2011

Contribution to signature and identification schemes

Pairat Thorncharoensri
University of Wollongong

Contribution to Signature and Identification Schemes

A thesis submitted in fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Pairat Thorncharoensri

School of Computer Science and Software Engineering
March 2011

Dedicated to

My family

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Pairat Thorncharoensri
March 29, 2011

Abstract

In this thesis, we provide contributions to signature schemes and identification schemes in four different ways.

First, we make contributions to universal designated verifier signatures. We propose the notion of a one-time universal designated verifier signature such that the number of verifiers verifying the signature is controlled by the signer. We also propose the notion of a universal designated verifier signature with threshold-signers such that the privacy and anonymity of the signer can be achieved.

Second, we propose a new notion called "policy-controlled signatures". In this notion, a policy-controlled signature can be verified by a verifier that satisfies a policy assigned by a signer. We provide two extensions to this notion, which are "universal policy-controlled signatures" and "multi-level controlled signatures". Universal policy-controlled signatures allow a party called "a policy signer" to apply a policy on a signature on a particular message such that only a verifier that satisfies this policy can verify this policy-controlled signature. In practice, some policies can be simply represented by a level of the security, for example, "POLICY= more than the fifth level of security". From the above idea, a definition of multi-level controlled signatures is introduced. It allows a signer to eliminate the unnecessary chain of attributes in the policy and simply assign the level of security as a policy instead. Hence, the size of the policy remains constant.

Next, a new notion called "fair multi-signatures" is proposed. A multi-signature allows a group of parties to engage in an interactive protocol in order to generate a joint signature on an agreement. If all the signers follow the protocol honestly, then a multi-signature is generated and distributed fairly. However, if a dishonest signer refuses to complete his part in the protocol, but he has already obtained the other parties' contributions, then the honest signers cannot obtain a multi-signature and yet the dishonest signer can generate a multi-signature. Our notion of fair multi-signatures ensures that if the protocol is completed, then every signer involved in

the signing protocol can output a multi-signature. Meanwhile, if the protocol is not completed, then none of the signers involved in the signing protocol can output a multi-signature.

Finally, in modern communications, the public becomes aware of privacy issues. Some identification systems provide privacy for users, especially those that are based on zero knowledge proof. However, a malicious user may take advantage of privacy to deny his malicious acts. Hence, we propose a new notion called "escrowed deniable identification schemes". In this notion, a trusted party is introduced to act as a transaction opener such that it can generate evidence of the conversation from the deniable transcript generated during the interaction between a prover and a verifier. In an identification scheme, the major concern about security is impersonation. The strongest type of attack against identification schemes is the reset attack. In this thesis, we provide an identity-based identification scheme secure against reset attack. We also provide proof of our scheme which is secure against reset attack in the standard model.

Acknowledgement

I sincerely thank my supervisor, Professor Willy Susilo for his support and guidance throughout this thesis. With his vast knowledge of the cryptography area, he has guided me from the start of my research and has also provided invaluable suggestions and encouragement. I also sincerely thank my co-supervisor, Associate Professor Yi Mu, for his guidance and support during this research. From the beginning of my research career, he has encouraged me and has provided valuable comments and corrections. My thanks also go to Dr. Tianbing Xia for his advice and support during the years of my Master degree by Research. During the course of my Ph.D. studies, I had the opportunity to visit the City University of Hong Kong. I would like to thank Associate Professor Duncan S. Wong, Dr. Qiong Huang and Dr. Guomin Yang for their support and advice during my time there.

My life at the University of Wollongong has been joyful during my Ph.D. years because I have some very good friends who have been so helpful in discussions and in giving suggestions. These include Dr. Man Ho Au, Dr. Xinyi Huang, Dr. Mohammad Reza Reyhanitabar, Dr. Siamak Fayyaz Shahandashti, Dr. Rungrat Wiangsripanawan, Wei Wu, Jinguang Han, Fuchun Guo, Shekh Faisal Abdul Latip, Angela Piper, Shams Ud Din Qazi, Tsz Hon Yuen, Yi Qun Chen, Stevanus Wibowo, Shidi Xu, Ching Yu Ng and Juliet Richardson. I would like to thank Professor Fangguo Zhang for his advice and discussions during his visit to the University of Wollongong and during various conferences, as well as the anonymous referees who have reviewed the papers that are included in this thesis. I have had a wonderful experience and have enjoyed my study environment, so I would like to thank all the staff at the Centre for Computer and Information Security Research and the School of Computer Science and Software Engineering. For the financial support that I have received, I sincerely thank the Australian Postgraduate Award Industry scheme in helping me to achieve my goals. Finally, I would like to thank my family, including my wife (Agnes) So Wah Ng, my parents, my parents-in-law, my brother

and sister-in-law for all their love and encouragement. This work would not have been possible without their support.

Publications

During my PhD studies, the following presented or published papers are related to this thesis.

1. Pairat Thorncharoensri, Qiong Huang, Willy Susilo, Man Ho Au, Yi Mu, and Duncan Wong. *Escrowed deniable identification schemes*. International Journal of Security and Its Applications, 4(1):49–67, January 2010.

2. Pairat Thorncharoensri, Qiong Huang, Willy Susilo, Man Ho Au, Yi Mu, and Duncan Wong. *Escrowed deniable identification schemes*. In Dominik Ślęak, Tai hoon Kim, Wai-Chi Fang, and Kirk P. Arnett, editors, Security Technology, volume 58 of Communications in Computer and Information Science, pages 234–241. Springer, November 2009.

3. Pairat Thorncharoensri, Willy Susilo, and Yi Mu. *Identity-based identification scheme secure against concurrent-reset attacks without random oracles*. In Heung Youl Youm and Moti Yung, editors, WISA, volume 5932 of Lecture Notes in Computer Science, pages 94–108. Springer, 2009.

4. Pairat Thorncharoensri, Willy Susilo, and Yi Mu. *Policy-controlled signatures*. In Sihan Qing, Chris J. Mitchell, and Guilin Wang, editors, *ICICS*, volume 5927 of *Lecture Notes in Computer Science*, pages 91–106. Springer, 2009.

5. Pairat Thorncharoensri, Willy Susilo, and Yi Mu. *Universal designated verifier signatures with threshold-signers*. In Tsuyoshi Takagi and Masahiro Mambo, editors, IWSEC, volume 5824 of Lecture Notes in Computer Science, pages 89–109. Springer, 2009.

6. Pairat Thorncharoensri, Willy Susilo, and Yi Mu. *How to balance privacy with authenticity*. In Pil Joong Lee and Jung Hee Cheon, editors, ICISC, volume 5461 of Lecture Notes in Computer Science, pages 184–201. Springer, 2008.

Notation

$Abc.Xyz(.)$	an algorithm Abc executes a sub-algorithm Xyz
ABC or ABC-DEF	a security notion ABC or ABC-DEF
$\boldsymbol{\mathcal{ABC}}$	an oracle $\boldsymbol{\mathcal{ABC}}$
$\mathfrak{ABC}$	a list $\mathfrak{ABC}$
$\langle P_1, ..., P_n \rangle$	the execution of the n-party protocol, where P_n represents an algorithm of the party n
$\mathtt{d}$	a decision $\mathtt{d} \in \{0, 1\}$ or $\mathtt{d} \in \{Accept, Reject\}$
$\mathtt{t}$	a unit of computation time $\mathtt{t}$
$\mathbf{Pr}$	a probability $\mathbf{Pr}$
$\|\|$	the concatenation of two strings (or integers)
$\|x\|$	a bit length of a string x
$\{x_i\}$	a set of elements, where i is the index of elements in this set
PPT	a probabilistic polynomial-time algorithm
$F^{E(.)}(.)$	a PPT algorithm F privately accesses and executes another PPT algorithm E
ℓ	a security parameter
$poly(.)$	a deterministic polynomial function
$x \xleftarrow{\$} X$	the operation of picking x at random from a (finite) set X
$x \leftarrow y$	a value y is assigned to a variable x
$X(a, b) \rightarrow x$	taken values a and b as input, an algorithm X assigns the output to a variable x
$X(a, b) \rightarrow (x, y)$	taken values a and b as input, an algorithm X assigns the outputs to variables x and y
$\mathfrak{XX} \leftarrow \mathfrak{XX}(a, b)$	taken values a and b as input, a list $\mathfrak{XX}$ updates itself with the input and outputs an updated list $\mathfrak{XX}$.
$x << y$	x is much less than y

Abbreviations and Acronyms

DDH	Decisional Diffie-Hellman Problem
CDH	Computational Diffie-Hellman Problem
SDH	Strong Diffie-Hellman Problem
GDH	Gap Diffie-Hellman problem
DBDH	Decision Bilinear Diffie-Hellman
CR1	Concurrent-reset-1 Attack
CR2	Concurrent-reset-2 Attack
$\mathsf{CR1}^+$	Concurrent-reset-1-plus Attack
DVS	Designated Verifier Signature
UDVS	Universal Designated Verifier Signature
TC	Trapdoor Commitment
KH-IBI	Kurosawa-Heng Identity-based Identification
OT-UDVS	One-time Universal Designated Verifier Signature
TS-UDVS	Universal Designated Verifier Signature with Threshold-Signers
UPCS	Universal Policy-controlled Signature
PCS	Policy-controlled Signature
MLCS	Multi-level Controlled Signature
MS	Multi-signature
FMS	Fair Multi-signature
VES	Verifiable Encrypted Signatures
AS	Aggregate Signature
IBI-PA	Identity-based Identification Schemes against Impersonation under Passive Attack
IBI-CRA	Identity-based Identification Scheme against Impersonation under $\mathsf{CR1}^+$ Attack
EDID	escrowed deniable identification

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The development of advanced computers, smart phones and other digital devices has freed us from the limitations of communication and computation on the Internet. Online social networking is one of the major applications on the Internet that changes the way of socialising from a real-life social world to the digitally social world. With computers, smart phones or other digital devices, such as personal digital assistant (PDA), users can easily connect with one another at all times. They post messages and share files, photos or videos with one another through social networking sites. Information spreads faster through social networking sites than through a real-life social network. This information might be unexpectedly fallen into the wrong hands, since the digital information is easy to be copied, shared, distributed or searched through the Internet. In online social networking sites, many users usually disclose their identity or their relevant information via their profile to others. However, the aggregation of large amounts of information on the profiles of users poses new privacy risks. Hence, the awareness of user privacy and the security of the social networking and the Internet have also risen. User's privacy is currently one of the major security concerns in the social networking [AGH10, SPS10, KL10, DHP07]. In the topic of user's privacy and the security of the Internet towards the online social networking, there are many questions that have yet to be completed such as how to provide or control the privacy and the anonymity of users, how to revoke the identity of the malicious user when the dispute occurs and how to provide a fairness when users agree to sign a message together. Our aim is to address these questions.

1.1 Background and Problems

Public Key Cryptography.

Prior the modern cryptography, a secure communication between two parties can be obtained when both parties share a secret key. In the seminal paper "New Directions in Cryptography" [DH76], Diffie and Hellman introduced the concept of public key cryptography. In this concept, a secure communication between two parties can be obtained without sharing the secret key prior to the communication. They proposed that each user publishes a string called a "public key" while keeping the other related string called a "private key". In their scheme, a user A uses a public key of a user B to encrypt a message and produce a ciphertext and send it to the user B. The user B uses his private key associated with the public key to decrypt the ciphertext and obtain the message. This concept is called an asymmetric-key cryptosystem. Public key cryptography is a foundation of many cryptographic algorithms and cryptosystems. It has also been widely implemented on modern technologies around the world.

Signature Schemes.

From the concept of public key cryptography, the notion of digital signature has also been introduced in [DH76] to function as a traditional handwritten signature. A digital signature on a message is generated by a party called a "signer" to guarantee that it is the signer who signed the message such that the message cannot be altered. A private key is used to sign a message. Hence, a signature is generated. A public key, on the other hand, is used to verify the validity of the signature on a message. The properties of digital signatures include integrity, authentication and non-repudiation. Integrity protects a message from being modified by an unauthorised party. Authentication assures a verifier that a message originates from a signer. Non-repudiation prevents a signer from denying the ownership of his message and his signature.

Privacy and Anonymity of a Signer.

It is a fact that digital signatures are publicly verifiable and hence, non-repudiable. The privacy of a signer and the anonymity of a signature holder are important, too. The privacy of a signer is exposed whenever a signature on a message is generated

by the signer and released to the public. The notion of designated verifier signatures was proposed by Jakobsson, Sako and Impagliazzo [JSI96] to provide privacy for the signer. In this notion, the verification of the signer's signature is limited only to a designated verifier. However, a designated verifier signature provides the one-to-one privacy which is between a signer and a verifier. The question is how to provide the one-to-many privacy which is between a signer and the designated verifiers. In other words, how a signer assigns a set of verifiers such that only these verifiers can verify the signature on a particular message signed by the signer while others cannot do so.

Steinfeld, Bull, Wang and Pieprzyk [SBWP03] invented a new notion called a universal designated verifier signature scheme that provides privacy for a signer. The privacy of a signer is protected even though the signature is released to the public. Universal designated verifier signatures preserve the signer's authenticity, ensuring the signer's privacy, the message's integrity and the anonymity of a signature holder.

The question on universal designated verifier signatures is how to provide the anonymity and the privacy for a signer. The anonymity and privacy of a signer provide that a verifier is convinced only that a designated verifier signature is generated by one (or more) out of n signers. However, a verifier does not know who actually signed the message.

In universal designated verifier signatures, a signature holder is given a special privilege such that he can generate a designated verifier signature from a signature on a message signed by a signer and designate it to verifiers of his choice. However, as long as a signature holder holds this signature, he can generate many designated verifier signatures as he wants and designates to verifiers of his choice. Therefore, it is interesting to control the ability of the signature holder to convince only one verifier. Interestingly, there are real applications where this situation is desirable.

Fairness of Multi-signature.

One of the major applications of digital signatures is the notion of the multi-signature, which enables many co-signers to authorise a document on behalf of them. Multi-signature scheme was firstly proposed by Itakura and Nakamura [IN83]. However, a major impediment to the success of this notion relies on the need to have all the signers behave correctly in accordance with the protocol. If one of the signers does not release his signature, then all of the other signers will be disadvantaged,

while the malicious signer can obtain a valid multi-signature on behalf of the others with the knowledge that he has on his partial signature. The above problem raises the question on how to provide fairness to all signers in multi-signatures.

Identification Scheme.

Fiat and Shamir [FS86] introduced the concept of an identification scheme. This concept allows a user called the prover to prove his/her identity to another user called the verifier. The basic requirement of an identification scheme is that the other party cannot impersonate the prover. Shamir [Sha84] proposed the idea of identity-based cryptosystems, which allows a prover to select a public key that represents the identity of the prover, such as an email address. In the modern Internet, signature schemes and identification schemes are components in many Internet protocols, such as the Secure Shell (SSH) protocol. It is essential to consider the security of identity-based identification schemes against active and concurrent attacks. Many devices used in the Internet can be reset to their initial state; for example, a smart card can be reset by disconnecting and reconnecting its power source [BFGM01, CGGM00]. Based on the study by Canetti, Goldwasser, Goldreich and Micali in [CGGM00] and the security analysis in [BFGM01], reset attacks play an important part in the security of (identity-based) identification protocols. Bellare, Fischlin, Goldwasser and Micali [BFGM01] gave a formal definition of concurrent-reset attacks where an adversary has the power to reset the prover to the initial state and obtains information that leads to the associated private key before attempting to impersonate. The open question on identity-based identification schemes is how to provide an identity-based identification scheme that is secure against reset attacks.

Privacy of a Prover.

Now, we discuss about the privacy of a prover in identification schemes. The erosion of privacy in our society is increasing and hence, the issue of the preservation of privacy has become essential. An identification scheme based on zero knowledge protocol provides deniability and the privacy of a prover. However, a non-deniable property is needed for cases where there is a dispute. A question is how to provide both privacy and non-deniable property in identification scheme for the cases where there is a dispute.

1.2 Objectives of this Thesis

From the aforementioned problems in the previous section, this thesis focuses on the answer to those problems related to designated verifier signatures, universal designated verifier signatures, and their variants, multi-signatures and identification schemes. The aims of this thesis address four aspects as follows.

1. Construct a universal designated verifier signature scheme that provides the privacy and anonymity of signer. In addition, we also aim to construct a universal designated verifier signature scheme that allows a signer to control the ability of the signature holder to convince only one single verifier. Hence, the privacy of signer can be limited by the signer.

2. Provide variants of designated verifier signatures that allows a signer to assign a set of conditions or policies such that any verifier that satisfies the conditions can verify the signature on a message signed by the signer.

3. Present a new multi-signature scheme that provides fairness to all signers in multi-signatures.

4. For identification cryptosystems, our objectives are to provide an identity-based identification scheme that is secure against the reset attack and to provide the revokable privacy to the identification scheme.

1.3 Organisation of this Thesis

We provide the background knowledge of number theory, basic algebra, bilinear pairing and provable security in Chapter 2. In the same chapter, we present some basics on group and bilinear pairings. We also present some number-theoretic problems, which are commonly believed to be hard. The security of our proposed schemes is based on their hardness.

In Chapter 3, we provide some background knowledge of signature schemes and identification schemes. We briefly describe Boneh, Lynn and Shacham's signature scheme [BLS01], Boneh and Boyen's signature scheme [BB04], Waters' signature scheme [Wat05], Schnorr's identification scheme [Sch91] and a trapdoor commitment scheme [BCC88, KH06] as basic building blocks for our schemes in the following

chapters. We also provide a review of Kurosawa and Heng's identification scheme [KH04] as a comparison with our identity-based identification scheme in Chapter 7.

In Chapter 4, we introduce two new notions: "one-time universal designated verifier signatures" and "universal designated verifier signatures with threshold-signers". We describe a definition of a one-time universal designated verifier signature scheme and its security model. We present our concrete construction scheme of a one-time universal designated verifier signature scheme and its security analysis. Universal designated verifier signatures with threshold-signers provide privacy and anonymity for the signer and the signature holder. They also ensure the authenticity of a message and preserve the integrity of a message. In the same chapter, we also present a definition of a universal designated verifier signature with threshold-signers and its security model. We describe a concrete construction of a universal designated verifier signature with threshold-signers together with its security analysis.

In Chapter 5, we introduce three new notions: a "policy-controlled signature", a "universal policy-controlled signature" and a "multi-level controlled signature". We present definitions of a policy-controlled signature scheme and a universal policy-controlled signature scheme. Their security models are also provided. We propose a policy-controlled signature scheme and a universal policy-controlled signature scheme and present their security analysis. Next, we describe a definition of a multi-level controlled signature scheme and its security model. Then, we give our first proposed multi-level controlled signature scheme. In this scheme, the size of the signer's private key is constant. We also provide the security analysis of the first proposed multi-level controlled signature scheme. We present our second proposed multi-level controlled signature scheme. In this scheme, the size of the verifier's credentials is constant. We then give the security analysis of the second proposed multi-level controlled signature scheme.

