_k\}$. Finally, the verifier checks whether $h(s_{ij_1}) = v_{j_1}, \cdots, h(s_{ij_k}) = v_{j_k}$.

Security: Security of the scheme is based on collision-resistant hash function. The Oblivious Transfer protocol provides unconditional security for the proxy signer, in which the probability of success of the original signer to cheat the proxy signer is $\frac{1}{n}$. The signature generated by the proxy signer are indistinguishable from those created by the original signer.

A.5.1.10 Malkin, Obana and Yung [Malkin et al., 2004]

Presented a formal model for fully hierarchical proxy signatures with warrant that supported chains of several levels of delegation.

Assumption: DLP is hard.

The Protocol
- The signers’ selects secret key $x_i$ and gets public key $y_i \leftarrow KG_{\text{DLP}}(\text{paramsDLP}, x_i)$.

Signing rights Generation: It takes public keys of a designator $y_{i_{L-1}}$ and a proxy signer $y_{i_L}$, the signing key of which the designator delegates its signing right (i.e., the signing key is either a signing key $x_{i_{L-1}}$ or a proxy signing key $x_{i_o \cdots i_{L-1}}$ depending on whether $i_{L-1}$ is original signer or proxy signer), a warrant up to previous delegation $W_{L-1}$ and a warrant $\omega_L$ set in current delegation as inputs; outputs delegation rights.

Proxy Key Generation: It takes public keys of a designator $y_{i_{L-1}}$ and a proxy signer $y_{i_L}$, the secret key of the proxy signer $x_{i_{L}}$ as inputs and outputs a proxy signing key $x_{i_o \cdots i_{L}}$ and a warrant $\omega$.

Proxy Signature Generation: The proxy signing algorithm, which takes a proxy signing key $x_{i_o \cdots i_{L}}$, a message $m$ and a warrant $\omega$ as input, outputs a proxy signature.

Proxy Signature Verification: The proxy verification algorithm, which takes a public key $y_{i_o}$ of the original designator, a message $m$, a warrant $\omega$ and a proxy signature as input, outputs accept or reject.

Proxy Identification: The proxy identification algorithm, which takes a warrant $\omega$ and
APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

A proxy signature as input, outputs a list of identity (i.e., public key) \( y_i \) in the delegation chain.

**Security:** If the DLP is hard, the proposed scheme is secure in the random oracle model [Bellare and Rogaway, 1993].

### A.5.1.11 Herranz and Saez [Herranz and Saez, 2004]

Proposed distributed proxy signature schemes. The work extended the security definitions in [Boldyreva et al., 2003] to the scenario of fully distributed proxy signature schemes.

**Assumption:** DLP is hard.

**The Protocol**

**Generation of Keys:** Let \( E = \{P^{(1)}, P^{(2)}, \ldots, P^{(n)}\} \) be a distributed entity formed by \( n \) participants. There is an access structure \( \Gamma \subset 2^E \), which is formed by those subsets of participants which are authorized to perform the secret task. The access structure must be monotone increasing; that is, if \( A_1 \in \Gamma \) is authorized, and \( A_1 \subset A_2 \subset E \), then \( A_2 \) must be authorized. The joint generation of discrete logarithm keys is as follows:

Each participant \( P^{(l)} \in E \) obtains a secret value \( x^{(l)} \in \mathbb{Z}_{p-1} \). These values \( \{x^{(l)}\}_{P^{(l)} \in E} \) form a sharing of the secret key \( x \in \mathbb{Z}_{p-1} \), according to some linear secret sharing scheme realizing the access structure \( \Gamma \). The corresponding public key \( y = g^x \mod p \) is made public, along with other values (commitments) which ensure the robustness of the protocol. The execution of this protocol is denoted as \( Jo-DL-KG(E, \Gamma) = (y, \{x^{(l)}\}_{P^{(l)} \in E}) \).

The fully distributed triple Schnorr proxy signature scheme is generated in a similar line of the Boldyreva et al.’s scheme [Boldyreva et al., 2003].

**Security:** If the DLP is hard, the proposed scheme is secure in the random oracle model [Bellare and Rogaway, 1993].
A.5.2 RSA-Based Proxy Signature

Conventions and Notation for RSA based schemes

<table>
<thead>
<tr>
<th>Alice</th>
<th>Original signer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Proxy signer</td>
</tr>
<tr>
<td>$N_o, N_p$</td>
<td>RSA Modulus of Alice and Bob, respectively</td>
</tr>
<tr>
<td>$y_o$</td>
<td>Public key of Alice, where $1 &lt; y_o &lt; \phi(N_o)$</td>
</tr>
<tr>
<td>$y_p$</td>
<td>Public key of Bob, where $1 &lt; y_p &lt; \phi(N_p)$</td>
</tr>
<tr>
<td>$x_o$</td>
<td>Secret key of Alice, where $x_oy_o \equiv 1 \mod \phi(N_o)$</td>
</tr>
<tr>
<td>$x_p$</td>
<td>Secret key of Bob, where $x_py_p \equiv 1 \mod \phi(N_p)$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>A warrant</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>A collision-resistant one-way hash function.</td>
</tr>
</tbody>
</table>

A.5.2.1 Okamoto, Tada and Okamoto [Okamoto et al., 1999]

Proposed a proxy signature scheme that diminishes the computational cost and the memory during execution and is suitable for the application of smart cards.

Assumption: IFP is hard and smart card is tamper resistant.

Protocol:
Alice chooses a public key $y_o$ and generates secret key as $x_o \leftarrow KG_{RSA}(paramsRSA, y_o)$.

Signing rights Generation: Alice computes $\sigma_o \leftarrow OS(paramsRSA, x_o, \omega)$.

Signing rights Verification: Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow OV(paramsRSA, y_o, \omega, \sigma_o)$.

Proxy Signature Generation: To sign a message $m$, Bob generates a random number $k_p \in \mathbb{Z}_N^*$, and computes

- $r = g^{k_p h(m)} \sigma_o \mod N_o$
- $s = g^{-y_ox_p} \mod N_o$

The proxy signature of message $m$ is the tuple $(m, (r, s), I_p)$.

Proxy Signature Verification: The verifier checks whether $(ID_p, \omega) = h(I_p)r^{y_oh(m)} \mod N_o$. If it does, the verifier accepts it as a valid proxy signature. Otherwise, rejects it.