In Chapter 6, a new notion for multi-signature schemes called "fair multi-signatures" is described. First, we introduce a definition of a fair multi-signature scheme and its security model. Then, we illustrate our generic construction scheme of fair multi-signatures and perform its security analysis. We present two instantiations of our generic construction scheme with their security analysis. The first instantiation scheme is constructed from Boneh et al.'s verifiable encrypted signature scheme [BGLS03]. The second instantiation scheme is constructed from Lu et al.'s verifiable encrypted signature scheme [LOS$^+$06].

In Chapter 7, we give a variant definition of the reset attacks proposed in

[BFGM01] and we name it $\mathsf{CR1}^+$ attack. We present identity-based identification schemes that are secure under passive attack and secure against $\mathsf{CR1}^+$ attack, and we also present an escrowed identification scheme. First, we describe a definition of an identity-based identification scheme and its security model. We then present our identity-based identification scheme that is secure against passive attack and $\mathsf{CR1}^+$ attack. We also give a comparison between our identity-based identification scheme and the state-of-the-art identification scheme proposed by Kurosawa and Heng [KH05]. Next, we propose an escrowed deniable identification scheme and perform its security analysis.

Finally, in Chapter 8, we conclude this thesis.

Chapter 2

Preliminaries: Mathematical Foundations

In this chapter, we provide a background of number theory, basic algebra, bilinear pairing and provable security for a better understanding of the following chapters. The aim of this chapter is to make this thesis self-contained. Hence, in the following sections, we give an explanation of some mathematic notations and problems. They are generally useful to readers who are not familiar with these topics. They are also to provide cryptographic foundations for understanding proofs of the security of the proposed schemes in this thesis. Hence, readers who are familiar with theory of cryptography may skip this chapter. We refer the reader to [MvOV97, Gol00, Gol04, Gol05] for a more in-depth knowledge on the theory of cryptography. We refer the reader to [BLS01, HMV03, BSSC05] for a more in-depth knowledge on the topics of elliptic curves and bilinear paring.

2.1 Number Theory and Basic Algebra Foundations

2.1.1 Group

In mathematics, a group is a set of objects with an operation that combines any two of its elements to form a third element. The set and its operations must satisfy four properties called group axioms, namely closure, associativity, existence of identity element and existence of inverse element. Let $\mathbb{G}$ be a non-empty set and $*$ be an operation such that $* : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$.

Definition 2.1 (Group) *A group is a set $\mathbb{G}$ together with an operation $*$ satisfying the following properties:*

Closure: $\forall a, b \in \mathbb{G} : a * b \in \mathbb{G}$.

Associativity: $\forall a, b, c \in \mathbb{G} : (a * b) * c = a * (b * c)$.

Existence of identity element: *There exists* $1_{\mathbb{G}} \in \mathbb{G}$ *(called an identity element)* : $\forall a \in \mathbb{G}, 1_{\mathbb{G}} * a = a * 1_{\mathbb{G}} = a$.

Existence of inverse element: $\forall a \in \mathbb{G}, \exists a^{-1}$ *(called an inverse element)* $\in \mathbb{G}$: $a^{-1} * a = a * a^{-1} = 1_{\mathbb{G}}$.

A group $\mathbb{G}$ is said to be Abelian group (or commutative group) if $\forall a, b \in \mathbb{G} : a * b = b * a$. If $|\mathbb{G}|$ is finite then $\mathbb{G}$ is finite. A group $\mathbb{G}$ is cyclic if there exists $g \in \mathbb{G}$ (called a generator), $\forall a \in \mathbb{G} : a = g^i$ for some integer i. Throughout this thesis, the multiplicative group notation is used. It means that for any positive integer n, a^n means a is multiplied n-times. It is also defined in the same way for the inverse element denoted as $(a^{-1})^n = a^{-n}$.

2.1.2 Bilinear Pairings

The following notation is defined in [BB04, BLS01].

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are two (multiplicative) cyclic groups of prime order p.
- g_1 (g is used where some scheme $\mathbb{G}_1$ equivalents to $\mathbb{G}_2$) is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.
- ψ is an existing isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$ (or from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_1) = g_2$).
- $\hat{e}$ is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ (or, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ where $\mathbb{G}_1$ is equivalent to $\mathbb{G}_2$).

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups as defined above and let $\mathbb{G}_T$ be a (multiplicative) cyclic group with the same order p such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map with the following properties:

1. *Bilinearity:* $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$.
2. *Non-degeneracy:* There exists $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $\hat{e}(g_1, g_2) \neq 1$.
3. *Computability:* There exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

A bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm $\mathcal{IG}$ that takes as input a security parameter ℓ and returns a uniformly random tuple $param = (p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g_1, g_2)$ of bilinear parameters, including a prime number p of size 1^ℓ, cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p, a multiplicative group $\mathbb{G}_T$ of order p, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a generator $g_1 \in \mathbb{G}_1$ and a generator $g_2 \in \mathbb{G}_2$. Note that there exists a $\psi(.)$ function mapping $\mathbb{G}_1$ to $\mathbb{G}_2$ or vice versa in one time unit. For simplicity, we assume that $\mathbb{G}_1 = \mathbb{G}_2$, and then a bilinear pairing is $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ where g is a generator of $\mathbb{G}_1$. For a group $\mathbb{G}$ of prime order p, we denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{1_\mathbb{G}\}$ where $1_\mathbb{G}$ is the identity element of the group.

2.2 Cryptographic Primitives and a Brief Review on Provable Security

In this section, we provide some number-theoretic problems and useful definition, which will be used throughout this thesis. In cryptography, the security of most cryptographic primitives relies on the intractability of some problems that believed to be hard to solve. The following mathematical problems and assumptions will be used in the rest of thesis. We will provide a brief review of the reset lemma proposed by Bellare and Palacio in [BP02]. This lemma was shown to be useful for proving the security of many cryptographic primitives including the primitives presented in this thesis. Throughout this thesis, we often refer to the random oracle model and the standard model. Hence, we will give a brief explanation of these models. Let $\mathbb{G}_1, \mathbb{G}_T$ be two cyclic (multiplicative) groups order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear pairing. The generator of $\mathbb{G}_1$ is g and $|\mathbb{G}_1| = |\mathbb{G}_T| = p$ for some large prime p. Assume that multiplication and inversion in $\mathbb{G}_1$ can be computed in a unit time.

2.2.1 Computational and Decisional Diffie-Hellman Problem

Let $a, b, c \in \mathbb{Z}_p^*$ be integers.

Definition 2.2 (Computational Diffie-Hellman (CDH) Problem.) *Given* (g, g^a, g^b) *as input, then the adversary attempts to output* $g^{ab} \in \mathbb{G}_1$. *An algorithm* $\mathcal{A}$ *can solve CDH with at least an* ϵ *advantage if*

$$Pr\left[\mathcal{A}(g, g^a, g^b) = (g^{ab})\right] \geq \epsilon$$

where the probability is over the randomly chosen a, b and the random bits consumed by $\mathcal{A}$.

Assumption 2.1 (Computational Diffie-Hellman Assumption.) *We say that the computational $(\mathtt{t}, \epsilon)$-CDH assumption holds if no PPT algorithm with time complexity $\mathtt{t}(.)$ has advantage at least ϵ in solving the CDH problem.*

Definition 2.3 (Decisional Diffie-Hellman (DDH) Problem.) *Given a randomly chosen $g \in \mathbb{G}_1$, as well as g^a, g^b, g^c, for some $a, b, c \in \mathbb{Z}_p^*$, decide whether $c \stackrel{?}{=} ab$ holds with equality.*

It is well-known that DDH problem in $\mathbb{Z}_p^*$ is easy, by performing Menezes, Okamoto and Vanstone's (MOV) reduction in [MOV93], which states the discrete logarithm problem (DLP) in $\mathbb{G}_1$ is no harder than the DLP in $\mathbb{Z}_p^*$.

2.2.2 Variants of Diffie-Hellman Problem

Definition 2.4 (q-Strong Diffie-Hellman (q-SDH) Problem.) *Given a $(q+1)$-tuple $(g, g^x, g^{x^2}, ..., g^{x^q})$ as input, output a pair $(c, g^{\frac{1}{x+c}})$ where $c \in \mathbb{Z}_p^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH if $\mathbf{Pr}\left[\mathcal{A}(g, g^x, g^{x^2}, ..., g^{x^q}) = (c, g^{\frac{1}{x+c}})\right] \geq \epsilon$, where the probability is over the random choice of $x \in \mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.*

Assumption 2.2 ($(q, \mathtt{t}, \epsilon)$-Strong Diffie-Hellman Assumption [BB04].) *We say that the $(q, \mathtt{t}, \epsilon)$-SDH assumption holds if no PPT algorithm with time complexity $\mathtt{t}(.)$ has advantage at least ϵ in solving the q-SDH problem.*

2.2.3 Gap Diffie-Hellman problem

The Gap Diffie-Hellman (GDH) problem was first proposed by Okamoto and Pointcheval [OP01]. The problem is described as follows. The generator of $\mathbb{G}_1$ and $\mathbb{G}_2$ are g_1 and g_2, respectively. Assume that multiplication and inversion in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be computed in a unit time. An efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is required for GDH group. When $\mathbb{G}_1 = \mathbb{G}_2$ and $g_1 = g_2$ one could take to be the identity map. When $\mathbb{G}_1 \neq \mathbb{G}_2$ we will need to describe explicitly an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Such a ψ function is essential for security of GDH.

Definition 2.5 (Computational co-Diffie-Hellman (co-CDH) Problem on $(\mathbb{G}_1, \mathbb{G}_2)$.) *Let $a \in \mathbb{Z}_p^*$. Given $(g_2 \in \mathbb{G}_2, A = g_2^a \in \mathbb{G}_2, B \in \mathbb{G}_1)$ as input, then the adversary attempts to output $B^a \in \mathbb{G}_1$.*

Definition 2.6 (Decisional Diffie-Hellman (co-DDH) Problem on $(\mathbb{G}_1, \mathbb{G}_2)$.) *Given $g_2 \in \mathbb{G}_2, h \in \mathbb{G}_1, g_2^a \in \mathbb{G}_2, h^b \in \mathbb{G}_1$, for some $a, b \in \mathbb{Z}_p^*$, decide whether $a \stackrel{?}{=} b$ holds with equality.*

The above problems are reduced to standard CDH and DDH when $\mathbb{G}_1 = \mathbb{G}_2$.

2.2.4 Bilinear Diffie-Hellman problem

Definition 2.7 (Decision Bilinear Diffie-Hellman (DBDH) Problem.) *Given a 4-tuple $(g, g^a, g^b, g^c \in \mathbb{G}_1)$ and a random integer $Z \in \mathbb{G}_T$ as input, decide whether or not $Z = \hat{e}(g, g)^{abc}$. An algorithm $\mathcal{A}$ is said to $(\mathtt{t}, \epsilon')$ solves the DBDH problem in $\mathbb{G}_1, \mathbb{G}_T$ if $\mathcal{A}$ runs in time $\mathtt{t}$ and*

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^d) = 1]| \geq \epsilon',$$

where the probability is taken over the random choices of $a, b, c, d \in \mathbb{Z}_p$, $g \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{A}$.

Assumption 2.3 (Decision Bilinear Diffie-Hellman Assumption.) *We say that the $(\mathtt{t}, \epsilon)$-DBDH assumption in $\mathbb{G}_1, \mathbb{G}_T$ holds if no PPT algorithm with time complexity $\mathtt{t}(.)$ has advantage at least ϵ in solving the DBDH problem.*

2.2.5 One-Way Pairing problem

The inverting pairing problem is when the pairing function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a one-way bilinear function. A pairing is a one-way function when, given $g \in \mathbb{G}_1$ and $y \in \mathbb{G}_T$, it is hard to invert the pairing; that is, to find an element $h \in \mathbb{G}_1$ such that $\hat{e}(g, h) = y$.

Definition 2.8 ($(\mathtt{t}, \epsilon)$-One-Way Pairing (OWP).) *Given $(g \in \mathbb{G}_1, Z = \hat{e}(g, g)^a \in \mathbb{G}_T)$ as input, then the adversary attempts to output $g^a \in \mathbb{G}_1$ with at most running time $\mathtt{t}$. An algorithm $\mathcal{A}$ can solve $(\mathtt{t}, \epsilon)$-OWP with at least an ϵ advantage if*

$$Pr\left[\hat{e}(g, \mathcal{A}(g, Z)) = Z\right] \geq \epsilon$$

where the probability is over the randomly chosen a and the random bits consumed by $\mathcal{A}$.

Assumption 2.4 ($(\mathtt{t},\epsilon)$-One-Way Pairing Assumption.) *We say that the computational $(\mathtt{t},\epsilon)$-CDH assumption holds if no $\mathtt{t}$-algorithm has advantage at least ϵ in solving the One-Way Pairing problem.*

2.2.6 Reset Lemma

Reset lemma is a lemma proposed by Bellare and Palacio [BP02]. This lemma is used in security proof of many schemes and it is used later at the time of exploiting proof in Chapter 7. More precisely, this lemma provides an upper bound of the probability that a cheating prover $\widetilde{P}$ can convince the verifier V to accept a certain experiment that resets the prover to obtain two accepting conversation transcripts.

The conversation transcripts refer to a canonical (three-move) protocol. Let V be the verifier in a canonical protocol and given η as input. V executes the pair of algorithm $(ChSet_V, DEC_V)$, which defines the challenge set algorithm and the deterministic decision predicate algorithm. Let P be the prover in a canonical protocol. P inputs with $(\xi, \$)$. P executes the pair of algorithm (CMT_P, RSP_P), which defines the commitment generator algorithm and the response generator algorithm. Let $\mathtt{St}$ be some state of information. Let $\$$ be a random tape. The prover's first message called a commitment is C_{mt} which is generated by executing $CMT_P(\mathtt{St})$. The verifier selects a challenge Ch uniformly at random from $ChSet_V$ and send it to the prover. Upon receiving a response R_{sp} from the prover P generated by executing $RSP_P(Ch, \mathtt{St})$, the verifier applies $DEC_V(C_{mt}, Ch, R_{sp})$ to compute a decision $\mathtt{d} \in \{Accept, Reject\}$.

Lemma 2.1 *(Reset Lemma [BP02].)*
Let $acc(\xi,\eta)$ be the probability that V accepts (ξ,η) in its interaction with P and returns Accept in the following experiment.

> *Choose random tape $\$$ for P;* $\mathtt{St} \leftarrow (\xi, \$)$; $(C_{mt}, \mathtt{St}) \leftarrow CMT_P(\mathtt{St})$.
>
> $Ch \xleftarrow{\$} ChSet_V$; $(R_{sp}, \mathtt{St}) \leftarrow RSP_P(Ch, \mathtt{St})$; $\mathtt{d} \leftarrow DEC_V(C_{mt}, Ch, R_{sp})$.
>
> *Return* $\mathtt{d}$.

Let $res(\xi,\eta)$ be the probability that V accepts (ξ,η) in its interaction with P and the following reset experiment return 1.

> *Choose random tape $\$$ for P;* $\mathtt{St} \leftarrow (\xi, \$)$; $(C_{mt}, \mathtt{St}) \leftarrow CMT_P(\mathtt{St})$.
>
> $Ch_1 \xleftarrow{\$} ChSet_V$; $(R_{sp_1}, \mathtt{St}) \leftarrow RSP_P(Ch_1, \mathtt{St})$; $\mathtt{d}_1 \leftarrow DEC_V(C_{mt}, Ch_1, R_{sp_1})$.

$Ch_2 \xleftarrow{\$} ChSet_V$; $(R_{sp_2}, \mathtt{St}) \leftarrow RSP_P(Ch_2, \mathtt{St})$; $\mathtt{d}_2 \leftarrow DEC_V(C_{mt}, Ch_2, R_{sp_2})$.

If $(\mathtt{d}_1 = Accept \bigwedge \mathtt{d}_2 = Accept \bigwedge Ch_1 \neq Ch_2)$ *then return 1 else return 0.*

Then

$$acc(\xi, \eta) \leq \frac{1}{|Chset_V|} + \sqrt{res(\xi, \eta)}.$$

2.2.7 Random Oracle Model and Standard Model

The random oracle model in cryptography is the model of computation in which there is an oracle that maps every possible query to a random response from its output domain [BR93]. Random oracles have been long discussed in cryptography research. The inspiration of random oracles was from Goldreich, Goldwasser and Micali's works [GGM86, GGM84] and Fiat and Shamir's work [FS86]. The random oracle methodology was proposed and formalised by Bellare and Rogaway [BR93] and revisited by Canetti, Goldreich and Halevi [CGH98, CGH04].

In fact, a random oracle works in the same ways as a theoretical black box or an ideal hash function. It takes any certain type of input and looks up its internal database to see whether this query has been answered before outputting the response. If it is in its internal database then it outputs the corresponding value. Otherwise, it randomly obtains an answer from its source of randomness. It is assumed that each answer from the source of randomness of the random oracle is uniquely and uniformly distributed. A random oracle is a powerful tool but unrealistic[CGH04]. In practice, we provide a proof under the random oracle model and, in implementation, we replace the random oracle by conventional cryptographic hash functions. A proof in the random oracle model generally shows that a system or a protocol is secure by showing that, in order to break the protocol or the system, an adversary has to solve some mathematical problem believed to be hard or it must make an impossible query such that the oracle cannot answer. Nevertheless, a cryptographic scheme that is secure in the random oracle model does not implied that it is secure when it is instantiated with real-world implementation of hash function [CGH04]. Canetti, Goldreich and Halevi [CGH04] given some constructions of those cryptographic schemes that are not secure when implementation. Some argued that the those constructions were not practical systems and were intentionally designed such that they are not secure in the random oracle model [DP06, Sti06]. In addition, there are artificially constructed which is intended to show that the random

oracle model is not always secure in the real-world implementation. A practical scheme that is secure in the random oracle model with a heuristically-secure hash function, such as SHA-1, seems to provide a sufficient security guarantee [DP06].

However, it is interesting to design systems that are provably secure without relying on the random oracle model even through the random oracle model is still widely accepted in the cryptographic community. In order to achieve the security of a system in the standard model, some cryptographic protocols or systems require only cryptographic hash functions with some properties such as collision resistance, preimage resistance and second preimage resistance [JLO97, GHR99, CKW04, BB04, Wat05]. In the standard model, an adversary is only limited by the amount of time and computational power available and hence, a proof with a standard cryptographic hash function (which at least has a property such as collision resistance, preimage resistance and second preimage resistance) is sufficient [BB04]. Nevertheless, it is well known that security proofs in the standard model are difficult to achieve. Therefore many proofs in the cryptographic primitives are provided only in the random oracle model.

Chapter 3

Background and Cryptographic Tools

In this chapter, we provide the background knowledge required in this thesis. Some identification schemes, some signature schemes and a trapdoor scheme are presented in this chapter as basic cryptographic tools for constructing our primitives.

3.1 Signature Schemes

Proposed by Diffie and Hellman in [DH76], and formalised by Goldwasser, Micali and Rivest in [GMR88], a digital signature scheme allows a signer to preserve the integrity of a message, as well as the authenticity and non-repudiation of the signer. A digital signature scheme Σ is a triple $(SKeyGen, Sign, Verify)$, which is described as follows.

Key Generation ($KeyGen$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, outputs strings (sk_S, pk_S), which denote a private key and a public key of a signer, respectively. That is,

$$KeyGen(1^\ell) \rightarrow \{pk_S, sk_S\}.$$

Signature Signing ($Sign$):

This is a probabilistic algorithm that, given a signer's private key sk_S, a signer's public key pk_S and a message M as input, outputs a signer's signature σ. That is,

$$Sign(M, sk_S, pk_S) \rightarrow \sigma.$$

Signature Verification ($Verify$):

This is a deterministic algorithm that, given a signer's public key pk_S, a

message M and a signature σ as input, outputs a verification decision $\mathtt{d} \in \{Accept, Reject\}$. That is,

$$Verify(M, \sigma, pk_S) \rightarrow \mathtt{d}.$$

3.1.1 Security of Signature Scheme

Unforgeability

Formalised by Goldwasser, Micali and Rivest in [GMR88], the existential unforgeability property of signature schemes aims to provide assurance that, with access to a signer's public key and a signing oracle, an adversary should be unable to produce a signature on a new message chosen by himself/herself.

The following game describes the existential unforgeability of a signature scheme. Let CMA be an adaptively chosen message attack and let EUF be the existential unforgeability of a signature scheme. We denote by $\mathcal{A}$ the adaptively chosen message adversary. We also denote by $\mathcal{F}$ the simulator. First, $\mathcal{F}$ runs $KeyGen$ to obtain a private key (sk_S) and a public key (pk_S) of a signer. Then $\mathcal{F}$ constructs the signing oracle $\boldsymbol{\mathcal{SSO}}$, which is defined as follows.

$\boldsymbol{\mathcal{SSO}}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a signature σ on its choice of a message M. As a response, $\boldsymbol{\mathcal{SSO}}$ runs the $Sign$ algorithm to generate a signature σ on a message M corresponding to a signer's public key pk_S. $\boldsymbol{\mathcal{SSO}}$ then returns σ, M to $\mathcal{A}$.

Then we begin the experiment $\mathrm{Expt}^{\mathcal{A}^{CMA}_{EUF}}(\ell)$ as follows: given a choice of messages M, and access to the $\boldsymbol{\mathcal{SSO}}$ oracle, $\mathcal{A}$ arbitrarily makes queries to the oracle in an adaptive way. At the end of the above queries, we assume that $\mathcal{A}$ outputs a forged signature σ^* on a new message M^* with respect to the signer's public key pk_S. We say that $\mathcal{A}$ wins the game if:

1. $\mathcal{A}$ never made any requests for a signature on input M^*, pk_S to the $\boldsymbol{\mathcal{SSO}}$ oracle.

2. $\mathtt{Accept} \leftarrow Verify(M^*, \sigma^*, pk_S)$.

The success probability that $\mathcal{A}$ wins the above game is defined as $Succ^{CMA}_{EUF}(.)$.

Definition 3.1 *A signature scheme is said to be ($\mathtt{t}$,q_{SS},ϵ)-secure existential unforgeable under an adaptive chosen message attack if there is no PPT adversary $\mathcal{A}$*

such that the success probability $Succ_{EUF}^{CMA}(\ell) = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}_{EUF}^{CMA}$ *runs in time at most* $\mathtt{t}$*, and makes at most* q_{SS} *queries to the* $\boldsymbol{\mathcal{SSO}}$ *oracle.*

Strong Existential Unforgeability and Weak Chosen Message Attack

In 2004, Boneh and Boyen [BB04] gave other two security definitions for signature schemes named strong existential unforgeability and existential unforgeability under weak chosen message attack.

The existential unforgeability property under weak chosen message attack is similar to the existential unforgeability property described earlier. However, in the existential unforgeability under a weak chosen message attack, an adversary must submit the q_{SS} messages that he wishes to make a request of signatures to the $\boldsymbol{\mathcal{SSO}}$ oracle. Hence, the adversary ability is limited to the non-adaptive chosen message.

On the other hand, an adversary breaks the strong existential unforgeability property if the adversary can only output the message-signature pair that is different from the queried message-signature pair. The security definition of the strong existential unforgeability and the existential unforgeability under a weak chosen message attack are as follows.

Definition 3.2 *A signature scheme is said to be* ($\mathtt{t}$,q_{SS},ϵ)*-secure existential unforgeable under a weak chosen message attack if there is no PPT adversary* $\mathcal{A}$ *such that the success probability* ϵ *is non-negligible in* ℓ*, where* $\mathcal{A}$ *runs in time at most* $\mathtt{t}$*, and makes at most* q_{SS} *queries to the* $\boldsymbol{\mathcal{SSO}}$ *oracle.*

Definition 3.3 *A signature scheme is said to be* ($\mathtt{t}$,q_{SS},ϵ)*-secure strong existential unforgeable under an adaptive chosen message attack if there is no PPT adversary* $\mathcal{A}$ *such that the success probability* ϵ *is non-negligible in* ℓ*, where* $\mathcal{A}$ *runs in time at most* $\mathtt{t}$*, and makes at most* q_{SS} *queries to the* $\boldsymbol{\mathcal{SSO}}$ *oracle.*

In the following, we will review several signature schemes in the literature that are related to this thesis.

3.1.2 BLS's Short Signature Scheme from Bilinear Pairing

Introduced by Boneh, Lynn and Shacham [BLS01], a (BLS) short signature scheme Σ is a triple ($KeyGen$, $Sign$, $Verify$). The definition of this signature scheme can be found in [BLS01]. Let p be a prime and H be a hash function onto the group $\mathbb{G}_1$. We elaborate the BLS signature scheme as follows.

Key Generation ($KeyGen$):

Let $\mathsf{param} = (p, \hat{e}, g \in \mathbb{G}_1, H, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T)$ be a system parameter. Choose a random private key $x \in \mathbb{Z}_p$. Let us denote by $X = g^x$ a public key of a signer. Hence, $KeyGen$ returns $pk_S = X$ and $sk_S = x$ as a public key and a private key of a signer, respectively.

Signing ($Sign$):

Given a message M, pk_S and sk_S, S computes $\sigma = H(M)^x$ as a signature on message M.

Verification ($Verify$):

Given pk_S, σ and a message M, a verifier V checks whether $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H(M), X)$ holds or not. If not, then it outputs `reject`. Otherwise, it outputs `accept`.

3.1.3 Boneh-Boyen Short Signature without Random Oracles

In this section, we briefly describe the Boneh-Boyen signature scheme [BB04], which we will incorporate to construct our escrowed deniable identification in the standard model in Chapter 7. The Boneh-Boyen signature scheme is described as follows.

Key Generation ($KeyGen$):

$KeyGen$ randomly selects $g_a \in \mathbb{G}_1$; $g_b \in \mathbb{G}_2$; $\alpha, \eta \in \mathbb{Z}_p$ and computes $\mathcal{U} = g_2^{\alpha}$, $\mathcal{V} = g_2^{\eta}, \mathcal{Z} = \hat{e}(g_1, g_2)$. The public key is $pk_S = (g_1, g_2, \mathcal{U}, \mathcal{V}, \mathcal{Z})$ and the private key is $sk_S = (\alpha, \eta)$.

Signing ($Sign$):

Given a private key sk_S, a public key pk_S and a message $M \in \mathbb{Z}_p$, $Sign$ randomly selects $r \in z_p$ and computes $\sigma \leftarrow g_1^{1/(\alpha + r \cdot \eta + M)}$. The signature on message M is (σ, r).

Verification ($Verify$):

Given a signature, a public key pk_S and a message $M \in \mathbb{Z}_p$, $Verify$ checks whether $\hat{e}(\sigma, \mathcal{U} \cdot \mathcal{V}^r \cdot g_2^M) = \mathcal{Z}$. If it holds, then a verifier outputs `Accept`. Otherwise it outputs `Reject`.

Theorem 3.1 *If the (t', q, ϵ')-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the Boneh-Boyen signature scheme is (t, q_S, ϵ)-secure against strong existential forgery under an*

adaptive chosen message attack where T is the maximum time for an exponentiation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{Z}_p$, $q_s \leq q$, $\epsilon \approx 2\epsilon'$ and $t \leq t' - \Theta(q^2 T)$ [BB04].

Note that, for simplicity, we define $\mathbb{G}_1 = \mathbb{G}_2$ in our scheme. We also note that the weakly secure Boneh-Boyen short signature denoted as $BB04$ is different from the full Boneh-Boyen signature. The difference is $\mathcal{V}, \eta$ from the public key and private key are removed in $BB04$ and hence, a very short signature as $\sigma \leftarrow g_1^{1/(\alpha+M)}$ is obtained.

3.1.4 Waters's Short Signatures without Random Oracles

We describe the Waters's signature scheme presented in [Wat05] as follows.

System Parameters Generation ($Setup$):

$Setup$ sets $\mathsf{param} = (p, \hat{e}, g \in \mathbb{G}_1, u_0, u_1, ..., u_k \in \mathbb{G}_2, \psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2, H : \{0,1\}^* \rightarrow \{0,1\}^k, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T)$ be a system parameter.

Key Generation ($KeyGen$):

On input a system parameter param, $KeyGen$ chooses a random private key $x \in \mathbb{Z}_p$. Then, $KeyGen$ returns $pk_S = \mathbf{X} = \hat{e}(g,g)^x$ and $sk_S = x$ as a public key and a private key of a signer, respectively.

Signing ($Sign$):

Given a message M as a bit string $(M_1, ..., M_k) \in \{0,1\}^k$, pk_S and sk_S, S randomly chooses $r \in \mathbb{Z}_p$ and computes $\sigma = (\theta_1 = g^x (u_0 \prod_{i=1}^{k} u_i^{M_i})^r, \theta_2 = g^r)$ as a signature on message M.

Verification ($Verify$):

Given pk_S, σ and a message M, a verifier V checks whether

$$\hat{e}(\theta_1, g)\hat{e}(\theta_2, u_0 \prod_{i=1}^{k} u_i^{M_i})^{-1} \stackrel{?}{=} \mathbf{X}$$

holds or not. If not, then it outputs `reject`. Otherwise, it outputs `accept`.

3.2 Designated Verifier Signature Scheme

In 1996, Jakobsson, Sako and Impagliazzo [JSI96] proposed the notion of designated verifier signatures (DVS). In this notion, a signer is allowed to limit the verification

of his/her signature to a designated verifier. The designated verifier is convinced that the signature is indeed created by the original signer. It is ensured by the authenticity of this signature. However, the designated verifier is not allowed to pass or convey this conviction to another party. In other words, the verifier cannot convince the other party of the validity of this signature. Hence, the designated verifier signature ensures authenticity and privacy of signer's identity properties at the same time. We give the description of designated verifier signature schemes as follows.

A designated verifier signature scheme is a 5-tuple ($Setup$, $SKeyGen$, $VKeyGen$, $Sign$, $Verify$), which is described as follows.

System Parameter Generation ($Setup$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, outputs the system parameter param. That is,

$$Setup(1^{\ell}) \rightarrow \mathsf{param}.$$

Signer's Public Parameter and Secret Key Generator ($SKeyGen$):

This is a probabilistic algorithm that, given a system parameter param as input, outputs the private key (sk_S) and the public parameter (pk_S) of the signer. That is,

$$SKeyGen(\mathsf{param}) \rightarrow (pk_S, sk_S).$$

Verifier's Public Parameter and Secret Key Generator ($VKeyGen$):

This is a probabilistic algorithm that, given a system parameter param as input, outputs strings (sk_V, pk_V) where they denote the private key and the public parameter of signer, respectively. That is,

$$VKeyGen(\mathsf{param}) \rightarrow (pk_V, sk_V).$$

Designated Verifier Signature Signing ($DSign$):

This is a probabilistic algorithm that, given a signer's private key sk_S, a signer's public parameter pk_S, a verifier's public parameter pk_V, a message M as input, $DSign$ outputs a signer's designated verifier signature ρ. That is,

$$DSign(M, sk_S, pk_S, pk_V) \rightarrow \rho.$$

Designated Verifier Signature Verification ($DVerify$):

This is a deterministic algorithm that, given a signer's public parameter pk_S, a verifier's private key sk_V, a message M and a signature ρ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$DVerify(M, \rho, sk_V, pk_S) \rightarrow \mathsf{d}.$$

Simulation of a Delegated Signature ($DSimulate$): This is a probabilistic algorithm that, given verifier's public parameter pk_V, verifier's private key sk_V, signer's public parameter pk_S, and a message M as input, outputs a designated verifier signature ϱ such that

$$DVerify(M, \varrho, sk_V, pk_S) \rightarrow Accept.$$

That is,

$$DSimulate(M, pk_V, sk_V, pk_S) \rightarrow \varrho.$$

3.2.1 Security of Designated Verifier Signature Scheme

Unforgeability

There are actually two different types of unforgeability properties that we should consider and they are mentioned in [SBWP03, HSMW06]. They are "standard signature unforgeability" and "designated verifier unforgeability". Huang et al. [HSMW06] concluded that "designated verifier unforgeability" always implies "standard signature unforgeability". We refer the reader to [SBWP03] and [HSMW06] for the formal proof of this equivalence. Hence, from now on, when we discuss unforgeability property of DVS (and UDVS) schemes, we refer to "designated verifier unforgeability" property. A security model against existential unforgeability under adaptively chosen message attack of designated verifier scheme is formally given as follows: let denote by $EUF\text{-}DVS$, the existential unforgeability of DVS scheme. Let $\mathcal{F}$ be a simulator. We then define $\mathcal{A}$ as the adaptively chosen message adversary. Next, $\mathcal{F}$ constructs the designated verifier signing oracle $\boldsymbol{\mathcal{DDO}}$ and the random oracle $\boldsymbol{\mathcal{HO}}$ as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: $\mathcal{A}$ can make queries at most q_H times for a hash value on a string M. $\mathcal{F}$ responds each query by first, search its database for the duplicate. If it outputs yes, then $\mathcal{F}$ returns the match hash value. Otherwise, $\mathcal{F}$ randomly

selects an integer i from its random domain and outputs i as a hash value for the string M. Then, $\mathcal{F}$ updates i to its database.

$\mathcal{DDO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{DD} times for a designated verifier signature ρ on its choice of message M under the signer's public parameter pk_S. Then, in return, $\mathcal{F}$ runs the $DSign$ algorithm to generate a designated verifier signature ρ on a message M corresponding with pk_S. After that, $\mathcal{F}$ returns ρ, M to $\mathcal{A}$.

The game between $\mathcal{F}$ and $\mathcal{A}$ that describes the existential unforgeability of DVS scheme can be defined as follows. $\mathcal{A}$ is given a choice of message M and the signer's public parameter pk_S, the verifier's public key pk_V and an access to the designated verifier signing oracle and the hash oracle as input. Then, we assume that $\mathcal{A}$ returns a forged signature ρ_* on its choice of message M^*. We say that $\mathcal{A}$ wins the game if

1. $Accept \leftarrow DVerify(M^*, \rho_*, pk_V, sk_V, pk_S)$.
2. With M^* as input, $\mathcal{A}$ never made a request for a designated verifier signature to the $\mathcal{DDO}$ oracle.

Let $Succ^{CM-A}_{EUF-DVS}(.)$ be defined as the success probability of that $\mathcal{A}$ wins the above game.

Definition 3.4 *A designated verifier signature scheme is ($\mathtt{t}$,q_H,q_{DD},ϵ)-secure against existential unforgeability under adaptively chosen message attack if there is no PPT CM-A adversary $\mathcal{A}$ such that the success probability $Succ^{CM-A}_{EUF-DVS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}$ runs in time at most $\mathtt{t}$, make at most q_H and q_{DD} queries to the random oracle $\mathcal{HO}$ and the $\mathcal{DDO}$ oracle, respectively.*

Privacy of Signer's Identity

Laguillaumie and Vergnaud [LV04] formalised the privacy of signer's identity property of a designated verifier signature scheme. This property prevents the other party that does not know a verifier's private key to verify a designated verifier signature.

Let $\mathcal{A}$ be the adaptively chosen message distinguisher. Let $\mathcal{F}$ be a simulator. Let *PSI-DVS* be the existential privacy of signer's identity of designated verifier signature scheme. The game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential privacy of signer's identity of designated verifier signature scheme defined as follows.

The game is separated into two phases. $\mathcal{F}$ starts with generation of public keys as follows.

$$\begin{aligned} SKeyGen(\mathsf{param}) &\rightarrow (pk_{S_0}, sk_{S_0}), \\ SKeyGen(\mathsf{param}) &\rightarrow (pk_{S_1}, sk_{S_1}), \\ VKeyGen(\mathsf{param}) &\rightarrow (pk_V, sk_V). \end{aligned}$$

Next, $\mathcal{F}$ constructs the designated verifier signing oracle $\boldsymbol{\mathcal{DDO}}$ and the designated verifier verification oracle $\boldsymbol{\mathcal{DVO}}$, which are defined as follows.

$\boldsymbol{\mathcal{DDO}}$ oracle: First, $\mathcal{A}$ can make queries at most q_{DD} times for a designated verifier signature ρ on its choice of message M under the signer's public parameter pk_{S_0} (or pk_{S_1}). Then, in return, $\mathcal{F}$ runs the $DSign$ algorithm to generate a designated verifier signature ρ on a message M corresponding with pk_{S_0} (or pk_{S_1}). After that, $\mathcal{F}$ returns ρ, M to $\mathcal{A}$.