Security: The security of the protocol is based on the hardness of breaking RSA signature scheme. However, the scheme has a weak security as it is designed as a proxy-unprotected scheme, where Alice can frame Bob by signing the message and later claim that Bob has signed the message [Das et al., 2004b].
A.5.2.2 Lee, Kim and Kim [Lee et al., 2001b]

A mobile agent is constructed in the scheme using non-designated proxy signature which represents both the original signer’s (customer) and the proxy signer’s (remote server) signatures.

**Assumption:** IFP is hard.

**Protocol:**

Alice chooses a public key \( y_o \) and generates secret key as \( x_o \leftarrow KG_{RSA}(paramsRSA, y_o) \).

The mobile agent (proxy signer) chooses a public key \( y_p \) and generates secret key as \( x_p \leftarrow KG_{RSA}(paramsRSA, y_p) \).

**Signing rights Generation:** Alice computes \( \sigma_o \leftarrow OS(paramsRSA, x_o, (AliceID, req), req_o) \) where \( req_o \) is the customer requirements for purchase such as price range, date, delivery requirements etc.

**Signing rights Verification:** The mobile agent accepts \( \sigma_o \) if and only if

\[
\text{Accepts} \leftarrow OV(paramsRSA, y_o, (AliceID, req), \sigma_o). \]

**Proxy Signature Generation:** Let \( BID \) be the agent’s bid information which conforms to \( req_o \). The agent (server) tries to sell the product to Alice. The server computes

\[
x = h(AliceID, req, AgentID, BID)x_p \mod N_p, \quad y = h(AliceID, req)x \mod N_o, \quad z = \sigma_o^x \mod N_o.
\]

Then the server sends the tuple \((AliceID, req, AgentID, BID, x, y, z)\) to the mobile agent and the agent will get back to Alice with this tuple as a receipt of the purchase.

**Proxy Signature Verification:** When Alice receives \((AliceID, req, AgentID, BID, x, y, z)\) from the mobile agent, she can verify the validity of the purchase by the following:

- Whether \( h(AliceID, req, AgentID, BID) = x_p \mod N_p \).
- Whether \( y = h(AliceID, req)x \mod N_o \).
- Whether \( y = z^y \mod N_o \).
- Whether \( BID \in \{req\} \).

**Security:** Wang et al. [Wang et al., 2003] showed that the scheme is insecure.

A.5.2.3 Shao [Shao, 2003]

Proposed a proxy signature scheme based on the factoring problem, which combines the RSA signature scheme and the Guillou and Quisquater [Guillou and Quisquater, 1990] signature scheme.
Assumption: IFP hard and Guillou-Quisquater signature is secure.

Protocol:
Alice chooses a public key $y_o$ and generates secret key as $x_o \leftarrow KG_{RSA}(paramsRSA, y_o)$.
Bob chooses a public key $y_p$ and generates secret key as $x_p \leftarrow KG_{RSA}(paramsRSA, y_p)$.

Signing rights Generation: Alice computes the proxy key as $v = h(\omega, ProxyID) - x_o \mod N_o, u = \lfloor v/N_p \rfloor$ and $z = v^{y_p} \mod N_p$. The delegation is the tuple $(\omega, z, u)$.

Signing rights Verification: Bob recovers the proxy key by $v = u \times N_p + (z^{x_p} \mod N_p)$.

Proxy Signature Generation: Let $m$ be the message to be signed by Bob. Bob does the following:
- Randomly chooses an integer $t \in [1, N_o]$ and computes $r = t^{y_o} \mod N_o$.
- Compute $k = h(m, r)$ and $x = k^{x_p} \mod N_p$.
- Compute $y = tv^k \mod N_o$.

The proxy signature on message $m$ is $(m, \omega, x, y, ProxyID)$.

Proxy Signature Verification: The verifier checks the following:
- Compute $k = x^{y_p} \mod N_p$.
- Compute $r' = y^{y_p}h(\omega, ProxyID)^k \mod N_o$.
- Check whether $h(m, r') = k'$.

Security: The security of the scheme is based on Guillou-Quisquater signature scheme. However, the author does not analyze its security.

A.5.2.4 Das, Saxena and Gulati [Das et al., 2004b]

Proposed a proxy signature scheme that provides effective proxy revocation mechanism.

Assumption: IFP is hard.

Protocol:
In addition to Alice and Bob, a trusted server (TS) is actively participated in the scheme for time stamp issuance.

Alice chooses a public key $y_o$ and generates secret key as $x_o \leftarrow KG_{RSA}(paramsRSA, y_o)$.
Bob chooses a public key $y_p$ and generates secret key as $x_p \leftarrow KG_{RSA}(paramsRSA, y_p)$.
TS chooses a public key $y_s$ and generates secret key as $x_s \leftarrow KG_{RSA}(paramsRSA, y_s)$.

Signing rights Generation: Alice computes $\sigma_o \leftarrow OS(paramsRSA, x_o, (\omega, y_p, y_s))$.
Signing rights Verification: Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow OV(paramsRSA, y_o, (\omega, y_p, y_s), \sigma_o)$.
Proxy Signature Generation: Let $m$ be the message to be signed. Bob requests a time stamp to the TS and sends $(R, m, e_o, e_p)$, where $R \leftarrow S(m, x_p, (\omega, e_p, e_o))$. The TS verifies whether $Accepts \leftarrow V(R, m)$. If it holds, the TS ascertain the following conditions are true before the timestamp is issued:
- Alice’s public key $y_o$ is not in the public revocation list maintained by the TS.
- $\omega$ is not expired.

Now, TS computes $T_m \leftarrow S(m, x_s, (\omega, e_o, e_p, T))$, where $T$ denotes a time stamp. Then, TS sends $(T_m, T)$ to Bob over a public channel.

On receipt a time stamp, Bob verifies whether $Accepts \leftarrow V(T_m, m)$. If it holds, Bob generates proxy signature as $\sigma_p \leftarrow PS(m, \sigma_o, x_p, (\omega, y_p, T))$, otherwise rejects the time stamp and makes another request to the TS.

The proxy signature of message $m$ is $(m, \omega, T, T_m, \sigma_p)$.

Proxy Signature Verification: The verifier checks the following to validate a proxy signature:
- Whether $Accepts \leftarrow V(T_m, m)$. If it holds, the verifier is assured that the time stamp in the signed message is correct. Otherwise, he rejects the signed message.
- Whether $h(\omega, y_p, y_s) = (\sigma_p y_p \mod N_p \oplus h(m, \omega, y_p, T)) y_p \mod N_o$. If it holds, he accepts the signed message. Otherwise, he rejects it.