$\boldsymbol{\mathcal{DVO}}$ oracle: $\mathcal{A}$ can make queries at most q_{DV} times for the verification of a designated verifier signature ρ on its chosen message M with its corresponding public parameter pk_V and pk_{S_0} (or pk_{S_1}). Next, $\mathcal{F}$ runs the $DVerify$ algorithm to verify a designated verifier signature ρ on a message M corresponding with pk_V and pk_{S_0} (or pk_{S_1}). Finally, $\mathcal{F}$ returns *Accept* if a designated verifier signature ρ on a message M is valid regarding to pk_V and pk_{S_0} (or pk_{S_1}), otherwise, it returns *Reject*.

The game is then run as follows.

1. **Phase** 1**:** $\mathcal{A}$ is allowed to make a request to $\boldsymbol{\mathcal{DVO}}$ and $\boldsymbol{\mathcal{DDO}}$ oracles. The oracles answer as their design.

2. **Challenge:** When $\mathcal{A}$ is ready to challenge $\mathcal{F}$, it outputs M^* with the constraints that with M^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{DDO}}$ oracle and a request for a validation of designated verifier signature to the $\boldsymbol{\mathcal{DVO}}$ oracle. In response, $\mathcal{F}$ selects a bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 0$ then $\mathcal{F}$ outputs $\rho \leftarrow DSign(M^*, sk_{S_0}, pk_{S_0}, pk_V)$ to $\mathcal{A}$. Otherwise, $\mathcal{F}$ outputs $\rho \leftarrow DSign(M^*, sk_{S_1}, pk_{S_1}, pk_V)$ to $\mathcal{A}$.

3. **Phase** 2**:** $\mathcal{A}$ is allowed to return arbitrarily to *Phase* 1 or *Challenge* as many times as it wants. However, there are conditions that

a. with M^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\mathcal{DDO}$ oracle.

b. with M^* and ρ as input, $\mathcal{A}$ never submitted a request for a validation of the designated verifier signature ρ to the $\mathcal{DVO}$ oracle.

4. **Guessing:** Finally, $\mathcal{A}$ responds with a guess b', which is corresponding to the challenge M^*. The distinguisher wins the game if $b = b'$.

Let $Succ^{CM-A}_{PSI-DVS}(.)$ be the success probability of that $\mathcal{A}$ wins the above game.

Definition 3.5 *A designated verifier signature scheme is (*$\mathtt{t}$*,*q_{DD}*,*q_{DV}*,*ϵ*)-secure against existential privacy of signer's identity under adaptively chosen message attacks if there is no PPT CM-A distinguisher* $\mathcal{A}$ *such that the success probability* $Succ^{CM-A}_{PSI-DVS}(\ell)$ $= |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}$ *runs in time at most* $\mathtt{t}$*, make at* q_{DD} *and* q_{DV} *queries to the* $\mathcal{DDO}$ *oracle and* $\mathcal{DVO}$ *oracle, respectively.*

3.3 Universal Designated Verifier Signature Scheme

In Asiacrypt 2003, Steinfeld, Bull, Wang and Pieprzyk [SBWP03] proposed the notion of universal designated verifier signature (UDVS) schemes. This is a novel extension of the designated verifier signature scheme. A universal designated verifier signature is an ordinary designated verifier signature with an additional function that represents the authenticity of the possession of a signature on a message. A signature holder is introduced as a party who is given the privilege of designating the signature to any verifier that is chosen by him/her. In other words, a signature holder obtains a signature from the original signer and then designates this signature to a designated verifier. A universal designated verifier signature is provided, and hence the privacy of the original signer is protected and the anonymity of the signature holder is ensured. We give the description of universal designated verifier signature schemes as follows: first of all, it must be assumed that all parties must comply with a registration protocol with a certificate authority (CA) to prove the knowledge of their private key and to obtain a certificate on their public parameters. A universal designated verifier signature scheme is a 8-tuple ($Setup$, $SKeyGen$, $Sign$, $Verify$, $VKeyGen$, $Delegate$, $DVerify$, $DSimulate$), which is described as follows.

System Parameter Generation (*Setup*):

This is a probabilistic algorithm that, given a security parameter ℓ as input,

outputs the system parameter param. That is,

$$Setup(1^{\ell}) \rightarrow \mathsf{param}.$$

Signer's Public Parameter and Secret Key Generator ($SKeyGen$):

This is a probabilistic algorithm that, given a system parameter param as input, outputs the private key (sk_S) and the public parameter (pk_S) of the signer. That is,

$$SKeyGen(\mathsf{param}) \rightarrow (pk_S, sk_S).$$

Signature Signing ($Sign$):

This is a probabilistic algorithm that, given a private key sk_S, a public parameter pk_S, a message M as input, $Sign$ outputs signer's signature σ. That is,

$$Sign(M, sk_S, pk_S) \rightarrow \sigma.$$

Signature Verification ($Verify$):

This is a deterministic algorithm that, given a signer's public parameter pk_S, a message M and a signature σ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(M, \sigma, pk_S) \rightarrow \mathsf{d}.$$

Verifier's Public Parameter and Secret Key Generator ($VKeyGen$):

This is a probabilistic algorithm that, given a system parameter param as input, outputs strings (sk_V, pk_V) where they denote the private key and the public parameter of the verifier, respectively. That is,

$$VKeyGen(\mathsf{param}) \rightarrow (pk_V, sk_V).$$

Signature Delegation ($Delegate$):

This is a probabilistic algorithm that, given a verifier's public parameter pk_V, a signer's public parameter pk_S, a signer's signature σ, and a message M as input, outputs a designated verifier signature ρ. That is,

$$Delegate(M, \sigma, pk_V, pk_S) \rightarrow \rho.$$

Designated Verifier Signature Verification ($DVerify$):

This is a deterministic algorithm that, given a verifier's private key sk_V, a signer's public parameter pk_S, a message M and a designated verifier signature ρ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$DVerify(M, \rho, sk_V, pk_S) \rightarrow \mathsf{d}.$$

Simulation of a Delegated Signature ($DSimulate$): This is a probabilistic algorithm that, given a verifier's public parameter pk_V, a verifier's private key sk_V, a signer's public parameter pk_S, and a message M as input, outputs a designated verifier signature ϱ such that

$$DVerify(M, \varrho, sk_V, pk_S) \rightarrow Accept.$$

That is,

$$DSimulate(M, pk_V, sk_V, pk_S) \rightarrow \varrho.$$

3.3.1 Security of Universal Designated Verifier Signature Scheme

Unforgeability

Unforgeability under adaptively chosen message attack of universal designated verifier scheme is formally given as follows.

Denote $EUF\text{-}UDVS$ the existential unforgeability of UDVS scheme and denote by $CM\text{-}A$ the adaptively chosen message attack. Let $\mathcal{F}$ be a simulator. We then define $\mathcal{A}$ as the adaptively chosen message and chosen public key adversary. Next, $\mathcal{F}$ constructs the signing oracle $\boldsymbol{\mathcal{SSO}}$, the signature delegation oracle $\boldsymbol{\mathcal{DSO}}$ and the random oracle $\boldsymbol{\mathcal{HO}}$ as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: $\mathcal{A}$ can make queries at most q_H times for a hash value on a string M. $\mathcal{F}$ responds each query by first, search its database for the duplicate. If it outputs yes, then $\mathcal{F}$ returns the match hash value. Otherwise, $\mathcal{F}$ randomly selects an integer i from its random domain and outputs i as a hash value for the string M. Then, $\mathcal{F}$ updates i to its database.

$\boldsymbol{\mathcal{SSO}}$ oracle: First, $\mathcal{A}$ can make queries at most q_{SS} times for a signature σ on its choice of message M under the signer's public parameter pk_S. Then, in

return, $\mathcal{F}$ runs the *Sign* algorithm to generate a signature σ on a message M corresponding with pk_S. After that, $\mathcal{F}$ returns σ, M to $\mathcal{A}$.

$\mathcal{DSO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{DS} times for a designated verifier signature ρ on its choice of message M under the signer's public parameter pk_S and the verifier's public parameter pk_V. Next, $\mathcal{F}$ runs the *Delegate* algorithm to generate a designated verifier signature ρ on a message M corresponding with pk_S, pk_V. Finally, $\mathcal{F}$ outputs ρ, M and returns them to $\mathcal{A}$.

The game between $\mathcal{F}$ and $\mathcal{A}$ that describes the existential unforgeability of UDVS scheme can be defined as follows. $\mathcal{A}$ is given a choice of message M and the signer's public parameter pk_S, the verifier's public key pk_V and an access to the designated verifier signing oracle and the hash oracle as input. Then, we assume that $\mathcal{A}$ returns a forged signature ρ_* on its choice of message M^* corresponding with the signer's public parameter pk_{S^*} and the verifier's public parameter pk_{V^*}. We say that $\mathcal{A}$ wins the game if

1. $Accept \leftarrow DVerify(M^*, \rho_*, pk_{V^*}, sk_{V^*}, pk_{S^*})$.
2. With M^*, pk_{V^*} and pk_{S^*} as input, $\mathcal{A}$ never made a request for a signature to the $\mathcal{SSO}$ oracle.
3. With M^* and pk_V^* as input, $\mathcal{A}$ never made a request for a designated verifier signature to the $\mathcal{DSO}$ oracle.

Let $Succ^{CM-A}_{EUF-UDVS}(.)$ be defined as the success probability of that $\mathcal{A}$ wins the above game.

Definition 3.6 *A universal designated verifier signature scheme is ($\mathtt{t}$,q_H,q_{SS},q_{DS},ϵ)-secure against existential unforgeability under adaptively chosen message attack if there is no PPT CM-A adversary $\mathcal{A}$ such that the success probability $Succ^{CM-A}_{EUF-UDVS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}$ runs in time at most $\mathtt{t}$, make at most q_H, q_{SS} and q_{DD} queries to the random oracle $\mathcal{HO}$, the $\mathcal{SSO}$ oracle and the $\mathcal{DSO}$ oracle, respectively.*

Non-transferability Privacy

Introduced in [SBWP03] by Steinfeld et al., the non-transferability privacy property of a universal designated verifier signature scheme prevents a designated verifier to

generate evidence, which convinces a third-party the validity of a designated verifier signature ρ.

Let $\mathcal{A}$ be the adaptively chosen message distinguisher. Let $\mathcal{F}$ be a simulator. Let $ENT\text{-}UDVS$ be the existential non-transferable privacy of universal designated verifier signature scheme. The game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential non-transferable privacy of universal designated verifier signature scheme defined as follows.

The game is separated into two phases. Start with $\mathcal{F}$ to construct the hash oracle $\mathcal{HO}$, the verifier's key generator oracle $\mathcal{VKO}$, the signature signing oracle $\mathcal{SSO}$ and the signature delegation oracle $\mathcal{DSO}$, which are defined as follows.

$\mathcal{HO}$ oracle: $\mathcal{A}$ can make queries at most q_H times for a hash value on a string M. $\mathcal{F}$ responds each query by first, search its database for the duplicate. If it outputs yes, then $\mathcal{F}$ returns the match hash value. Otherwise, $\mathcal{F}$ randomly selects an integer i from its random domain and outputs i as a hash value for the string M. Then, $\mathcal{F}$ updates i to its database.

$\mathcal{VKO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{VP} times for the public parameter pk_V and the private key of the verifier. Then, in return, $\mathcal{F}$ runs the $VKeyGen$ algorithm to generate a private key sk_V and a public parameter pk_V of a verifier. $\mathcal{F}$ outputs pk_V, sk_V and returns to $\mathcal{A}$.

$\mathcal{SSO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{SS} times for a signature σ on its choice of message M under the signer's public parameter pk_S. Then, in return, $\mathcal{F}$ runs the $Sign$ algorithm to generate a signature σ on a message M corresponding with pk_S. After that, $\mathcal{F}$ returns σ, M to $\mathcal{A}$.

$\mathcal{DSO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{DS} times for a designated verifier signature ρ on its choice of message M under the signer's public parameter pk_S and the verifier's public parameter pk_V. Next, $\mathcal{F}$ runs the $Delegate$ algorithm to generate a designated verifier signature ρ on a message M corresponding with pk_S, pk_V. Finally, $\mathcal{F}$ outputs ρ, M and returns them to $\mathcal{A}$.

The game is then run as follows.

1. **Phase** 1**:** $\mathcal{A}$ is allowed to make a request to $\mathcal{VKO}$, $\mathcal{SSO}$ and $\mathcal{DSO}$ oracles. The oracles answer as their design.

2. **Challenge:** When $\mathcal{A}$ is ready to challenge $\mathcal{F}$, it outputs M^*, pk_{S^*} and pk_{V^*} with the constraints that

   a. with M^* and pk_{S^*} as input, $\mathcal{A}$ never made a request for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. with M^*, pk_{S^*} and pk_{V^*} as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{DSO}}$ oracle.

   In response, $\mathcal{F}$ selects a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$ then $\mathcal{F}$ computes $\sigma \leftarrow Sign(M^*, sk_S, pk_S)$ to $\mathcal{A}$ and outputs $\rho \leftarrow Delegate(M^*, \sigma, pk_V, pk_S)$ to $\mathcal{A}$. Otherwise, $\mathcal{F}$ outputs $\varrho \leftarrow DSimulate(M^*, pk_V, sk_V, pk_S)$ to $\mathcal{A}$.

3. **Phase** 2**:** $\mathcal{A}$ is allowed to return arbitrarily to *Phase* 1 or *Challenge* as many times as it wants. However, there is a condition that $\mathcal{A}$ must have at least one set of challenge M^* such that

   a. with M^* and pk_{S^*} as input, $\mathcal{A}$ never made a request for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. with M^*, pk_{S^*} and pk_{V^*} as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{DSO}}$ oracle.

4. **Guessing:** Finally, $\mathcal{A}$ responds with a guess b', which is corresponding to the challenge M^*, pk_{S^*} and pk_{V^*}. The distinguisher wins the game if $b = b'$.

Let $Succ^{CM-A}_{ENT-UDVS}(.)$ be the success probability of that $\mathcal{A}$ wins the above game.

Definition 3.7 *A universal designated verifier signature scheme is* ($\mathtt{t}$,q_H,q_{SS},q_{DS},q_{VP},ϵ)*-secure against existential non-transferable privacy under adaptively chosen message attack chosen message attacks if there is no PPT CM-A distinguisher* $\mathcal{A}$ *such that the success probability* $Succ^{CM-A}_{ENT-UDVS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}$ *runs in time at most* $\mathtt{t}$*, make at most* q_H*,* q_{SS}*,* q_{DS} *and* q_{VP} *queries to the random oracle* $\boldsymbol{\mathcal{HO}}$*, the* $\boldsymbol{\mathcal{SSO}}$ *oracle, the* $\boldsymbol{\mathcal{DSO}}$ *oracle and the* $\boldsymbol{\mathcal{VKO}}$ *oracle, respectively.*

3.4 Trapdoor Commitment Scheme

Introduced by Brassard, Chaum and Crépeau in [BCC88], a trapdoor commitment scheme allows a prover to generate some commitment bits that involve himself/herself such that a verifier learns nothing from these bits without his/her help.

A trapdoor commitment scheme TC is a triple ($Setup$, $Tcom$, $Topen$), which is described as follows.

Key Generation ($KeyGen$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, generates a public parameter pk_D and a trapdoor key sk_D. That is,

$$KeyGen(1^\ell) \rightarrow \{pk_D, sk_D\}.$$

Trapdoor Commitment Generation ($Tcom$):

This is a probabilistic algorithm that, given a public key pk_D, a message M and a random integer r as input, outputs a commitment value T. That is,

$$Tcom(pk_D, M, r) \rightarrow T.$$

Trapdoor Commitment Opening ($Topen$):

This is a probabilistic algorithm that, given a trapdoor key sk_D, a public parameter pk_D, a message M, a new message M', an integer r, outputs an integer r' such that $T = Tcom(pk_D, M, r) = Tcom(pk_D, M', r')$. That is,

$$Topen(sk_D, pk_D, T, M, M', r) \rightarrow r'.$$

3.4.1 A Concrete Scheme of a Trapdoor Commitment Scheme

The idea of transforming an identification scheme into a trapdoor commitment scheme was presented by Kurosawa and Heng in PKC 2006 [KH06]. We elaborate the Schnorr trapdoor commitment scheme transformed from the Schnorr identification scheme [Sch91] as follows.

Key Generation ($KeyGen$):

On input a security parameter ℓ, $KeyGen$ randomly selects a prime α such that $\alpha = poly(1^\ell)$. Next, let $\mathbb{G}_\alpha$ be a multiplicative group order α and then choose a random generator $g_\alpha \in \mathbb{G}_\alpha$ and a random number $y \in \mathbb{Z}_\alpha^*$. Let us denote by $\mathsf{param} = (\alpha, g_\alpha)$ the system parameter and by $Y = g_\alpha^y$ a public key. Finally, $KeyGen$ outputs public parameter $pk_D = (\mathsf{param}, Y)$ and a secret trapdoor key $sk_D = y$.

Trapdoor Commitment Generation ($Tcom$):

On input public parameter pk_D and two integers $M, r \in \mathbb{Z}_\alpha^*$, $Tcom$ computes an output $T = g_\alpha^r Y^M$. Then, $Tcom$ responds with T.

Trapdoor Commitment Opening ($Topen$):

On input public parameter pk_D, a private key sk_D and three integers $M', M, r \in \mathbb{Z}_\alpha^*$, $Topen$ computes r' such that $T = g_\alpha^r Y^M = g_\alpha^{r'} Y^{M'}$. Then, $Topen$ returns r'.

We supply the security of the above trapdoor commitment scheme as follows.

Definition 3.8 *We say that a trapdoor commitment scheme TC is secure if, on input pk_D, it is computationally infeasible to compute (M, r) and (M', r') such that $Tcom(pk_D, M, r) = Tcom(pk_D, M', r')$ where $M \neq M'$. [KH06]*

Theorem 3.2 ([KH06]) *The above trapdoor commitment scheme is secure if the discrete logarithm assumption holds.*

3.5 Identification Scheme

The fundamental study of identification schemes was undertaken by Fiat and Shamir [FS86]. Following this work, other important identification schemes such as [Sch89, OO88, Oka92, GQ88, FFS88] have been proposed. In 1988, Feige, Fiat, and Shamir [FFS88] introduced an identification scheme, which is based on the difficulty of inverting RSA. Subsequently, Guillou and Quisquarter [GQ88] provided an identification scheme based on RSA (we refer this scheme as the GQ scheme from now on). Later, Schnorr [Sch89] introduced his identification scheme based on a discrete logarithm problem. Both GQ and Schnorr's schemes are still the most efficient and well-studied identification schemes in the literature, and their security has been analysed and proven by Bellare and Palacio [BP02]. The security of their schemes has been reduced to a standard computational problem such as, factoring or discrete logarithms for a passive attack setting and one-more RSA and one-more discrete logarithm for a concurrent attack. There are several other identification schemes based on a similar problem to the above schemes, such as the Ohta and Okamoto identification scheme [OO88], which is a variant of the Feige, Fiat, and Shamir identification scheme, and the Ong and Schnorr identification scheme [OS90], which is a variant of the Fiat and Shamir identification scheme based on a factorisation of 2^m-th root.