Security: The scheme provides an effective proxy revocation mechanism. The scheme is secure on the assumption that IFP is hard. However, the scheme does not work when $N_p > N_o$.

### A.5.3 ECDLP Based Proxy Signature Schemes

#### A.5.3.1 Chen, Chung and Huang [Chen et al., 2004]

Proposed scheme for proxy multi-signature that increases the efficiency on applying elliptic curve cryptosystem on Sun’s scheme [Sun, 2000b].

**Assumption:** ECDLP is hard.

**Protocol**

Signing rights Generation: For each $1 \leq i \leq n$, the original signer $A_i$ selects a random number $1 \leq k_i \leq t - 1$, computes $r_i = k_i \times B = (x_{r_i}, y_{r_i})$ and $s_i = x_i \cdot x_{Q_i} \cdot h(\omega, r_i) k_i \mod t$, where $B = (x_B, y_B)$ is a point in $E(F_q)$ for a large prime $q$, the order of $B$ is assumed as $t$.

Proxy verification and alteration the proxy: For each $1 \leq i \leq n$, the proxy signer com-
puentes $U_i = (x_{Q_i} \cdot h(\omega, r_i) \mod t) \times Q_i - s_i \times B = (x_{U_i}, y_{U_i})$ using $(\omega, r_i, s_i)$. If $x_{U_i} = x_{r_i} \mod t$, the proxy signer accepts $s_i$ as a valid delegation of signing right; otherwise, he rejects it. If the proxy signer validates all $(\omega, r_i, s_i)$ in which $1 \leq i \leq n$, he then computes $d = d_p \cdot x_{Q_p} + \sum_{i=1}^{n} s_i \mod t$ as a valid proxy key.

Proxy Signature Generation: When the proxy signer signs a message $m$ for $A_1, \cdots, A_n$, he executes the signing operation of a designated signature scheme using the signing key $d$. The resulting proxy signature is the tuple $(m, \sigma(m), r_1, r_2, \cdots, r_n, \omega)$.

Proxy Signature Verification: The verifier computes the proxy public key $Q$ corresponding to the proxy key $s_p$ for verifying the proxy signature by the designated signature scheme: $Q = x_{Q_p} \times Q_p + (x_{Q_1} \cdot h(\omega, r_1) \mod t \times Q_1 + \cdots + (x_{Q_n} \cdot h(\omega, r_n) \mod t \times Q_n(r_1 + \cdots + r_n))$. With the newly generated proxy public key $Q$, the verifier confirms the validity of $\sigma(m)$ by validating the verification equation of the designated signature scheme.

Security: The authors do not consider any security model for their scheme, instead, a heuristic security analysis is given to safeguard the scheme.

### A.5.4 Pairing-Based Proxy Signature

#### Conventions and Notation for Pairings based schemes

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>A cyclic additive group, whose order is prime $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_2$</td>
<td>A cyclic multiplicative group of the same order $p$</td>
</tr>
<tr>
<td>$P$</td>
<td>A generator of $G_1$</td>
</tr>
<tr>
<td>$\hat{e}$</td>
<td>A bilinear pairing map</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>Map-to-Point</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>Collision-resistant one-way hash function</td>
</tr>
<tr>
<td>$s$</td>
<td>PKG’s master-key</td>
</tr>
<tr>
<td>$Pub_{PKG}$</td>
<td>PKG’s public key, where $Pub_{PKG} = sP$</td>
</tr>
<tr>
<td>$Q_o, s_o$</td>
<td>Original signer’s public key and secret key, respectively</td>
</tr>
<tr>
<td>$Q_p, s_p$</td>
<td>Proxy signer’s public key and secret key, respectively</td>
</tr>
</tbody>
</table>

#### A.5.4.1 Zhang and Kim [Zhang and Kim, 2003]

Presented an ID-based proxy signature scheme using partial delegation with warrant based on Hess’s ID-based signature scheme [Hess, 2002] and the construction is similar to Kim et al.’s scheme [Kim et al., 1997].

**Assumption:** WDHP is hard.
APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

Protocol:

Alice extracts her secret key $x_o$ as $x_o \leftarrow \mathcal{E}(paramsCDHP, Q_o)$.
Bob extracts his secret key $x_p$ as $x_p \leftarrow \mathcal{E}(paramsCDHP, Q_p)$.

Signing rights Generation : Alice chooses a random number $k_o \in \mathbb{Z}_p^*$ and computes $r_o = e(P, P)^{k_o}$. Then Alice computes $c_o = H_1(\omega, r_o)$ and $\sigma_o \leftarrow \mathcal{OS}(paramsCDHP, x_o, (k_o, c_o))$.

Signing rights Verification : Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow \mathcal{OV}(paramsCDHP, r_o, c_o, Q_o, \sigma_o)$.

Proxy Key Generation : Bob computes proxy key as $\rho_p \leftarrow \mathcal{PG}(paramsCDHP, \sigma_o, x_p)$.

Proxy Signature Generation : Bob chooses a random number $k_p \in \mathbb{Z}_p^*$ and computes $r_p = e(P, P)^{k_p}$. Then Bob computes $c_p = H_1(m, r_p)$ and $\sigma_p \leftarrow \mathcal{PS}(paramsCDHP, m, \rho_p, (k_p, c_p))$.

Proxy Signature Verification : The verifier accepts the proxy signature if and only if $Accepts \leftarrow \mathcal{PV}(paramsCDHP, \omega, Q_o, Q_p, c_p, \omega, m, \sigma_p)$.


A.5.4.2 Zhang, Safavi-Naini and Lin [Zhang et al., 2003]

Proposed an ID-based proxy signature based on Hess’s ID-based signature scheme.

Assumption: WDHP is hard.

Protocol:

Alice extracts her secret key $x_o$ as $x_o \leftarrow \mathcal{E}(paramsCDHP, Q_o)$.
Bob extracts his secret key $x_p$ as $x_p \leftarrow \mathcal{E}(paramsCDHP, Q_p)$.

Signing rights Generation : Alice computes $\sigma_o \leftarrow \mathcal{OS}(paramsCDHP, x_o, \omega)$.

Signing rights Verification : Bob accepts $\sigma_o$ if and only if $Accepts \leftarrow \mathcal{OV}(paramsCDHP, Q_o, \omega, \sigma_o)$.

Proxy Key Generation : Bob computes proxy key as $\rho_p \leftarrow \mathcal{PG}(paramsCDHP, \sigma_o, x_p, \omega)$.

Proxy Signature Generation : Bob chooses a random number $k_p \in \mathbb{Z}_p^*$ and computes $r_p = e(P, P)^{k_p}$. Then Bob computes $c_p = H_1(m, r_p)$ and $\sigma_p \leftarrow \mathcal{PS}(paramsCDHP, m, \rho_p, (k_p, c_p))$.

Proxy Signature Verification : The verifier accepts the proxy signature if and only if $Accepts \leftarrow \mathcal{PV}(paramsCDHP, \omega, Q_o, Q_p, c_p, \omega, m, \sigma_p)$. 
Security: The scheme is as secure as Hess’s ID-based signature scheme.

A.5.4.3 Chen, Zhang and Kim [Chen et al., 2003a]

Proposed a multi-proxy signature scheme, where Alice delegates her signing capability to \( l \) proxy signers.

Assumption: CDHP is hard.

Protocol:
Alice extracts her secret key \( x_o \) as \( x_o \leftarrow \mathcal{E}(\text{paramsCDHP}, Q_o) \).
Each proxy signer extracts his secret key \( x_{p_i} \) as \( x_{p_i} \leftarrow \mathcal{E}(\text{paramsCDHP}, \ Q_{p_i}) \).

Signing rights Generation : Alice chooses a random number \( k_o \in \mathbb{Z}_{p-1}^* \) and computes \( r_o = \hat{e}(P, P)^{k_o} \). Then Alice computes \( c_o = H_1(\omega, r_o) \) and \( \sigma_o \leftarrow \mathcal{OS}(\text{paramsCDHP}, x_o, \ (k_o, c_o)). \)

Signing rights Verification : Each proxy signer accepts \( \sigma_o \) if and only if 
\( \text{Accepts} \leftarrow \mathcal{OV}(\text{paramsCDHP}, r_o, c_o, Q_o, \sigma_o). \)

Proxy Key Generation : Each proxy signer computes proxy key as 
\( \rho_{p_i} \leftarrow \mathcal{PG}(\text{paramsCDHP}, \sigma_o, x_{p_i}). \)

Proxy Signature Generation : Each proxy signer performs the following operations to sign a message \( m \):
- Picks randomly \( k_{p_i} \in \mathbb{Z}_{p-1}^* \), computes \( r_{p_i} = \hat{e}(P, P)^{k_{p_i}} \) and broadcasts \( r_{p_i} \) to the remaining \( l-1 \) proxy signers.
- Computes \( r_p = \prod_{i=1}^{l} r_{p_i} \) and \( c_p = H_1(m, r_p), \sigma_{p_i} = c_p \rho_{p_i} + k_{p_i} P \). Then, sends \( \sigma_{p_i} \) to a designated clerk (one of the proxy signers).
- The clerk verifies the individual proxy signature and computes \( \sigma_p = \sum_{i=1}^{l} \sigma_{p_i} \). The proxy signature of message \( m \) is the tuple \( (m, \omega, r_o, c_p, \sigma_p) \).

Proxy Signature Verification : The verifier accepts the proxy signature of message \( m \) if and only if \( c_p = H_1(m, \hat{e}(\sigma_{p_i}, P)) (\hat{e}(\sum_{i=1}^{l} (Q_o + Q_{p_i}, PKG_{Pab}) H_1(\omega, r_o) \cdot r_o^{l-1} - c_p). \)

Security: The security of the scheme is based on CDHP.

A.5.4.4 Xu, Zhang and Feng [Xu et al., 2004]

Formalized a notion of security for ID-based proxy signature schemes and presented a proxy signature scheme.

Assumption: CDHP is hard.
APPENDIX A. TEN YEARS RESEARCH ON PROXY SIGNATURES: A SURVEY

Protocol:
Alice extracts her secret key $x_o$ as $x_o \leftarrow E(\text{paramsCDHP}, Q_o)$.  
Bob extracts his secret key $x_p$ as $x_p \leftarrow E(\text{paramsCDHP}, Q_p)$.  
Signing rights Generation: Alice randomly picks $k_o \in \mathbb{Z}_p^*$, computes $r_o = k_o P$ and $H_o = H_2(ID_o, \omega, r_o)$. Then she Computes $\sigma_o = x_o + k_o H_o$.  
Signing rights Verification: Bob checks whether $\hat{e}(\sigma_o, P) = \hat{e}(PKG_{Pub}, Q_o)\hat{e}(r_o, H_o)$.  
Proxy Key Generation: Bob computes proxy key as $\rho_p = x_p \sigma_o$.  
Proxy Signature Generation: To sign a message $m$, Bob does the following:  
- Chooses a random number $r_p \in \mathbb{Z}_{p-1}^*$, computes $r_p = k_p P$ and then puts $H_p = H_3(ID_p, m, r_p)$.  
- Computes $\sigma_p = \rho_p + k_p H_p$.  
The proxy signature of message $m$ is the tuple $(\omega, m, ID_p, r_o, r_p, \sigma_p)$.  
Proxy Signature Verification: The verifier accepts the proxy signature of message $m$ if and only if $\hat{e}(\rho_p, P) = \hat{e}(PKG_{Pub}, Q_o)\hat{e}(r_p, H_p)\hat{e}(r_o, H_o)$.  

Security: Security of the scheme is based on the BDHP in the random oracle model. The protocol takes large computation cost.

A.5.4.5 Zhang, Safavi-Naini and Susilo [Zhang et al., 2004]

Proposed a proxy signature scheme based on a new short signature scheme.

Assumption: CDHP is hard.

Protocol:
Alice extracts her secret key $x_o$ as $x_o \leftarrow E(\text{paramsCDHP}, Q_o)$.  
Bob extracts his secret key $x_p$ as $x_p \leftarrow E(\text{paramsCDHP}, Q_p)$.  
Signing rights Generation: Alice computes $\sigma_o = (s_o + H_1(\omega))^{-1}Q_p$.  
Signing rights Verification: Bob verifies whether $\hat{e}(H_1(\omega)P + Q_o, \sigma_o) = \hat{e}(P, Q_p)$.  
Proxy Key Generation: Bob verifies whether $\hat{e}(H_1(\omega)P + Q_o, \sigma_o) = \hat{e}(P, Q_p)$.  
Proxy Signature Generation: To sign a message $m$, Bob does the following:  
- Chooses a random number $r \in \mathbb{Z}_{p-1}^*$ and computes $U = r \cdot (H_1(\omega)P + Q_o)$.  
- Computes $t = H_2(m, U)$ and $\sigma_p = (t + r)^{-1}\rho_p$.  
The proxy signature of message $m$ is $(U, \sigma_p, \omega)$.  
Proxy Signature Verification: The verifier verifies whether $e(U + H_2(m, U)(H_1(\omega)P + Q_o), \sigma_p) = e(Q_p, Q_p)$.  