Since the introduction of bilinear maps in cryptography, many new problems such as the Gap Diffie-Hellman problem, the Bilinear Diffie-Hellman problem and

others have been studied. Based on these new problems, many new identification schemes have been proposed in the literature. Kim and Kim [KK02a] introduced the first identification scheme based on the Bilinear Diffie-Hellman problem. Their scheme was later broken and improved by Yao, Wang and Wang [YWW04].

The security of identification schemes under active attack was first formalised by Shoup [Sho99]. Shoup also provided an analysis of several identification schemes that are based on the square root or 2^m-th root problem, such as the Guillou and Quisquarter (GQ) identification scheme [GQ88]. In Crypto'02 [BP02], Bellare and Palacio formalised the security for both identification schemes and signature schemes (together with their identity-based variants). They also added a formal definition for all impersonation attacks of identification schemes, including passive, active and concurrent attacks. Later, a summary of the security proof for identity-based identification schemes and signature schemes was proposed by Bellare, Namprempre and Neven in [BNN04]. They also formalised the definition of attacks, standard identification schemes, identity-based identification schemes and the security properties of those schemes.

The strongest security model for identification schemes is impersonation against reset attack. In reset attacks, an adversary is given the power to reset the initial state of the honest prover in the concurrent setting. The first identification scheme secure under reset attack was proposed by Canetti, Goldwasser, Goldreich and Micali [CGGM00]. However, their scheme in the public key model is inefficient in practice.

Later, in Eurocrypt'01 [BFGM01], Bellare, Fischlin, Goldwasser and Micali defined two types of reset attack, namely the concurrent-reset-1 (CR1) attack and the concurrent-reset-2 (CR2) attack. An adversary $\mathcal{A}$ in the CR1 setting is allowed to execute many identification protocols concurrently with a prover P. While interacting with the prover P (or its clones), $\mathcal{A}$ can reset P (or its clones) to the initial state. Finally, after $\mathcal{A}$ obtains sufficient information, it then attempts to impersonate P.

Similar to the CR1 setting, the adversary $\mathcal{A}$ in CR2 setting still can go back to execute many identification protocols concurrently with the honest prover P during the impersonation stage. Moreover, during the executions after impersonation, $\mathcal{A}$ still can reset P to the initial state. The above setting can be viewed as a man in the middle plus reset attack setting. Hence, a CR1 attack is a special case of CR2 attack from the above point of view. Therefore, an identification protocol secured under a CR2 attack implies that it is also secure under a CR1 attack.

3.5.1 Types of Attack

In general, an identification scheme is said to be compromised if an adversary succeeds in an impersonation attempt (making the verifier accept with non-negligible probability) [FS86]. We can classify the types of attack according to the interaction allowed to the adversary before an impersonation attempt [Sho99]. There are several types of attack that an adversary, trying to impersonate a prover, may attempt. However, we will describe only three basic types of attack for background to this thesis. These three types of attack are introduced and formalised in [FS86, Sho99, BFGM01]. For $\mathsf{CR1}^+$ attack that we have introduced in [TSM09a], we will give full details in Chapter 7.

Passive Attack (PA)

The weakest form of attack is the passive attack in which adversary is an eavesdropper that attempts an impersonation using only his knowledge of the public key of the prover and eavesdropped transcripts. Furthermore, the adversary is not allowed to interact with the system at all before attempting an impersonation. Other attacks at an intermediate level, such as an eavesdropping attack or honest-verifier attack, are essentially equivalent to a passive attack.

Active Attack (AA)

A stronger form of attack is the active attack, in which an adversary is allowed to interact with P several times, posing as V. In this sense, the adversary is a cheating prover and a cheating verifier. We may consider active attacks as adaptive chosen cipher text attacks. We should note that active attacks are quite feasible in many practical situations. For example, the man-in-the-middle-attack is equivalent to the active attack where the adversary pretends to be a verifier and interacts with a prover to obtain some information prior the impersonation. Security against active attacks is hence clearly preferable to security against only passive attacks.

Concurrent Attack (CA) and Concurrent Reset Attack (CR)

Another stronger form of passive attack is the concurrent-active attack. In this scenario, an adversary is allowed to interact with an honest prover several times prior to impersonation, acting as a cheating verifier. Furthermore, he could interact with

many different provers (clones) concurrently. All clones have the same private key. However, they maintain their own independent states. Security against impersonation under a concurrent attack implies security against impersonation under an active attack [BNN04]. In [BFGM01], Bellare *et al.* provided a formal definition of the concurrent reset attack and further divided it into two different classes, namely CR1 and CR2, depending on whether the adversary is still allowed to execute identification protocols with the honest prover during the impersonation stage or not. CR1 can be viewed as a special case of CR2.

CR1$^+$ Attack

In [TSM09a], we defined a strong type of concurrent reset attack, namely CR1$^+$. This attack is stronger than CR1 in the sense that the adversary is allowed to reset the prover (or clones) to *any* state, instead of just the initial state as defined in CR1. This means that the adversary can still play the role of a cheating verifier prior to impersonation as in the concurrent attack, CR1. However, these states of the prover (clones) can be reset to the initial state or to any other state. Security against impersonation under the CR1$^+$ thus implies security against impersonation under an active and concurrent attacks, as well as a CR1 attack.

The Properties of Identification Scheme

The primary objectives of an identification protocol are

- **Completeness** in the case of honest parties, the prover is successfully able to authenticate itself to the verifier;
- **Soundness** a dishonest prover has a negligible probability of convincing a verifier.

3.5.2 Definition of Identification Scheme

In an identity-based identification scheme (ID-scheme), the algorithms can be classified into one PPT algorithm and one protocol as follows.

1. **Key Generation ($KeyGen$):**
   Given a security parameter ℓ, $KeyGen$ takes 1^ℓ as input and generates a pair of public instance (pk_P) and witness instance (sk_P). The equation is $KeyGen(1^\ell) \rightarrow (pk_P, sk_P)$.

2. **Identification Protocol ($\langle P, V \rangle$):**
   A canonical protocol of an ID-scheme can be formalised by $CID = (Commit, Response, Check)$. $Commit$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier.

   - Step 1. P chooses r at random from a certain domain and computes $x \leftarrow Commit(r)$. P then sends x to V.
   - Step 2. V chooses a challenge c at random from a certain set and sends it to P.
   - Step 3. P computes a response $y \leftarrow Response(sk_P, r, c)$ and sends y to V.
   - Step 4. V computes $\mathsf{d} \leftarrow Check(pk_P, x, c, y)$, where d is a decision $\mathsf{d} \in \{Accept, Reject\}$. V accepts P if and only if $\mathsf{d} = Accept$.

3.5.3 Security of Identification Scheme

The following security notations of a secure identification scheme are formally defined using the same notations as in [SMP88, KK02b, Sho99]. We use the similar conventions as in [FFS88]:

1. $\overline{P}$ represents an honest prover that follows its designated protocol, $\widetilde{P}$ is a polynomial-time cheater, and P acts as $\overline{P}$ or $\widetilde{P}$.
2. $\overline{V}$ represents a valid verifier that follows the designated protocol, $\widetilde{V}$ is an arbitrary polynomial-time algorithm, which may try to extract additional information from P, and V acts as $\overline{V}$ or $\widetilde{V}$.
3. $\langle P, V \rangle$ represents the execution of the two party protocol, where P is the prover and V is the verifier.

An adversary $(\widetilde{P}, \widetilde{V})$ is a pair of probabilistic polynomial-time interactive algorithms. Given the key pair pk_P, an adversary $\widetilde{V}$ allows to interact with a prover P several times and output a string h. The string h (called a "help string") is used as input to the adversary $\widetilde{P}$ who attempts to convince $\overline{V}$. We denote by $\langle \widetilde{P}(h), \overline{V}(pk_P) \rangle$ the execution of protocol between $\overline{V}$ and $\widetilde{P}$ where $\widetilde{P}$ takes h as input and $\overline{V}$ takes pk_P as input. We adopt the definition of security against active attacks with respect to such adversaries as described by [Sho99] as follows.

Definition 3.9 *An identification scheme* $(KeyGen, P, V)$ *is secure against active attacks if for all adversaries* $(\widetilde{P}, \widetilde{V})$*, for all constants* $c > 0$*, and for all sufficiently large* ℓ*,*

$$\Pr\left[string = 1 \;\middle|\; \begin{array}{l} (pk_P; sk_P) \leftarrow KeyGen(1^\ell); \\ h \leftarrow \langle \overline{P}(pk_P, sk_P), \widetilde{V}(pk_P)\rangle; \\ string \leftarrow \langle \widetilde{P}(h), \overline{V}(pk_P)\rangle. \end{array}\right] < \ell^{-c}.$$

3.5.4 Schnorr's Identification Scheme

Schnorr proposed the first identification scheme based on a discrete logarithm problem [Sch89]. Later, Bellare and Palacio provided the security proof for Schnorr's identification scheme [BP02].

The security of Schnorr's identification scheme relies on discrete logarithm assumption for passive attacks and one-more discrete logarithm assumption for concurrent attacks. This scheme is still the most efficient and well-studied identification scheme in the literature and hence, many cryptography primitives construction are based on Schnorr's identification scheme. We briefly describe Schnorr's identification scheme as follows.

Protocol Description

1. **Key Generation (**$KeyGen$**):**
   Given a security parameter ℓ, which is a positive integer, key generation works as follows.

   (a) Take the security parameter 1^ℓ as input and select primes p and q, such that $q|p-1$ and $|q| = l$. (Note: e.g. $q \geq 2^{160}$, and $p \geq 2^{1024}$.)

   (b) Select an element $g \in \mathbb{Z}_p^*$ with order q and a security parameter t, such that $t = O(|p|)$ (e.g., $t \geq 20$).

   (c) Select a random integer $s \in \mathbb{Z}_q^*$ and compute $v = g^{-s} \bmod p$.

   Here, a pair of private key and public parameter is generated where a public parameter pk_P is (p, q, g, t, v) and a private key sk_P is (s).

2. **Identification protocol (**$\langle P, V\rangle$**):**
   A canonical protocol of an identification scheme can be formalised by $CID = (Commit, Response, Check)$. $Commit$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier.

- Step 1. P chooses a random integer $r \in \mathbb{Z}_q^*$ and, with the public parameter $pk_P = (p, q, g, t, v)$, computes $Commit(p, g, r) = Y = g^r \ (mod\ p)$. P then sends Y to V.
- Step 2. V chooses a random challenge integer $c \in \mathbb{Z}_{2^t}$ and sends it to P.
- Step 3. P computes a response $Response(sk_P, r, c) = z$ where $z = r + c \cdot s \ (mod\ q)$ and sends z to V.
- Step 4. V checks if $Y \equiv g^z v^c \ (mod\ p)$ then $Check(pk_P, Y, c, z) \rightarrow Accept$, otherwise $Check(pk_P, Y, c, z) \rightarrow Reject$. V accepts P if and only if, $\mathsf{d} = Accept$.

3.6 Identity-based Identification Scheme

The idea of an identity-based cryptosystem was first introduced in 1984 by Shamir [Sha84]. The first identity-based identification scheme was introduced by Feige, Fiat, and Shamir [FFS88] in 1988. Since then, many identity-based identification schemes have been proposed [KH05, KH04, KH06, Fre05].

3.6.1 Definition of Identity-based Identification Scheme

In an identity-based identification scheme (ID-scheme), the algorithms can be classified into two PPT algorithms and one protocol as follows.

1. **Key Generation ($KeyGen$):**
   Given a security parameter ℓ, $KeyGen$ takes 1^ℓ as input and generates a public parameter (pk_K) and a master private key (sk_K). The equation is $KeyGen(1^\ell) \rightarrow (pk_K, sk_K)$.

2. **Key Extraction ($Extract$):**
   Given the identity of the prover (ID) and sk_K, $Extract$ takes (ID) and sk_K as input and computes a witness instance (a prover's private key) sk_P then gives it to the prover. The equation is $Extract(ID, sk_K) \rightarrow sk_P$.

3. **Identification Protocol ($\langle P, V \rangle$):**
   A canonical protocol of ID-scheme can be formalised by $CID = (Commit, Response, Check)$, where $Commit$, $Response$ and $Check$ are PPT algorithms used in the following protocol where P is the prover and V is the verifier.

- Step 1. P chooses r at random from a certain domain and computes $x \leftarrow Commit(r)$. P then sends x to V.
- Step 2. V chooses a challenge c at random from a certain set and sends it to P.
- Step 3. P computes a response $y \leftarrow Response(sk_P, x, c)$ and sends y to V.
- Step 4. V computes $\mathsf{d} \leftarrow Check(pk_K, ID, x, c, y)$, where d is a decision $\mathsf{d} \in \{Accept, Reject\}$. V accepts P if and only if, $\mathsf{d} = Accept$.

The above protocol (P, V) in both general and identity-based identification schemes is often called a canonical protocol. We say that (x, c, y) is a valid transcript corresponding to pk_K and ID if it satisfies the equation above in Step 4. Note that most identification schemes are transformable to or from digital signature schemes. This is a quick way to construct an identification scheme; however, the security in these schemes is not intended to provide security against active and concurrent attacks.

3.6.2 Security of Identity-based Identification Scheme

We consider a security notion as those defined in [BNN04, KH04], which was an adaptation of the notion first proposed by Feige, Fiat and Shamir [FFS88] for an ID-based setting. That is, we consider three types of attack on an honest, private key equipped prover, namely, passive attack, active attack and concurrent attack. Generally, a two-phase game is considered between a challenger and an adversary. In a standard identification model, the above attacks should take place and be completed before an impersonation attempt, i.e. the attacks should be completed in Phase 1. However, in an identity-based setting, it is natural to allow an adversary to interact with real provers with identities other than the challenge identity ID_* even in Phase 2. In passive attacks, an adversary can eavesdrop and is in possession of transcripts of conversations between the provers and verifiers. In active and concurrent attacks, the adversary first plays the role of a cheating verifier, interacting with the provers several times before an impersonation attempt, even in Phase 2 (for provers with identities $\neq ID_*$).

A two-phase attack game between a passive or active/concurrent impersonator $\mathcal{A}$ and a challenger is described below.

Key Generation:

The challenger takes as input ℓ and runs the setup algorithm S. It gives $\mathcal{A}$ the resulting public parameter pk_K and keeps the master-key to itself.

Phase 1:

1. $\mathcal{A}$ issues some key extraction queries $ID_1, ID_2, ...$ The challenger responds by running the extraction algorithm $Extract$ to generate the private key d_i corresponding to the public identity ID_i. It returns d_i to $\mathcal{A}$. Let $\mathfrak{ID}$ be the list that keep the extraction queries. The challenger updates for every query for (ID_i, d_i) to $\mathfrak{ID}$.
2. $\mathcal{A}$ issues some transcript queries (in a passive attack) or some identification queries on ID_j (in an active/concurrent attack).
3. The queries in Step 1 and Step 2 above can be interleaved and asked adaptively. Without loss of generality, we may assume that $\mathcal{A}$ will not query the same ID_i that has been issued in the key extraction queries, the transcript queries or identification queries again.

Phase 2:

1. $\mathcal{A}$ outputs a challenge identity $ID_* \notin \mathfrak{ID}$, which it wishes to impersonate. Next, $\mathcal{A}$ plays the role of a cheating prover (an impersonation attempt on the prover holding the public identity ID_*), trying to convince the verifier.
2. $\mathcal{A}$ can still issue some key extraction queries as well as transcript queries or identification queries in Phase 2, with the restriction that no queries on the challenged identity ID_* are allowed.

We say that $\mathcal{A}$ succeeds in impersonating if it can make the verifier accept it.

Definition 3.10 *We say that an identity-based identification scheme IBI is (t, q, ϵ)-secure under passive (active and concurrent) attacks if for any passive (active and concurrent) impersonator $\mathcal{A}$ who runs in time t,*

$$\Pr[\mathcal{A} \textit{ can impersonate}] < \epsilon,$$

where $\mathcal{A}$ can make at most q key extraction queries.

3.6.3 Kurosawa-Heng Identity-based Identification without Random Oracles Scheme

The Kurosawa-Heng identity-based identification scheme (KH-IBI) [KH05] is the first identity-based identification scheme, which is provably secure against impersonation under active and concurrent attacks in the standard model. KH-IBI schemes are derived from the Boneh- Boyen signature scheme [BB04], a scheme that is secure against existential unforgeability under adaptive chosen message attack in the standard model, based on the Strong Diffie-Hellman (SDH) assumption.

Identity-Based Identification Scheme Secure against Passive Attack (KH-IBI-P)

Protocol Description

1. **Key Generation ($KeyGen$):**
   Given a security parameter ℓ, which is a positive integer, $KeyGen$ works as follows.

   (a) Take a security parameter 1^ℓ as input and select a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Then run the GDH Parameters Generator to obtain the system parameter $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, H\}$.

   (b) Randomly choose integers $x, y \in \mathbb{Z}_q^*$ such that $u = g_2^x \in \mathbb{G}_2$ and $v = g_2^y \in \mathbb{G}_2$.

   Here, a pair of secret and public parameter is generated where $pk_K = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, u, v, H)$ is the public parameter and $sk_K = (x, y)$ is the master private key of the key generator centre (KGC).

2. **Key Extraction ($Extract$):**
   Given the identity of the prover ($ID \in \{0,1\}^*$), the public parameter pk_K and the master private key sk_K, $Extract$ takes (ID) and (x, y) as input and computes a witness instance sk_P as follows.

   - Randomly choose $s \in \mathbb{Z}_q^*$ and compute $W_1 = g_1^{1/(x+H(ID)+ys)} \in \mathbb{G}_1$.
   - If $(x + H(ID) + ys) \bmod p = 0$, which is unlikely, then randomly reselect $s \in \mathbb{Z}_q^*$ and computes W_1 again, as in the previous step.

After this, the user's public key $pk_P = ID$ and user's private keys $sk_P = (W_1, s)$ are given to prover.

3. **Identification Protocol ($\langle P, V \rangle$):**
   A canonical protocol of an identity-based identification scheme can be formalised by $CID = (Commit, Response, Check)$. $Commit$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier.