Security: Security of the scheme is based on CDHP in the random oracle model.
A.6 Concluding Remarks

In conclusion, we first provide a brief analysis of computational time required in each case. The computation time for a proxy signature falls into two parts. The first part consists of the computation time of the signing rights generation, signing rights verification and proxy key generation, which is a one-time computation and remains same for the entire delegation period. We call it as proxy generation. The second part consists of the proxy signature generation and proxy signature verification, which is required as and when a proxy signature is generated or verified. In the following, we give the computation time for proxy generation, proxy signature generation and proxy signature verification for each case, namely Table A.1 depicts computation time for DLP-based proxy signature schemes; Table A.2 depicts computation time for RSA-based proxy signature schemes; and Table A.3 gives computation time for pairing-based proxy signature schemes.

According to some of the well referred works [Barreto and Kim, 2001], [Barreto et al., 2002], [Bertoni et al., 2005], the time complexity\(^5\) for different constructions is as follows:

- RSA(1024-bit modulus) signature-exponentiation operation \(\sim 7.90\) ms
- RSA(1024-bit modulus) verification-exponentiation operation \(\sim 0.40\) ms
- DSA(1024-bit \(p\), 160-bit \(q\)) signature \(\sim 4.09\) ms
- DSA(1024-bit \(p\), 160-bit \(q\)) verification \(\sim 4.87\) ms
- \(E(F_{3^{97}})\) Tate-pairing computation \(\sim 20\) ms
- \(E(F_{3^{97}})\) Map-to-Point computation \(\sim 5.53\) ms.

The notations used in the following tables are as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_{DLP})</td>
<td>Time taken for a DLP-based signature (4.09 ms)</td>
</tr>
<tr>
<td>(V_{DLP})</td>
<td>Time taken for a DLP-based verification (4.87 ms)</td>
</tr>
<tr>
<td>(t_{es})</td>
<td>Time taken for an RSA signature-exponentiation operation (7.90 ms)</td>
</tr>
<tr>
<td>(t_{ev})</td>
<td>Time taken for an RSA verification-exponentiation operation (0.40 ms)</td>
</tr>
<tr>
<td>(P_o)</td>
<td>Time taken for a pairing operation (20 ms)</td>
</tr>
<tr>
<td>(H)</td>
<td>Time taken for Map-to-Point (5.53 ms)</td>
</tr>
</tbody>
</table>

---

\(^5\)We neglect hash computation time and simple arithmetic operation time.
DLP-based Proxy Signature Schemes

<table>
<thead>
<tr>
<th>Phases → Schemes ↓</th>
<th>Proxy Key Generation (ms)</th>
<th>Proxy Signature Generation (ms)</th>
<th>Proxy Signature Verification (ms)</th>
<th>Secure Channel</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Mambo et al., 1996]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>Yes</td>
<td>Delegation misuse</td>
</tr>
<tr>
<td>[Kim et al., 1997]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>Yes</td>
<td>Secure</td>
</tr>
<tr>
<td>[Petersen and Horster, 1997]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>Yes</td>
<td>Insecure</td>
</tr>
<tr>
<td>[Lee et al., 2001b]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>No</td>
<td>Insecure</td>
</tr>
<tr>
<td>[Boldyreva et al., 2003]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>No</td>
<td>Formalize security model</td>
</tr>
<tr>
<td>[Li et al., 2003]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>No</td>
<td>Generalized scheme</td>
</tr>
<tr>
<td>[Malkin et al., 2004]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>No</td>
<td>Multi-level trust</td>
</tr>
<tr>
<td>[Herranz and Saez, 2004]</td>
<td>$1S + 1V$ (8.96)</td>
<td>$1S$ (4.09)</td>
<td>$1V$ (4.87)</td>
<td>No</td>
<td>Distributed proxy</td>
</tr>
</tbody>
</table>

Table A.1: Computation Time : DLP-based Proxy Signatures

---

6To make uniformity, we take one original signer and one proxy signer model.
### RSA-based Proxy Signature Schemes

<table>
<thead>
<tr>
<th>Phases → Schemes ↓</th>
<th>Proxy Key Generation (ms)</th>
<th>Proxy Signature Generation (ms)</th>
<th>Proxy Signature Verification (ms)</th>
<th>Secure Channel</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Okamoto et al., 1999]</td>
<td>( t_{es} + t_{ev} ) (8.30)</td>
<td>( 2t_{es} ) (15.80)</td>
<td>( t_{es} + t_{ev} ) (8.30)</td>
<td>Yes</td>
<td>Proxy unprotected</td>
</tr>
<tr>
<td>[Lee et al., 2001b]</td>
<td>( 2t_{es} + 2t_{ev} ) (8.30)</td>
<td>( 3t_{es} ) (23.70)</td>
<td>( t_{es} + 2t_{ev} ) (8.70)</td>
<td>Yes</td>
<td>Insecure</td>
</tr>
<tr>
<td>[Shao, 2003]</td>
<td>( 2t_{es} + t_{ev} ) (16.20)</td>
<td>( t_{es} + t_{ev} ) (16.20)</td>
<td>( t_{es} + 2t_{ev} ) (8.70)</td>
<td>Yes</td>
<td>No security analysis</td>
</tr>
<tr>
<td>[Das et al., 2004a]</td>
<td>( t_{es} + t_{ev} ) (8.30)</td>
<td>( 3t_{es} + 2t_{ev} ) (24.50)</td>
<td>( 3t_{ev} ) (1.20)</td>
<td>No</td>
<td>Proxy revocation</td>
</tr>
</tbody>
</table>

Table A.2: Computation Time: RSA-based Proxy Signatures
Pairings-based Proxy Signature Schemes

<table>
<thead>
<tr>
<th>Phases → Schemes ↓</th>
<th>Proxy Key Generation (ms)</th>
<th>Proxy Signature Generation (ms)</th>
<th>Proxy Signature Verification (ms)</th>
<th>Secure Channel</th>
<th>Key Escrow</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Zhang and Kim, 2003]</td>
<td>$3P_o + 2H$ (71.06)</td>
<td>$P_o + H$ (25.53)</td>
<td>$2P_o + 2H$ (51.06)</td>
<td>No</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>[Zhang et al., 2003]</td>
<td>$2P_o + 3H$ (56.59)</td>
<td>$P_o + H$ (25.53)</td>
<td>$2P_o + 2H$ (51.06)</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>[Chen et al., 2003a] $^6$</td>
<td>$3P_o + 2H$ (71.06)</td>
<td>$P_o + H$ (25.53)</td>
<td>$2P_o + 2H$ (51.06)</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>[Xu et al., 2004]</td>
<td>$3P_o + 3H$ (76.59)</td>
<td>$H$ (5.53)</td>
<td>$5P_o + 4H$ (122.12)</td>
<td>Yes</td>
<td>Yes</td>
<td>Not efficient</td>
</tr>
<tr>
<td>[Zhang et al., 2004]</td>
<td>$2P_o + 3H$ (56.59)</td>
<td>–</td>
<td>$2P_o$ (40)</td>
<td>Yes</td>
<td>Yes</td>
<td>Efficient</td>
</tr>
</tbody>
</table>

Table A.3: Computation Time: Pairing-based Proxy Signatures

It is observed that many times, a paper typically breaks a previous scheme and proposes a new one, which someone breaks later and, in turn, proposes a new one, and so on. Most of such work, though quite important and useful, essentially provides an incremental advance to the same basic theme.

The authors tried to explore whether there are any real implementation of various proposed proxy signatures. They contacted over email individual author(s) of some of the papers, to learn whether their proposed scheme is used in real life scenario? Unfortunately, the responses from several authors indicated that they were not aware of such applications which use their scheme. Some even suggested that, since the authors work with real life banks problems, whether it will possible to try their scheme in such applications.

We believe that the actual deployment of proxy signatures is yet to start in a big way. However, as and when this happens, the research work being carried out will certainly provide practically usable implementations.

---

$^6$To make uniformity, we take one original signer and one proxy signer model.
Appendix B

Proxy Signature Scheme based on RSA

B.1 Introduction

In 1999, Okamoto et al. [Okamoto et al., 1999] proposed a proxy signature scheme, where the proxy signature and verification process are based on RSA and done by a smart card. Afterwards, Lee et al. [Lee et al., 2001b] showed that a secure mobile agent can be constructed using strong non-designated proxy signature, and they proposed RSA-based proxy signature scheme. Both the schemes [Okamoto et al., 1999] and [Lee et al., 2001b] are designed as proxy-unprotected notion, where the original signer can forge the proxy signature by signing the message and then claim that the proxy signer has signed the message. Thus, these schemes do not meet the strong unforgeability property of proxy signatures. Moreover, Wang et al. [Wang et al., 2003] showed the forgery attacks of [Lee et al., 2001b]. In 2003, Shao [Shao, 2003] proposed a proxy signature scheme based on factoring. However, Shao’s scheme does not present the security analysis of the scheme. In addition, both the schemes [Okamoto et al., 1999] and [Shao, 2003] require a secure channel to deliver the proxy key. These schemes do not provide any proxy revocation mechanism, thereby the original signer cannot revoke his delegated rights if he wants to do so, which is a practical requirement.

B.2 The Proposed Scheme [Das et al., 2004b]

We present two proxy signature schemes. Our first scheme provides proxy signatures without revocation and the second scheme is a proxy signature with revocation. In both the schemes, the signature generation and verification process are based on RSA. The schemes are divided into five phases: setup parameters, proxy key generation,
proxy key verification, proxy signature generation, and proxy signature verification.

### B.2.1 Proxy Signature without Revocation

**Setup Parameters:**

- The original signer generates RSA public-secret key pair \((e_o, d_o)\), where \(d_o\) is kept secret, \((e_o, N_o)\) is the certified public key and \(N_o\) is the product of two large safe primes.

- The proxy signer generates RSA public-secret key pair \((e_p, d_p)\), where \(d_p\) is kept secret, \((e_p, N_p)\) is the certified public key and \(N_p\) is the product of two large safe primes.

- The original signer creates a signature on a warrant \(m_w\) and gives it to the proxy signer and then the proxy signer uses it to generate a proxy key.

**Proxy Key Generation:** The original signer does the following:

- Compute \(\sigma_o = h(m_w\|e_p)^{d_o} \mod N_o\).

- Send \((\sigma_o, m_w)\) to the proxy signer over a public channel.

**Proxy Key Verification:** The proxy signer checks whether \(h(m_w\|e_p) = \sigma_o^{e_o} \mod N_o\). If it holds, the proxy signer accepts it as a valid proxy key; otherwise, rejects it.

**Proxy Signature Generation:** To sign message \(m\), the proxy signer does the following:

- Compute \(\sigma_p = (\sigma_o \oplus h(m\|m_w\|e_p))^{d_p} \mod N_p\).

- The proxy signature of message \(m\) is \((m, m_w, \sigma_p, e_o, e_p)\).

**Proxy Signature Verification:** The verifier verifies whether \(h(m_w\|e_p) = (\sigma_p^{e_o} \mod N_p \oplus h(m\|m_w\|e_p))^{e_o} \mod N_o\). If it holds, he accepts it as a valid proxy signature; otherwise, rejects it.

### B.2.2 Proxy Signature with Revocation

The proxy revocation mechanism can be categorized into two types. The first approach is to publish the public key of the original signer in a trusted server public revocation list. To validate a proxy signature, the verifier should first check whether the public key of the original signer is in the public revocation list. If the original
signer’s public key is published in the revocation list, the verifier will reject the signed message. The second approach is to use a time stamp along with the warrant in the signature generation phase, thereby enabling the verifier to validate the time of proxy signature generation on checking the validity period of warrant attached with the delegated rights. Moreover, from the time stamp-based approach, the verifier can be assured of the exact time when a proxy signature was created.

The proposed scheme comprises with four participants, namely, an original signer, a proxy signer, a trusted server (TS) and a verifier. The TS is responsible to maintain a public key revocation list. The TS will issue time stamp to the proxy signer after verifying both the revocation list and warrant.

Setup Parameters:

- The original signer generates RSA public-secret key pair \((e_o, d_o)\), where \(d_o\) is kept secret, \((e_o, N_o)\) is the certified public key and \(N_o\) is the product of two large safe primes.