   - Step 1. P chooses a random generator $r \in \mathbb{G}_1$ and computes $Commit(r, s_1) = (R, s)$ such that $R = \hat{e}(r, u \cdot g_2^{H(ID)} \cdot v^s)$. P then sends (R, s) to V.
   - Step 2. V chooses a random challenge integer c, $c \in \mathbb{Z}_p^*$ and sends it to P.
   - Step 3. P computes a response $Response(sk_P, y, c) = Z = r + {W_1}^c \in \mathbb{G}_2$ and sends Z to V.
   - Step 4. V checks if $R \cdot \hat{e}(g_1, g_2)^c \equiv \hat{e}(Z, u \cdot g_2^{H(ID)} \cdot v^s)$ then $Check(pk_K, R, ID, c, Z) \rightarrow Accept$, otherwise $Check(pk_K, R, ID, c, Z) \rightarrow Reject$. V accepts P if and only if, $\mathsf{d} = Accept$.

Identity-Based Identification Scheme Secure against Active and Concurrent Attack (KH-IBI-AC)

1. **Key Generation ($KeyGen$):**
   Given a security parameter ℓ, which is a positive integer,$KeyGen$ works as follows.

   (a) Take the security parameter 1^ℓ as input and select a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Then run the GDH Parameters Generator to obtain the system parameter $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, H\}$.

   (b) Randomly choose integers $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q^*$ such that $u_1 = g_2^{x_1}, u_2 = g_2^{x_2} \in \mathbb{G}_2$ and $v_1 = g_2^{y_1}, v_2 = g_2^{y_2} \in \mathbb{G}_2$.

   Here, a pair of secret and public parameter is generated where $pk_K = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, u_1, u_2, v_1, v_2, H)$ is the public parameter and $sk_K = (x_1, x_2, y_1, y_2)$ is the master private key of key generator centre (KGC).

2. **Key Extraction (*Extract*):**
   Given the identity of the prover ($ID \in \{0,1\}^*$), the public parameter pk_K and the master private key sk_K, *Extract* takes ID and (x, y) as input and computes a witness instance sk_P as follows.

   - Randomly choose $s_1, s_2 \in \mathbb{Z}_q^*$ and compute $W_1 = g_1^{1/(x_1+H(ID)+y_1 \cdot s_1)}, W_2 = g_1^{1/(x_2+H(ID)+y_2 \cdot s_2))} \in \mathbb{G}_1$.
   - If $(x_1 + H(ID) + y_1 \cdot s_1) \bmod p$ or $(x_2 + H(ID) + y_2 \cdot s_2) \bmod p$ are equal to 0, which is unlikely, then randomly reselect s_1 or (and) $s_2 \in \mathbb{Z}_q^*$ and computes W_1 or (and) W_2 again as in the previous step.

   After this, the user's public key (ID) and user's private keys (W_1, W_2, s_1, s_2) are given to the prover.

3. **Identification Protocol ($\langle P, V \rangle$):**
   A canonical protocol of an identity-based identification scheme can be formalised by $CID = (Commit, Response, Check)$, where *Commit*, *Response* and *Check* are PPT algorithms used in the following protocol where P is the prover and V is the verifier. Assume that P uses W_1, s_1 as witnesses in the following protocol:

   - Step 1. P chooses random generators $r, Z_2 \in \mathbb{G}_1$ and a random integer $c_2 \in \mathbb{Z}_p^*$ and computes $Commit(r, l, s_1, s_2) = (R_1, R_2, s_1, s_2)$, where $R_1 = \hat{e}(r, u_1 \cdot g_2^{H(ID)} \cdot v_1^{s_1}) \in \mathbb{G}_T$ and $R_2 = \hat{e}(Z_2, u_2 \cdot g_2^{H(ID)} \cdot v_2^{s_2}) / \hat{e}(g_1, g_2)^{c_2} \in \mathbb{G}_T$. P then sends (R_1, R_2, s_1, s_2) to V.
   - Step 2. V chooses a random challenge integer c, $c \in \mathbb{Z}_p^*$ and sends it to P.
   - Step 3. P first computes $c_1 = c - c_2 \bmod p$ and then computes a response $Response(W_1, r, Z_2, c, c_1, c_2) = Z_1 = r + W_1^{c_1} \in \mathbb{G}_2$. It then sends (Z_1, Z_2, c_1, c_2) to V.
   - Step 4. V checks if $c = c_1 + c_2 \bmod p$, $R_1 \cdot \hat{e}(g_1, g_2)^{c_1} \equiv \hat{e}(Z, u_1 \cdot g_2^{H(ID)} \cdot v_1^{s_1})$ and $R_2 \cdot \hat{e}(g_1, g_2)^{c_2} \equiv \hat{e}(Z, u_2 \cdot g_2^{H(ID)} \cdot v_2^{s_2})$ then $Check(pk_K, R, ID, c, Z) \rightarrow Accept$, otherwise $Check(pk_K, R, ID, c, Z) \rightarrow Reject$. V accepts P if and only if, $\mathsf{d} = Accept$.

Chapter 4

Universal Designated Verifier Signature Schemes

In this chapter, two variants of universal designated verifier signature schemes called "one-time universal designated verifier signature" and "universal designated verifier signature with threshold-signers" are described. Part of this chapter appeared in *ICISC 2008* [TSM08] and *IWSEC 2009* [TSM09c].

4.1 Introduction

Invented by Jakobsson, Sako and Impagliazzo in [JSI96], a designated verifier signature is a signature that provides not only authentication of a message but also provides the deniability property allowing the signer to deny the signature. Besides, the (designated) verifier can also generate such a signature all by himself/herself. In other words, the only person who will be convinced of the authenticity of the signature on the message is the designated verifier since this signature can be constructed either by the signer or by himself/herself.

Extended from the above signature scheme, a universal designated-verifier signature proposed by Steinfeld, Bull, Wang and Pieprzyk in [SBWP03] is a signature that allows a party called "a signature holder" to generate a designated verifier signature from a regular signature given by the signer. Basically, the properties of this signature are identical to the designated verifier signature. The only additional functionality of this signature is a signature holder's privilege to designate the signature to any verifier arbitrarily chosen by him. Since then, there are many studies in the security research that extends the above notion, including the works presented in this Chapter. The first contribution of our work in this chapter is the notion of one-time universal designated verifier signature. In this notion, the signer is allowed to limit his/her signature to be used to generate a designated verifier signature only

one time. A signature holder is permitted to compute only one designated verifier signature from the signer's signature, otherwise, the signer's signature can be revealed. In the following, we provide a scenario describing our notion in the real life.

A doctor, Susan informs her patient, Henry that he is positive for some sensitive disease such as cancers or AIDS. Since Henry may not want to reveal his identity when he brings the prescription to get medication or when he needs to see the specialist for special treatment, Susan may provide a statement for Henry such that Henry can show it to pharmacy or a specialist. However, this statement should be used only one time due to the medical restriction. These are three properties required for the above scenario. There are *the privacy* of the patient, *the authenticity* provided by Susan and *the restriction* of signature usage. Henry wants to protect his privacy. Susan wants the authenticity and the restriction of signature usage to provide an authenticated statement to Henry such that this authenticated statement can be used only once or, otherwise, the information about the signature is revealed and hence, Henry's privacy will not be protected any longer. The above scenario cannot be solved by a universal designated verifier signature but the one-time universal designated verifier signature is the solution. We note that the above scenario can be converted to its associated digital scenario counterpart, where the signatures refer to digital signatures instead.

The second contribution of our work in this chapter is the notion of universal designated verifier signature with threshold signer. In this notion, privacy is provided for both the signer and the signature holder. The signature holder is allowed to provide anonymity for the signer(s) and the signature(s) that he has in his possession. In other words, a designated verifier is convinced that a designated verifier signature is constructed by the signature(s) from a signer (or t signers) in the list of n signers. In the following, a scenario describing our notion in real life is provided. This scenario is motivating from the law-suit. The jurors want to confirm a conviction statement to a judge. Information of the jurors' identity and their convictions stating guilty or not guilty must be protected. The only information that can be revealed is the statement that at least t out of n jurors voted along with this conviction statement. This statement may be designated to a judge so that only he/she can know and can be convinced that it is voted by at least t out of n jurors. Another scenario in the law-suit is when a witness(es) wants to give a statement of the crime that he/she (they) witnessed. In this case, a lawyer or district attorney will act as

a middle man to deliver the statement to a judge. However, the privacy between witness(es) and judge, the anonymity of witness(es), and authenticity of witness(es) cannot be guaranteed in the traditional method. The notion of universal designated verifier signature with threshold signer provides the privacy between witnesses (as signers) and a judge (as a verifier), and the anonymity and authenticity of witness in the above scenario.

4.1.1 Related Work

Since the novel invention of designated verifier signatures proposed by Jakobsson, Sako and Impagliazzo in [JSI96] and the novel extension of universal designated verifier signatures proposed by Steinfeld, Bull, Wang and Pieprzyk in [SBWP03], the topics of designated verifier signatures and universal designated verifier signatures areas have been extensively studied. These include [LV07a, LV04, LLP05, LWB05, HMSZ05, SZM04].

In ACNS'05, Zhang, Furukawa and Imai [ZFI05] were the first to propose a universal designated verifier signature scheme in the standard model. Later, two universal designated verifier signature schemes were proposed by Laguillaumie, Libert and Quisquater in [LLQ06]. Their schemes have been proven in the standard model and they claimed that their schemes are more efficient than previous works[LLQ06]. In UIC'06, the restricted universal designated verifier signature scheme was proposed by Huang et al. [HSMZ06]. This restricted universal designated verifier signature scheme limits power of a signature holder so that a signature holder can generate a designated verifier signature from a signer's signature for only k times. If designated verifier signatures were generated more than k times, then these k designated verifier signatures can be used to deduce a standard signature. However, in their scheme setting, the party who has a privilege to assign the limited number (k), is a signature holder himself/herself. Hence, by possessing the power to the limited number (k) of designated verifier, a signature holder does not need to process with the scheme requirement. Therefore, this scheme is not practical compared with our notion, which a signer is a party to assign the limited number (k) of designated verifier where $k = 1$. However, in ISC'07, Laguillaumie and Vergnaud [LV07b] pointed out that the restricted universal designated verifier signature scheme in [HSMZ06] has failed to achieve the restriction property as it should be. They demonstrated that the restricted designated verifier signature is no longer restricted since the signature

holder can provided a proof of possession of signature by using zero knowledge proof.

The attack of universal designated verifier signature, which is widely discussed, is delegatability attack. This was originally inspired by Lipmaa et al. in [LWB05]. This attack is a case when the signer or verifier reveals some information without uncovering his/her private key such that a signature holder can compute a designated verifier signature on any message of choice. They claim that non-delegatability property is needed for universal designated verifier signature and some examples of applications that needed this notion were given. This includes hypothetical e-voting protocol. They also provide the claim proposed by Steinfeld, Bull, Wang and Pieprzyk in [SBWP03] is delegatable.

Compared to the notion of ring signatures introduced and formalised by Rivest, Shamir and Tauman in [RST01], the notion of universal designated verifier signature with threshold signers provides not only the integrity of the message, and the authenticity, non-repudiation and anonymity of the signer, but the privacy. In this notion, a key to achieve the anonymity of the signer is signer-ambiguity. Without cooperating with other signers, a signer alone can compute a signature that has been signed by one of the signers in a set of signers (a ring of signers). Therefore, a verifier can be convinced that the authentication of a message is generated by one signer in the ring. Hence, the signer remains anonymous to everyone. The ring signatures topic has been widely studied by many researchers, including [BSS02, TWC$^+$04, HS03, ZK02, BKM06, SW07, FS07, LW04, LWW03]. The notion of threshold ring signatures introduced and formalised by Bresson, Stern and Szydlo in [BSS02] is the closest feature to the notion of ring signature. In this notion, multi signers allow constructing a signature with multi signers-ambiguity property. The threshold ring signatures topics have been widely studied by many researchers, including [TWC$^+$04, LW04, LWW03, IT05, CHY04].

The closest feature to our notion of universal designated verifier signature with threshold signers is the primitive notion called universal designated verifier ring signature proposed by Li and Wang [LW06]. In this notion, the signature holder can designate a ring signature to a specific verifier. It seems that this scheme has met the requirement of our notion where $t = 1$, nevertheless, it is argued that the goals are different. In their scheme, a ring signature on a message is provided by a signer, but not by a signature holder. Hence, a signature holder does not know exactly who signed the message. Compared to our notion of universal designated verifier signature with threshold signers, a signature holder knows who signs the

message. Moreover, the signature holder should be the one who does not want to reveal which signer's signature he possessed and hence, he may just want to prove that he possessed some signatures.

From the above, it is stated that none of the existing primitives achieve the requirements needed as stated in the above motivating scenarios. Therefore, we are the first to propose and formalise new notions called "one-time universal designated verifier signature" and "universal designated verifier signature with threshold signers".

4.1.2 Our Contributions

In this chapter, the notion of the one-time universal designated verifier signature (OT-UDVS) scheme and universal designated verifier signature with threshold-signers (TS-UDVS) schemes are introduced. A model of the OT-UDVS scheme is provided together with its security notions. We present a concrete construction scheme of the OT-UDVS scheme and its security analysis. The notion of threshold-signers universal designated verifier signature (TS-UDVS) schemes to capture the above requirements is provided. A model of the TS-UDVS scheme and its security notions to capture the integrity of a message, the authenticity, non-repudiation, privacy and anonymity of the signers is also provided. A concrete scheme is also presented, together with its security proof to show that our scheme is secure in our model. This is the first time this kind of primitive scheme has been introduced into the literature.

Chapter Organisation

This chapter is organised as follows. In Section 4.2, the definition of OT-UDVS and its security model will be presented. In Section 4.3, the OT-UDVS scheme is presented. Then, the security proof of our concrete scheme is presented in Section 4.4. The definition of TS-UDVS and its security notations will be described in Section 4.5. Next, our TS-UDVS scheme will be given in Section 4.6. Proof of the security of our concrete scheme is described in Section 4.7 followed by the chapter conclusion.

4.2 Definition of One-Time Universal Designated Verifier Signatures

In this section, a definition of a one-time universal designated verifier signature (OT-UDVS) scheme, which allows a signer to limit the usage of his/her signature used by a signature holder is given.

4.2.1 Outline of OT-UDVS

First of all, it must be assumed that all parties must comply with a registration protocol with a certificate authority (CA) to obtain a certificate on their public parameters. A one-time universal designated verifier signature scheme Σ is a 8-tuple ($SKeyGen$, $Sign$, $Verify$, $VKeyGen$, $Delegate$, $DVerify$, $DSimulate$, $Open$). The definition of $SKeyGen$, $Sign$, $Verify$, $VKeyGen$, $Delegate$, $DVerify$ and $DSimulate$ can be found in Section 3.3 where param is 1^ℓ. The $Open$ algorithm is described as follows.

Opening a Delegated Signature ($Open$): This is a probabilistic algorithm that, given two designated verifier signatures $\hat{\rho}, \tilde{\rho}$ and their designated verifiers' public parameters $pk_{\widehat{V}}, pk_{\widetilde{V}}$, signer's public parameter pk_S and a message M as input, outputs a signer signature σ. That is,

$$Open(\hat{\rho}, \tilde{\rho}, pk_{\widehat{V}}, pk_{\widetilde{V}}, pk_S, M) \rightarrow \sigma.$$

4.2.2 Completeness

Remark: For all $\ell \in \mathbb{N}$, all $(pk_S, sk_S) \in SKeyGen(1^\ell)$, all $(pk_V, sk_V) \in VKeyGen(1^\ell)$ and all messages M, a one-time universal designated verifier signature must satisfy the following properties:

Completeness of a Signature:

$$\forall \sigma \in Sign(M, sk_S, pk_S), \mathbf{Pr}[Verify(M, \sigma, pk_S) = Accept] = 1. \tag{4.1}$$

Completeness of a Designated Verifier Signature:

$$\begin{aligned} &\forall \rho \in Delegate(M, \sigma, pk_V, pk_S), \\ &\mathbf{Pr}[DVerify(M, \rho, pk_V, sk_V, pk_S) = Accept] = 1. \end{aligned} \tag{4.2}$$

Completeness of a Simulated Designated Verifier Signature:

$$\begin{aligned}&\forall \varrho \in DSimulate(M, pk_V, sk_V, pk_S),\\ &\mathbf{Pr}[DVerify(M, \varrho, pk_V, sk_V, pk_S) = Accept] = 1.\end{aligned} \tag{4.3}$$

Completeness of an Opened Signature:

$$\begin{aligned}&\forall \hat{\rho} \in Delegate(M, \sigma, pk_{\widehat{V}}, pk_S); \forall \tilde{\rho} \in Delegate(M, \sigma, pk_{\widetilde{V}}, pk_S) :\\ &DVerify(M, \hat{\rho}, pk_{\widehat{V}}, sk_{\widehat{V}}, pk_S) = Accept;\\ &DVerify(M, \tilde{\rho}, pk_{\widetilde{V}}, sk_{\widetilde{V}}, pk_S) = Accept;\\ &\sigma \leftarrow Open(\hat{\rho}, \tilde{\rho}, pk_{\widehat{V}}, pk_{\widetilde{V}}, pk_S, M),\\ &\mathbf{Pr}[Verify(M, \sigma, pk_S) = Accept] = 1.\end{aligned} \tag{4.4}$$

A signature is referred to as 'open' if there are two valid designated signatures dedicated to two different designated verifiers issued by the signature holder.

Source Hiding:

$$\begin{aligned}&\forall \rho \in Delegate(M, \sigma, pk_V, pk_S); \forall \varrho \in DSimulate(M, pk_V, sk_V, pk_S) :\\ &DVerify(M, \rho, pk_V, sk_V, pk_S) = Accept;\\ &DVerify(M, \varrho, pk_V, sk_V, pk_S) = Accept;\\ &\phi \xleftarrow{\$} \{\rho, \varrho\}, |\mathbf{Pr}[\phi = \rho] - \mathbf{Pr}[\phi = \varrho]| \ is \text{ negligible},\\ &\text{when taken all possible choice of } \rho \text{ and } \varrho.\end{aligned} \tag{4.5}$$

In the following subsections, the other security properties of one-time universal designated verifier signatures will be discussed in detail. These include the unforgeability, the single designatability and the non-transferability privacy (which implies the source hiding property). The unforgeability and the non-transferability privacy are introduced in [SBWP03]. The single designatability is introduced in [TSM08] to capture the requirement of the OT-UDVS scheme. The single designatability property captures the sense that no signature holder can convince more than one designated verifier or produce more than one designated verifier signature. The original signature will be revealed, as a proof of this misbehaviour, if the signature holder did anything against the above condition.

4.2.3 Unforgeability

A chosen public key attack plays an important role that captures a collusion attack launched by a malicious signature holder, malicious signers and malicious verifiers. The chosen public key attack is to simulate a situation that, in addition to the target signer and the target verifier, an adversary possesses the knowledge of secret keys of other signers or verifiers, and designated verifier signatures prior to the attack. This reflects the collusion attack where collusion happens among a malicious signature holder, malicious signers and malicious verifiers.