- The proxy signer generates RSA public-secret key pair \((e_p, d_p)\), where \(d_p\) is kept secret, \((e_p, N_p)\) is the certified public key and \(N_p\) is the product of two large safe primes.

- The TS generates RSA public-secret key pair \((e_s, d_s)\), where \(d_s\) is kept secret, \((e_s, N_s)\) is the certified public key and \(N_s\) is the product of two large safe primes.

- The original signer creates a signature on a warrant \(m_w\) and gives it to the proxy signer and then the proxy signer uses it to generate a proxy key.

Proxy Key Generation: The original signer does the following:

- Compute \(\sigma_o = h(m_w \| e_p \| e_s)^{d_o} \mod N_o\).

- Send \((\sigma_o, m_w)\) to both proxy signer and TS over the public channel.

Proxy Key Verification: The proxy signer and TS check whether

\[ h(m_w \| e_p \| e_s) = \sigma_o^{e_o} \mod N_o. \]

If it holds, they accept it; otherwise, reject it.

Proxy Signature Generation:

- To sign message \(m\), the proxy signer first requests a time stamp to the TS. For this, the proxy signer computes \(R = h(m \| m_w \| e_o \| e_p)^{d_p} \mod N_p\), and then sends \((R, m, e_o, e_p)\) to the TS over the public channel.
The TS verifies whether \( h(m\|m_w\|e_o\|e_p) = R^{e_p} \mod N_p \). If it holds, the TS must ascertain the following conditions are true before issuing a time stamp:

- original signer’s public key \( e_o \) is not in the TS public revocation list.
- \( m_w \) is not expired.

- If it holds, the TS computes \( T_m = h(m\|m_w\|e_o\|e_p\|t)^{d_s} \mod N_s \), where \( t \) denotes a time stamp. Then TS sends \( (T_m, t) \) to the proxy signer over a public channel.

- The proxy signer verifies whether \( h(m\|m_w\|e_o\|e_p\|t) = T_m^{e_s} \mod N_s \).

- The proxy signer computes \( \sigma_p = (\sigma_o \oplus h(m\|m_w\|e_p\|t))^{d_e} \mod N_p \).

- The proxy signature on \( m \) is \( (m, m_w, \sigma_p, t, T_m, e_o, e_p) \).

Proxy Signature Verification:

- The verifier checks whether the original signer’s public key \( e_o \) is in the TS public revocation list.

- The verifier checks whether \( h(m\|m_w\|e_o\|e_p\|t) = T_m^{e_s} \mod N_s \).

- The verifier verifies whether \( h(m_w\|e_p\|e_s) = (\sigma_p^{e_p} \mod N_p \oplus h(m\|m_w\|e_p\|t))^{e_o} \mod N_o \). If these checks hold, he accepts it as a valid proxy signature; otherwise, rejects it.

### B.3 Security Analysis

In the following, we show that the proposed schemes satisfy the security features, namely, strong unforgeability, strong undeniability, strong identifiability, verifiability and prevention of misuse.

**Strong unforgeability**: The proxy signature is created with the proxy signer’s secret key \( d_p \) and delegated proxy key \( \sigma_o \). The proxy key is binding with the original signer’s secret key \( d_o \). No one (including the original signer) can construct the proxy signature without having the knowledge of the secret keys \( d_p \) and \( d_o \). Obtaining these secret keys by any other party is as difficult as breaking RSA, which is believed to be a hard problem. Moreover, the verification of \( h(m_w\|e_p) \) with the signed message prevents the dishonest party from the creation of forged proxy signatures. Therefore, any party including the original signer cannot forge a valid proxy signature and thus, the
proposed schemes satisfy the strong unforgeability property.

Strong undeniability: From a proxy signature of the proposed scheme, the involvements of both original signer and proxy signer are ascertained by
- the warrant $m_w$
- the connection of the public keys $e_p$ and $e_o$ in the verification process.

Thus, the proxy signer and the original signer cannot deny their involvement in a valid proxy signature, that is, the schemes satisfy the strong undeniability property.

Strong identifiability: The verification process of the proposed schemes requires proxy signer’s public key $e_p$ and warrant $m_w$. Any verifier can determine the identity of the proxy signer from the signed message, because the signed message is computed as $\sigma_p = (\sigma_o \oplus h(m||m_w||e_p||t))^{d_p} \mod N_p$, where $\sigma_o$ is a signed warrant by the original signer. Therefore, in the verification process any verifier can determine the identity of the proxy signer from $m_w$.

Verifiability: The proxy signatures of the proposed schemes consist of an explicit warrant $m_w$ and a signed warrant $\sigma_o$ delegated by the original signer. Any verifier can be convinced of the signers’ agreement and validity of the proxy signature of the transmitted message from $m_w$.

Prevention of misuse: Both the proxy signer and the original signer’s misuse are prevented in our schemes. The proxy signer cannot forge the delegated rights. In case of the proxy signer’s misuse, the responsibility of the proxy signer is determined from the warrant $m_w$. The original signer’s misuse is also prevented because he cannot compute a valid proxy signature against the name of the proxy signer, which is the strong unforgeability property. Furthermore, if the proxy signer’s misuse of delegated rights is noticed to the original signer, the original signer immediately revokes his public key with the help of TS.

In addition to the above security properties, the proposed scheme with revocation uses a trusted server, who verifies the validity of the original signer and the proxy signer from the request $R = h(m||m_w||e_o||e_p)^{d_p} \mod N_p$ sent by the proxy signer, and then issues the time stamp.

B.4 Conclusion

We proposed two proxy signature schemes based on RSA cryptosystems. Our first scheme does not consider proxy revocation mechanism, but it is efficient than the ex-
isting RSA-based schemes [Okamoto et al., 1999], [Lee et al., 2001b] and [Shao, 2003].
Our second scheme provides an effective proxy revocation mechanism, in which the
proxy signer takes a time stamp from the TS and signs message on behalf of the origi-
nal signer. We refer to the table A.2 for the comparative results on various RSA-based
proxy signature schemes. With the proxy revocation protocol, the proxy signer can-
not create a valid proxy signature after the expiry of the delegated rights or once the
original signer’s public key is published in the TS public revocation list. Moreover,
the time stamp in the proxy revocation protocol will assure the verifier about the
exact time when a proxy signature was created. The proposed schemes satisfy the
necessary security requirements of proxy signatures and do not require any secure
channel to deliver the proxy key, whereas, a secure channel is must for the schemes
[Okamoto et al., 1999], [Lee et al., 2001b]. The TS can remove the original signer’s
public key from the public revocation list once the delegation period has expired,
and thus the public revocation list will not grow unlimitedly.
Appendix C

Remote User Authentication based on RSA

We proposed a remote user authentication scheme using only hash function. The scheme has a few interesting features, viz. no verifier table, user-friendly password change option, prevents the distribution of login-IDs, etc. However, the construction of the scheme [Das et al., 2004a] was weak, in turn, suffers from the impersonation attacks. Subsequently, we proposed another scheme [Saxena et al., 2004] having all the merits of our previous scheme. But, in [Saxena et al., 2004], the protocol uses public key concept, in turn, costly than the scheme in [Das et al., 2004a]. In the following, we give only the modified one.