Unforgeability under adaptive chosen message and chosen public key attack are formally given as follows. The unforgeability property is intentionally to prevent an adversary corrupted with signature holder to generate an (OT-U)DVS signature ρ_* on a new message M^*. Given an access with its arbitrary choice of the verifier's public parameter pk_V to signing oracle, delegation oracle, and (designated verifier) verification oracle, a choice of message M and the signer's public parameter pk_S as input, the unforgeability provides an assurance that no adversary can produce a designated verifier signature on a new message. Denote $EUF\text{-}OT\text{-}UDVS$, the existential unforgeability of OT-UDVS scheme and $CM\text{-}CPK\text{-}A$, the adaptively chosen message and chosen public key attack. Then we define $\mathcal{A}^{CM-CPK-A}_{EUF-OT-UDVS}$ as the adaptively chosen message and chosen public key adversary and let $\mathcal{F}$ be a simulator. The game between $\mathcal{F}$ and $\mathcal{A}$ that describes the existential unforgeability of OT-UDVS scheme can be defined as follows.

$\mathcal{SPO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{SP} times for a public key of signer. Then, $\mathcal{F}$ runs the $\mathcal{SPO}$ algorithm to generate a private key sk_S and a public parameter pk_S of signer. After that, $\mathcal{F}$ returns pk_S to $\mathcal{A}$.

$\mathcal{VSO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{VS} times for the verification of a signature σ on a message M with its corresponding public parameter pk_S. Then, in return, $\mathcal{F}$ runs the $Verify$ algorithm to verify a signature σ on a message M corresponding with pk_S. Finally, $\mathcal{F}$ returns *Accept* if a signature σ on a message M is valid regarding to pk_S, otherwise, it outputs *Reject*.

$\mathcal{VPO}$ oracle: First, $\mathcal{A}$ can make queries at most q_{VP} times for a verifier's public parameter pk_V. Then, in return, $\mathcal{F}$ runs the $VKeyGen$ algorithm to generate a private key sk_V and a public parameter pk_V of a verifier. $\mathcal{F}$ outputs pk_V and returns it to $\mathcal{A}$.

$\mathcal{DVO}$ oracle: $\mathcal{A}$ can make queries at most q_{DV} times for the verification of a designated verifier signature ρ (or ϱ) on its chosen message M with its corresponding public parameters pk_S, pk_V. Next, $\mathcal{F}$ runs the $DVerify$ algorithm to verify a designated verifier signature ρ (or ϱ) on a message M corresponding with pk_S, pk_V. Finally, $\mathcal{F}$ returns *Accept* if a designated verifier signature ρ (or ϱ) on a message M is valid regarding to pk_S, pk_V, otherwise, it returns *Reject*.

$\mathcal{SDO}$ oracle: Given choices of a signer's public parameter pk_S and a verifier's public parameter pk_V, $\mathcal{A}$ can make queries at most q_{SD} times for a (simulated) designated verifier signature ϱ on its choice of message M, which ϱ must indeed generated by verifier. Next, $\mathcal{F}$ runs the $DSimulate$ algorithm to generate a (simulated) designated verifier signature ϱ on a message M corresponding with pk_S, pk_V. Finally, $\mathcal{F}$ outputs ϱ, M and returns them to $\mathcal{A}$.

$\mathcal{OPO}$ oracle: Given choices of a signer's public parameter pk_S and verifiers' public parameters $pk_{\widehat{V}}, pk_{\widetilde{V}}$, $\mathcal{A}$ can make queries at most q_{OP} times to open a signature σ on a message M from two designated verifier signatures $\hat{\rho}, \tilde{\rho}$. In return, $\mathcal{F}$ runs the *Open* algorithm to open a signature σ on a message M from $\hat{\rho}, \tilde{\rho}$ and its corresponding $pk_S, pk_{\widehat{V}}, pk_{\widetilde{V}}$. Finally, $\mathcal{F}$ returns σ if $\hat{\rho}, \tilde{\rho}$ signatures on a message M is valid regarding to $pk_S, pk_{\widehat{V}}, pk_{\widetilde{V}}$, otherwise, it returns *Reject*.

$\mathcal{SKO}$ oracle: $\mathcal{A}$ can make queries at most q_{SK} times for a private key sk_S (or sk_V) corresponding to a public parameter pk_S (or pk_V) of the signer (or the verifier). Then, $\mathcal{F}$ returns a corresponding private key sk_S (or sk_V) to $\mathcal{A}$.

Note that, for the definition of the $\mathcal{SSO}$ and $\mathcal{DSO}$ oracles, the reader may refer to Section 3.3.1. Then, we assume that, given the public parameters pk_S^*, pk_V^* as input, $\mathcal{A}$ returns a forged signature ρ_* on a message M^*. We say that $\mathcal{A}$ wins the game if

1. $Accept \leftarrow DVerify(M^*, \rho_*, pk_V^*, sk_V^*, pk_S^*)$.
2. Neither pk_S^* nor pk_V^* has been submitted as input of a query for a private key to the $\mathcal{SKO}$ oracle.
3. With M^*, pk_S^* as input, $\mathcal{A}$ never made a request for a signature to the $\mathcal{SSO}$ oracle.

4. With M^*, pk_S^* as input, $\mathcal{A}$ never made any request for a designated verifier signature to the $\mathcal{DSO}$ oracle.

5. With M^*, pk_V^* as input, $\mathcal{A}$ never made any request for a designated verifier signature to the $\mathcal{SDO}$ oracle.

Let $Succ^{CM-CPK-A}_{EUF-OT-UDVS}(.)$ be defined as the success probability of that $\mathcal{A}^{CM-CPK-A}_{EUF-OT-UDVS}$ wins the above game.

Definition 4.1 *One-time universal designated verifier signature scheme is (*$\mathtt{t}$*,*q_H*, *q_{SP}*,*q_{SS}*,*q_{VS}*,*q_{VP}*,*q_{DS}*,*q_{DV}*,*q_{SD}*,*q_{OP}*,*q_{SK}*,*ϵ*)-secure against existential unforgeability under a chosen message and chosen public key attack if there is no PPT CM-CPK-A adversary $\mathcal{A}^{CM-CPK-A}_{EUF-OT-UDVS}$ such that the success probability $Succ^{CM-CPK-A}_{EUF-OT-UDVS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-CPK-A}_{EUF-OT-UDVS}$ runs in time at most $\mathtt{t}$, make at most q_H, q_{SP}, q_{SS}, q_{VS}, q_{VP}, q_{DS}, q_{DV}, q_{SD}, q_{OP}, and q_{SK} queries to the random oracle $\mathcal{HO}$, the $\mathcal{SPO}$ oracle, the $\mathcal{SSO}$ oracle, the $\mathcal{VSO}$ oracle, the $\mathcal{VPO}$ oracle, the $\mathcal{DSO}$ oracle, the $\mathcal{DVO}$ oracle, the $\mathcal{SDO}$ oracle, the $\mathcal{OPO}$ oracle, and the $\mathcal{SKO}$ oracle, respectively.*

4.2.4 Non-transferability Privacy

The source hiding property is provided that, given the verifier's public parameter pk_V, the verifier's private key sk_V, the signer's public parameter pk_S, and a message M, one can compute a (simulated) designated verifier signature indistinguishable from a designated verifier signature generated by a signature holder. The above property does not imply the non-transferability privacy property introduced in [SBWP03] and elaborated in [HSMW06].

The non-transferability privacy property is different from the source hiding property where even one can obtain or review many designated verifier signatures $\rho_1, ..., \rho_q$ on the same message M designated to either same or different verifiers. In addition, the designated verifier signatures $\rho_1, ..., \rho_q$ are also generated by the same signature holder using the same signature σ. However, the non-transferability privacy property for one-time universal designated verifier signature schemes is different from above description. A signature holder can generate only one designated verifier signature ρ per message per verifier or else the original signature σ is revealed when one obtains two or more designated verifier signatures generated from the same signature σ. Therefore, the non-transferability privacy property for one-time universal

designated verifier signature schemes claims that (1) it is implies the source hiding property and, (2) even if an adversary obtains or investigates many designated verifier signatures $\rho_1, ..., \rho_q$ on its choices of message $M_1, ..., M_q$ designated to either same or different verifiers, he/she cannot convince other party that a signer is the one who is responsible for generating a signature σ related to a designated verifier signature $\rho \in \{\rho_1, ..., \rho_q\}$ on a message $M \in \{M_1, ..., M_q\}$.

Let $ENT\text{-}OT\text{-}UDVS$ be the existential non-transferable privacy of a one-time universal designated verifier signature scheme. Let $\mathcal{A}^{CM-CPK-A}_{ENT-OT-UDVS}$ be the adaptively chosen message and chosen public key distinguisher. Let $\mathcal{F}$ be a simulator. The game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential non-transferable privacy of one-time universal designated verifier signature scheme defined as follows.

The game is separated into two phases. Start with $\mathcal{F}$ constructs the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VSO}}$, $\boldsymbol{\mathcal{VPO}}$, $\boldsymbol{\mathcal{DSO}}$, $\boldsymbol{\mathcal{DVO}}$, $\boldsymbol{\mathcal{SDO}}$, $\boldsymbol{\mathcal{OPO}}$ and $\boldsymbol{\mathcal{SKO}}$ oracles, which are defined in Section 4.2.3.

The game is then run as follows.

1. **Phase** 1**:** $\mathcal{A}$ is allowed to make a request to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VSO}}$, $\boldsymbol{\mathcal{VPO}}$, $\boldsymbol{\mathcal{DSO}}$, $\boldsymbol{\mathcal{DVO}}$, $\boldsymbol{\mathcal{SDO}}$, $\boldsymbol{\mathcal{OPO}}$ and $\boldsymbol{\mathcal{SKO}}$ oracles. The oracles answer as their design.

2. **Challenge:** When $\mathcal{A}$ is ready to challenge $\mathcal{F}$, it outputs M^*, pk_S^*, pk_V^* with the constraints that

   a. With M^*, pk_S^* as input, $\mathcal{A}$ never made a request for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. With M^*, pk_S^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{DSO}}$ oracle.

   c. With M^*, pk_V^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{SDO}}$ oracle.

   d. With pk_S^* as input, $\mathcal{A}$ never submitted a request for a private key to the $\boldsymbol{\mathcal{SKO}}$ oracle.

   In response, $\mathcal{F}$ selects a bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$ then $\mathcal{F}$ outputs $\rho \leftarrow Delegate(M^*, \sigma, pk_V^*, pk_S^*)$ from the $\boldsymbol{\mathcal{DSO}}$ oracle to $\mathcal{A}$. Otherwise, $\mathcal{F}$ outputs $\varrho \leftarrow DSimulate(M^*, pk_V^*, sk_V^*, pk_S^*)$ from the $\boldsymbol{\mathcal{SDO}}$ oracle to $\mathcal{A}$.

3. **Phase 2:** $\mathcal{A}$ is allowed to return arbitrarily to *Phase* 1 or *Challenge* as many times as it wants. However, there is a condition that $\mathcal{A}$ must have at least one set of challenges M^*, pk_S^*, pk_V^* such that

   a. With M^*, pk_S^* as input, $\mathcal{A}$ never submitted a request for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. With M^*, pk_S^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{DSO}}$ oracle.

   c. With M^*, pk_V^* as input, $\mathcal{A}$ never submitted a request for a designated verifier signature to the $\boldsymbol{\mathcal{SDO}}$ oracle.

   d. With pk_S^* as input, $\mathcal{A}$ never submitted a request for a corresponding private key sk_S^* to the $\boldsymbol{\mathcal{SKO}}$ oracle.

4. **Guessing:** Finally, $\mathcal{A}$ responds with a guess b', which is corresponding to the challenge M^*, pk_S^*, pk_V^*. The distinguisher wins the game if $b = b'$.

Let $Succ^{CM-CPK-A}_{ENT-OT-UDVS}(.)$ be the success probability of that $\mathcal{A}^{CM-CPK-A}_{ENT-OT-UDVS}$ wins the above game.

Definition 4.2 *One-time universal designated verifier signature scheme is (*$\mathtt{t}$*,*q_H*, *q_{SP}*,*q_{SS}*,*q_{VS}*,*q_{VP}*,*q_{DS}*,*q_{DV}*,*q_{SD}*,*q_{OP}*,*q_{SK}*,*ϵ*)-secure against existential non-transferable privacy under a chosen message and chosen public key attack if there is no PPT CM-CPK-A distinguisher $\mathcal{A}^{CM-CPK-A}_{ENT-OT-UDVS}$ such that the success probability $Succ^{CM-CPK-A}_{ENT-OT-UDVS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-CPK-A}_{ENT-OT-UDVS}$ runs in time at most $\mathtt{t}$, make at most q_H, q_{SP}, q_{SS}, q_{VS}, q_{VP}, q_{DS}, q_{DV}, q_{SD}, q_{OP}, and q_{SK} queries to the random oracle $\boldsymbol{\mathcal{HO}}$, the $\boldsymbol{\mathcal{SPO}}$ oracle, the $\boldsymbol{\mathcal{SSO}}$ oracle, the $\boldsymbol{\mathcal{VSO}}$ oracle, the $\boldsymbol{\mathcal{VPO}}$ oracle, the $\boldsymbol{\mathcal{DSO}}$ oracle, the $\boldsymbol{\mathcal{DVO}}$ oracle, the $\boldsymbol{\mathcal{SDO}}$ oracle, the $\boldsymbol{\mathcal{OPO}}$ oracle, and the $\boldsymbol{\mathcal{SKO}}$ oracle, respectively.*

4.2.5 Single Designatability

The single designatability property is similar to the opening property introduced in [HSMW06] and analysed by Laguillaumie and Vergnaud in [LV07b]. The multi-time restricted delegation was mainly discussed in the above works. Nevertheless, Laguillaumie and Vergnaud claimed that there is always proof for the restricted UDVS scheme in [HSMW06], which a signature holder can generate a designated

verifier signature from a signature without getting a penalty for over spending. For one-time universal designated verifier signature schemes, the single designatability property restricts the signature holder such that he/she can only convince one verifier with one designated verifier signature without being penalised.

The single designatability property provides security against existential single designatability's adversary under adaptive chosen message and chosen public key attack. Generally, it prevents an adversary corrupted with signature holder to generate two (one-time universal) designated verifier signatures $\hat{\rho}, \tilde{\rho}$ on a message M such that both signatures are valid on the same message generated by the same signature holder. However, both designated verifier signatures could not be opened to reveal an original signature σ generated by the signer.

Let $ESD\text{-}OT\text{-}UDVS$ be the existential single designatability of one-time universal designated verifier signature scheme. We also denote by $\mathcal{A}$ the adaptively chosen message and chosen public key adversary and denote by $\mathcal{F}$ a simulator. The game between $\mathcal{F}$ and $\mathcal{A}$ that describes the existential single designatability of one-time universal designated verifier signature scheme is illustrated as follows.

In the following descriptions, the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VSO}}$, $\boldsymbol{\mathcal{VPO}}$, $\boldsymbol{\mathcal{DSO}}$, $\boldsymbol{\mathcal{DVO}}$, $\boldsymbol{\mathcal{SDO}}$, $\boldsymbol{\mathcal{OPO}}$ and $\boldsymbol{\mathcal{SKO}}$ oracles are illustrated in Section 4.2.3. $\mathcal{A}$ is allowed to access arbitrarily to these oracles. Finally after the queries, with public parameters $pk_S^*, pk_{\widehat{V}}, pk_{\widetilde{V}}$ as input, $\mathcal{A}$ outputs two designated verifier signatures $\hat{\rho}, \tilde{\rho}$ on a message M^*. We denote by σ a signature produced by the $\boldsymbol{\mathcal{SSO}}$ oracle on input M^*, pk_S^*. It is said that $\mathcal{A}$ wins the above game if:

1. $(Accept \leftarrow DVerify(M^*, \hat{\rho}, pk_{\widehat{V}}, sk_{\widehat{V}}, pk_S^*)) \bigwedge$
   $(Accept \leftarrow DVerify(M^*, \tilde{\rho}, pk_{\widetilde{V}}, sk_{\widetilde{V}}, pk_S^*)) \bigwedge \sigma \nleftarrow Open(\hat{\rho}, \tilde{\rho}, pk_{\widehat{V}}, pk_{\widetilde{V}}, pk_S^*, M^*)$.

2. With pk_S^* as input, $\mathcal{A}$ never submitted a request for a corresponding private key sk_S^* to the $\boldsymbol{\mathcal{SKO}}$ oracle.

3. With $M^*, pk_S^*, pk_{\widehat{V}}$ as input, $\mathcal{A}$ can make only one request for a designated verifier signature to the $\boldsymbol{\mathcal{DSO}}$ oracle or, with $M^*, pk_S^*, pk_{\widetilde{V}}$ as input, to the $\boldsymbol{\mathcal{SDO}}$ oracle.

Let $Succ_{ESD-OT-UDVS}^{CM-CPK-A}(.)$ be the success probability of that $\mathcal{A}_{ESD-OT-UDVS}^{CM-CPK-A}$ wins the above game.

Definition 4.3 *One-time universal designated verifier signature scheme is (*$\mathtt{t}$,q_H, q_{SP},q_{SS},q_{VS},q_{VP},q_{DS},q_{DV}, q_{SD},q_{OP},q_{SK},ϵ*)-secure against existential single designatability adversary under a chosen message and chosen public key attack if there is no PPT adversary* $\mathcal{A}$ *such that the success probability* $Succ^{CM-CPK-A}_{ESD-OT-UDVS}(\ell) = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}$ *runs in time at most* $\mathtt{t}$*, make at most* q_H*,* q_{SP}*,* q_{SS}*,* q_{VS}*,* q_{VP}*,* q_{DS}*,* q_{DV}*,* q_{SD}*,* q_{OP}*, and* q_{SK} *queries to the random oracle* $\mathcal{HO}$*, the* $\mathcal{SPO}$ *oracle,* $\mathcal{SSO}$ *oracle, the* $\mathcal{VSO}$ *oracle, the* $\mathcal{DSO}$ *oracle, the* $\mathcal{DVO}$ *oracle, the* $\mathcal{SDO}$ *oracle, the* $\mathcal{OPO}$ *oracle, and the* $\mathcal{SKO}$ *oracle, respectively.*

4.3 The Proposed OT-UDVS Scheme

We denote by $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ a hash function that maps any string to group $\mathbb{G}_1$ and denote by $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ a collision-resistant hash function. Let $\mathbb{G}_1, \mathbb{G}_T$ be two groups of prime order p. Let $\hat{e}$ be an efficient computationally bilinear mapping function that map $\mathbb{G}_1$ to $\mathbb{G}_T$. The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The proposed scheme is described as follows.