C.1 The Scheme [Saxena et al., 2004]

There are three entities in the proposed scheme, namely, the user, user’s smart card and the remote system. The scheme consists mainly of two phases - the registration phase and the authentication phase. Prior to the start of registration process, the remote system generates a RSA public-secret key pair \((e, d)\), and publishes \((N, e, g)\) while keeping the \(d\) as secret. The notations used throughout this paper are summarized as follows:

- \(U\) represents a user.
- \(ID\) denotes an identity chosen by \(U\).
- \(DID\) denotes \(U\)’s dynamic login identity.

\(^1\)Here \(N\) is product of two random large primes \(p\) and \(q\); \(e\) and \(d\) are integers \(\in Z_{\phi(N)}^*\) satisfying \(ed \equiv 1 \mod \phi(N)\), where \(\phi(.)\) is the Euler’s totient function.
- $PW$ denotes the password of $U$.
- $h(.)$ denotes a one-way hash function.

The scheme consists of registration phase, authentication phase and password change phase. The phases work as follows:

**Registration Phase**: This phase is invoked when a new user wants to register to the remote system.

R1. A new user $U_i$ sends a registration request to the remote system.

R2. $U_i$ submits his preferred identity $ID_i$ and password $PW_i$ to the remote system.

R3. Upon receiving the registration request, the remote system computes $B_i = h(PW_i) \oplus h(x)$, where $x$ is a remote system’s chosen secret number for all the registered users.

R4. The remote system personalizes a smart card with the parameters $ID_i$, $B_i$, $h(x)$, $e$, $N$, $h(.)$ and sends the smart card to $U_i$ over a secure channel.

**Authentication Phase**: This phase is executed every time the user logs into the system. The phase is further divided into login and verification phases. In the login phase, user sends a login request to the remote system. The login request comprises with a dynamic login-ID, called $DID_i$, which is dependent on the user’s password and remote system’s secret parameters. The remote system allows the user to access the system after successful verification of the login request.

**Login Phase**:

The user $U_i$ attaches the smart card to a terminal and keys $ID_i$ and $PW_i$. If $ID_i$ is identical to the one that is stored in the smart card, the smart card performs the following:

L1. Compute $B_i = h(PW_i) \oplus h(x)$. If $B_i$ is identical to the one that is stored in the smart card, proceeds to the step (L2); otherwise, terminates the operation.

L2. Compute a dynamic login-ID $DID_i = h(B_i, x, T)^e \mod N$, where $T$ is a time stamp.

L3. Send the login message $M = (DID_i, B_i, T)$ to the remote system over a public channel.

**Verification Phase**:

Let the remote system receives the login request $M$ at time $T^*(> T)$. The remote system performs the following steps to verify $M$:
APPENDIX C. REMOTE USER AUTHENTICATION BASED ON RSA

V1. Verify the validity of the time interval between $T^*$ and $T$. If $(T^*-T) \leq \Delta T$, the remote system proceeds to step (V2), where $\Delta T$ denotes the expected valid time interval for transmission delay. Otherwise, reject the login request.

V2. Check whether $h(B_i, x, T) = (DID_i)^d \mod N$. If it holds, the remote system accepts the login request; otherwise, rejects the login request.

Password Change Phase: This phase is invoked whenever a user $U_i$ wants to change his password. By invoking this phase, $U_i$ can easily change his password without taking any assistance from the remote system. The phase works as follows:

P1. $U_i$ attaches the smart card to a terminal and keys $ID_i$ and $PW_i$. If $ID_i$ is identical to the one that is stored in the smart card, proceeds to step (P2); otherwise, terminates the operation.

P2. $U_i$ submits a new password $PW_i^*$.

P3. The smart card computes $B_i^* = B_i \oplus h(PW_i) \oplus h(PW_i^*)$, which yields $B_i^* = h(PW_i^*) \oplus h(x)$.

P4. The password has been changed now with the new password $PW_i^*$ and the smart card replaces $B_i$’s value by $B_i^*$’s value.

C.2 Security Analysis

The security of the scheme is based on the one-way hash computation and the hardness of replicating a smart card. In the following, we show that the proposed scheme can resist the replay attacks, forgery attacks and stolen-verifier/insider attacks.

Replay attack: The replay attack cannot work because it fails the step (V1) of the verification phase for the time interval $(T_{new}^*-T)$, where $T_{new}^*$ denotes adversary’s computed time stamp.

Forgery Attack: A valid user login consists of $<DID_i, B_i, T>$, where $DID_i = h(B_i, x, T)^e \mod N$, $B_i = h(PW_i) \oplus h(x)$ and $T$ denotes time stamp. To forge a valid login, the adversary needs a secret parameter $x$. As $h(x)$ is stored in the user smart card, it is difficult to obtain it and thus, the adversary cannot forge a valid login.

Stolen-Verifier/Insider Attack: In many systems, the user uses a common password
to access several servers for his convenience. If the user’s login request is password-based and the remote system maintains verifier/password table for login request, an insider of a remote system could impersonate user’s login on stealing password and gets access of the other systems. In our scheme, the user’s login request is based on the user’s password as well as a secret parameter $x$ and the remote system does not maintain any verifier/password table through which an insider gets the user password and impersonate a valid login. Although, the user submits his password to the remote system during the registration process, he can change his password on invoking the password change phase. Thus, the scheme can resist stolen-verifier/insider attack.

C.3 Conclusion

We proposed a dynamic login-ID based remote user authentication scheme with smart cards. The scheme prevents the adversary from forged login-ID attacks by employing a dynamic login-ID in every login session. The use of dynamic login-ID with smart cards controls and manages the registered users and their access to the remote system. The scheme provides a flexible password change option, where users can change their passwords any time without any assistance of remote system.