SKeyGen: Given a security parameter ℓ as input, a signer S randomly selects a prime $p = poly(1^\ell)$ and a random generator $g \in \mathbb{G}_1$. Let $\mathsf{param} = (p, \hat{e}, g, H, h)$ be the system parameter. A private key and a public parameter are generated as follows. Select a random integer $x \in \mathbb{Z}_p$ and set $X = g^x$. Then, *SKeyGen* outputs $pk_S = (\mathsf{param}, X)$ as the public parameter of a signer and $sk_S = x$ as the private key of the signer.

VKeyGen: Given a security parameter ℓ as input, a verifier V runs a trapdoor commitment scheme's setup function $Setup(1^\ell)$ in Section 3.4 to obtain $l, g_l, Y = g_l^y, sk_V = y$. Denote $\bar{h} : \{0,1\}^* \rightarrow \mathbb{Z}_l^*$ a collision-resistant hash function selected by V. V then publishes $pk_V = (\mathsf{param} = (l, g_l, \bar{h}), Y)$ as his public key and keeps sk_V private.

Sign: Given sk_S, pk_S and a message M as input, S generates a signature σ on

message M as follows.

$$\begin{aligned} r_1 &\xleftarrow{\$} \mathbb{Z}_p, \\ \sigma_1 &= g^{r_1}, \\ M' &= M||\sigma_1||pk_S, \\ \sigma_2 &= H(M')^x, \\ \sigma_3 &= H(M')^{r_1}. \end{aligned}$$

The signature on message M is $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

Verify: Given pk_S, σ and a message M as input, a signature holder SH computes $M' = M||\sigma_1||pk_S$ at first and then checks whether

$$\begin{aligned} \hat{e}(\sigma_2, g) &\stackrel{?}{=} \hat{e}(H(M'), X) \bigwedge \\ \hat{e}(\sigma_3, g) &\stackrel{?}{=} \hat{e}(H(M'), \sigma_1) \end{aligned}$$

holds or not. If it is not, then it returns `reject`. Otherwise, it returns `accept`.

Delegate: Select a random integer $r_2 \in \mathbb{Z}_p$. With pk_V, σ and a message M as input, a signature holder SH generates a designate verifier signature ρ on message M as follows.

$$\begin{aligned} M' &= M||\sigma_1||pk_S, \\ T_V &= h(g_l^{r_2} Y^{\bar{h}(M')} \bmod l), \\ h_V &= h(pk_S||pk_V||M||T_V), \\ R' &= \sigma_2 \cdot \sigma_3^{h_V}, \\ \boldsymbol{\rho_1} &= \sigma_1, \\ \boldsymbol{\rho_2} &= \sigma_2^{T_V} \cdot R'. \end{aligned}$$

The designated verifier signature on message M is $\rho = (\boldsymbol{\rho_1}, \boldsymbol{\rho_2}, r_2)$.

DVerify: With pk_S, pk_V, sk_V, ρ and a message M as input, the designated verifier V computes $M' = M||\boldsymbol{\rho_1}||pk_S$, $T_V = h(g_l^{r_2} Y^{\bar{h}(M')})$, $h_V = h(pk_S||\ pk_V||M||T_V)$, and $R = X \cdot \boldsymbol{\rho_1}^{h_V}$. Next, V checks whether

$$\hat{e}(\boldsymbol{\rho_2}, g) \stackrel{?}{=} \hat{e}(H(M'), X^{T_V}) \hat{e}(H(M'), R)$$

holds or not. If not, then output `reject`. Otherwise, output `accept`.

DSimulate: With sk_V, pk_V, pk_S and a message M as input, V first randomly selects a generator $K \in \mathbb{G}_1$ and integers $k, \bar{r}_2 \in \mathbb{Z}_p$. Next, V computes as follows.

$$\begin{aligned}
\bar{M} &= M||K||pk_S, \\
T_V &= h(g_l^{\bar{r}_2} Y^{\bar{h}(\bar{M})}), \\
h_V &= h(pk_S||pk_V||M||T_V), \\
\boldsymbol{\rho_1} &= (g^k \cdot X^{-T_V} \cdot X^{-1})^{\frac{1}{h_V}}, \\
M' &= M||\sigma_{(1,SH)}||pk_S, \\
r_2 &= \bar{r}_2 + y \cdot \bar{h}(\bar{M}) - y \cdot \bar{h}(M'), \\
\boldsymbol{\rho_2} &= H(M')^k.
\end{aligned}$$

The designated verifier signature on message M is $\rho = (\boldsymbol{\rho_1}, \boldsymbol{\rho_2}, r_2)$.

Open: With $pk_{\widehat{V}}$, $pk_{\widetilde{V}}$, pk_S and two valid designated verifier signatures where the first signature $\hat{\rho} = (\hat{\sigma_1}, \hat{\sigma_2}, \hat{r_2})$ is designated to a verifier $\widehat{V}$ and the other signature $\tilde{\rho} = (\tilde{\sigma_1}, \tilde{\sigma_2}, \tilde{r_2})$ is designated to another verifier $\widetilde{V}$ as input, *Open* processes the necessary parameters as follows.

$$\begin{aligned}
M' &= M||\hat{\sigma_1}||pk_S = M||\tilde{\sigma_1}||pk_S, \\
T_{\widehat{V}} &= h(\hat{g}_{\hat{l}}^{\hat{r_2}} \hat{Y}^{\hat{h}(M')}), \\
h_{\widehat{V}} &= h(pk_S||pk_{\widehat{V}}||M||T_{\widehat{V}}), \\
T_{\widetilde{V}} &= h(\tilde{g}_{\tilde{l}}^{\tilde{r_2}} \tilde{Y}^{\tilde{h}(M')}), \\
h_{\widetilde{V}} &= h(pk_S||pk_{\widetilde{V}}||M||T_{\widetilde{V}}).
\end{aligned}$$

Let $(\hat{g}_{\hat{l}}, \hat{l}, \hat{Y}, \hat{h})$ and $(\tilde{g}_{\tilde{l}}, \tilde{l}, \tilde{Y}, \tilde{h})$ be the public parameter of $\widehat{V}$ and $\widetilde{V}$, respectively. If $\hat{\rho}$ and $\tilde{\rho}$ are generated from the same signature, then $\hat{\sigma_1} = \tilde{\sigma_1}$ always holds.

Hence, for two (or more) simulated designated verifier signatures or a pair of both simulated designated verifier signature and valid designated verifier signature, the probability that a designated verifier signature shares the first part of signature with other designated verifier signatures is negligible. From the Lagrange interpolating polynomial where the degree of the polynomial $P(h_V)$ is 2, $P(h_V) = \Sigma_{j=1}^{2} P_j(h_V)$, $P_j(h_V) = \Pi_{k=1,k\neq j}^{2} y_j(h_V - h_{V_k})/(h_{V_j} - h_{V_k})$ and $y = f(h_V) = x + r_1 \cdot h_V$, we can calculate the Lagrange coefficient of $P_{\widehat{V}}(0)$

and $P_{\widetilde{V}}(0)$ from $h_{\widehat{V}}$ and $h_{\widetilde{V}}$ as follows.

$$\begin{aligned} c_{\widehat{V}} &= \frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}, \\ c_{\widetilde{V}} &= \frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}. \end{aligned}$$

Therefore, *Open* processes a signature from two valid UDVS signatures as follows.

$$\begin{aligned} \sigma_1 &= \hat{\boldsymbol{\rho}}_{\mathbf{1}} = \tilde{\boldsymbol{\rho}}_{\mathbf{1}} = g^{r_1}, \\ \sigma_2 &= (\hat{\boldsymbol{\rho}}_{\mathbf{2}}{}^{c_{\widehat{V}}} \cdot \tilde{\boldsymbol{\rho}}_{\mathbf{2}}{}^{c_{\widetilde{V}}})^{\frac{1}{T_{\widehat{V}} \cdot c_{\widehat{V}} + T_{\widetilde{V}} \cdot c_{\widetilde{V}} + 1}} = H(M')^x, \\ \sigma_3 &= \hat{\boldsymbol{\rho}}_{\mathbf{2}} \cdot \sigma_2^{T_{\widehat{V}}} = \tilde{\boldsymbol{\rho}}_{\mathbf{2}} \cdot \sigma_2^{T_{\widetilde{V}}} = H(M')^{r_1}. \end{aligned}$$

Finally, *Open* outputs a signature on M as $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

4.4 Security Analysis of OT-UDVS

4.4.1 Completeness

Completeness of a Signature: The first part of the signature verification is as follows.

$$\begin{aligned} \hat{e}(\sigma_2, g) &\stackrel{?}{=} \hat{e}(H(M'), X) \\ \hat{e}(H(M')^x, g) &\stackrel{?}{=} \hat{e}(H(M'), X) \\ \hat{e}(H(M'), g^x) &\stackrel{?}{=} \hat{e}(H(M'), g^x) \end{aligned}$$

The last part of the signature verification is as follows.

$$\begin{aligned} \hat{e}(\sigma_3, g) &\stackrel{?}{=} \hat{e}(H(M'), \sigma_1) \\ \hat{e}(H(M')^{r_1}, g) &\stackrel{?}{=} \hat{e}(H(M'), g^{r_1}) \\ \hat{e}(H(M'), g^{r_1}) &\stackrel{?}{=} \hat{e}(H(M'), g^{r_1}) \end{aligned}$$

Completeness of a Designated Verifier Signature:

$$\begin{aligned}
\hat{e}(\boldsymbol{\rho_2}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), R) \\
\hat{e}(\boldsymbol{\rho_2}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot \boldsymbol{\rho_1}^{h_V}) \\
\hat{e}(\sigma_2^{T_V} \cdot \sigma_2 \cdot \sigma_3^{h_V}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot \sigma_1^{h_V}) \\
\hat{e}(H(M')^{x \cdot T_V} \cdot H(M')^{x} \cdot H(M')^{r_1 \cdot h_V}, g) &\stackrel{?}{=} \hat{e}(H(M'), g^{x \cdot T_V})\hat{e}(H(M'), g^{x} \cdot g^{r_1 \cdot h_V}) \\
\hat{e}(H(M')^{x \cdot T_V} \cdot H(M')^{x + r_1 \cdot h_V}, g) &\stackrel{?}{=} \hat{e}(H(M'), g^{x \cdot T_V})\hat{e}(H(M'), g^{x + r_1 \cdot h_V}) \\
\hat{e}(H(M')^{x \cdot T_V + x + r_1 \cdot h_V}, g) &\stackrel{?}{=} \hat{e}(H(M'), g^{x \cdot T_V + x + r_1 \cdot h_V})
\end{aligned}$$

Completeness of a Simulated Signature: Given a public parameter of a signer pk_S, a public parameter of designated verifier pk_V, a private key of designated verifier sk_V, a message M and a designate verifier signature ρ as input, first compute $M' = M||\boldsymbol{\rho_1}||pk_S, T_V = h(g_l^{r_2} Y^{\bar{h}(M')} \bmod l)$, $h_V = h(pk_S||pk_V||M||T_V)$, and $R = X \cdot \boldsymbol{\rho_1}^{h_V}$. Then check

$$\begin{aligned}
\hat{e}(\boldsymbol{\rho_2}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), R) \\
\hat{e}(H(M')^{k}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot \boldsymbol{\rho_1}^{h_V}) \\
\hat{e}(H(M')^{k}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot ((g^{k} \cdot X^{-T_V} \cdot X^{-1})^{\frac{1}{h_V}})^{h_V}) \\
\hat{e}(H(M')^{k}, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), g^{k} \cdot X^{-T_V}) \\
\hat{e}(H(M')^{k}, g) &\stackrel{?}{=} \hat{e}(H(M'), g^{k}).
\end{aligned}$$

Therefore, the above statements show that the simulated signature indeed holds.

Completeness of an Opened Signature:

$$
\begin{aligned}
\hat{e}(H(M'),X) &\stackrel{?}{=} \hat{e}(\sigma_2, g)\\
\hat{e}(H(M'),X) &\stackrel{?}{=} \hat{e}((\hat{\sigma_2}^{c_{\widehat{V}}} \cdot \tilde{\sigma_2}^{c_{\widetilde{V}}})^{\frac{1}{T_{\widehat{V}} \cdot c_{\widehat{V}} + T_{\widetilde{V}} \cdot c_{\widetilde{V}} + 1}}, g)\\
\hat{e}(H(M'),X) &\stackrel{?}{=} \hat{e}((H(M')^{T_{\widehat{V}} \cdot x} \cdot H(M')^{x} \cdot H(M')^{r_1 \cdot h_{\widehat{V}}})^{\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}}\\
&\quad \cdot (H(M')^{T_{\widetilde{V}} \cdot x} \cdot H(M')^{x} \cdot H(M')^{r_1 \cdot h_{\widetilde{V}}})^{\frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}}, g)^{\frac{1}{T_{\widehat{V}} \cdot c_{\widehat{V}} + T_{\widetilde{V}} \cdot c_{\widetilde{V}} + 1}}\\
\hat{e}(H(M'),X) &\stackrel{?}{=} \hat{e}((H(M')^{(T_{\widehat{V}} \cdot (\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}) + T_{\widetilde{V}} \cdot (\frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}})) \cdot x}) \cdot\\
&\quad (H(M')^{(\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}} + \frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}) \cdot x}) \cdot\\
&\quad (H(M')^{r_1 \cdot (h_{\widehat{V}} \cdot \frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}} + h_{\widetilde{V}} \cdot \frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}})}), g)^{\frac{1}{T_{\widehat{V}} \cdot (\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}) + T_{\widetilde{V}} \cdot (\frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}) + 1}}\\
\hat{e}(H(M'),X) &\stackrel{?}{=} \hat{e}(H(M')^{x \cdot (T_{\widehat{V}} \cdot (\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}) + T_{\widetilde{V}} \cdot (\frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}))}\\
&\quad \cdot H(M')^{x}, g)^{\frac{1}{T_{\widehat{V}} \cdot (\frac{-h_{\widetilde{V}}}{h_{\widehat{V}} - h_{\widetilde{V}}}) + T_{\widetilde{V}} \cdot (\frac{-h_{\widehat{V}}}{h_{\widetilde{V}} - h_{\widehat{V}}}) + 1}}\\
\hat{e}(H(M'),g^x) &\stackrel{?}{=} \hat{e}(H(M')^{x}, g).
\end{aligned}
$$

The above statements show that an opened signature is complete. □

4.4.2 Unforgeability

Theorem 4.1 *In the random oracle model, our one-time universal designated verifier scheme is existential unforgeability under an adaptive chosen message and chosen public key attack if the CDH assumption holds.*

Proof: The existential unforgeability under an adaptive chosen message and chosen public key attack of our OT-UDVS scheme will be proved by assuming if there exists a forger $\mathcal{A}$, which runs the game defined in Section 4.2.3, then there will exist an adversary $\mathcal{F}$ solving the CDH problem by using $\mathcal{A}$. First, the oracles defined in Section 4.2.3 will be constructed. Next, $\mathcal{F}$ is constructed and run over $\mathcal{A}$ with the existential unforgeability game defined in Section 4.2.3. The success probability of the existential unforgeability game under an adaptive chosen message and chosen public key attack is then concluded. Finally, it is shown that, from the existential unforgeability game and its success probability, the success probability of solving the CDH problem is non-negligible if the success probability of the above game is non-negligible.

The oracles are constructed and the existential unforgeability game is run as follows. On input an instance of the CDH problem g, g^a and g^b, $\mathcal{F}$ sets g^b as one

of the answers for hash query to random oracle $\mathcal{HO}$. $\mathcal{F}$ then sets $X = g^a$ in one of the signers' public parameters defined as pk_S^*. It is assumed that there exists an algorithm tracking a member on the list of each oracle. Hence, such algorithms will be omitted. Then $M' = M||\sigma_1||pk_S$ and $\breve{M} = M^*||\sigma_1||pk_S$ are parsed. From the above setting, it is easy for $\mathcal{F}$ to construct the $\mathcal{HO}$, $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{VSO}$, $\mathcal{VPO}$, $\mathcal{DSO}$, $\mathcal{DVO}$, $\mathcal{SDO}$, $\mathcal{OPO}$ and $\mathcal{SKO}$ oracles as follows.

$\mathcal{HO}$: Select $d \xleftarrow{\$} \{0,1\}$ such that the probability of $d = 1$ is $\frac{1}{q_H}$. If $d = 1$ then set $H(\breve{M}) = g^b$ return $H(\breve{M})$. Otherwise, $k \xleftarrow{\$} \mathbb{Z}_p; H(M') = g^k$ and return $H(M')$.

$\mathcal{SPO}$ oracle: Let $\mathsf{param} = (p, \hat{e}, h, O_H, g)$ and choose $\breve{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\breve{d} = 1$ is $\frac{1}{q_{SP}}$. If $\breve{d} = 1$ then set $X = g^a$ and return $pk_S^* = (\mathsf{param}, X^*)$. Otherwise, $t \xleftarrow{\$} \mathbb{Z}_p; X = g^t$ and then return $pk_S = (\mathsf{param}, X)$ and keep t as a private key.

$\mathcal{SKO}$ oracle: the $\mathcal{SKO}$ oracle responses every query on input pk_S and pk_V with its corresponding private key expected for pk_S^*, which the $\mathcal{SKO}$ oracle outputs $\perp$.

$\mathcal{SSO}$ oracle: Let $r_1 \xleftarrow{\$} \mathbb{Z}_p$. On input pk_S, M, the $\mathcal{SSO}$ oracle outputs $\sigma = (\sigma_1 = g^{r_1}, \sigma_2 = H(M')^t = g^{t \cdot k}, \sigma_3 = H(M')^{r_1} = g^{k \cdot r_1})$ for every query excepted when $pk_S = pk_S^*$ and $M = M^*$. In a case of (pk_S^*, M^*), the $\mathcal{SSO}$ oracle outputs $\perp$.

$\mathcal{DSO}$ oracle: Let $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. Compute M', T_V and h_V as described in Section 4.3. On input pk_S, pk_V, M, the $\mathcal{DSO}$ oracle outputs $\rho = (\boldsymbol{\rho_1} = g^{r_1}, \boldsymbol{\rho_2} = (H(M')^t)^{T_V} \cdot (H(M')^t \cdot (H(M')^{r_1})^{h_V} = g^{t \cdot k \cdot (T_V + 1) + k \cdot r_1 \cdot h_V}, r_2)$ for every query excepted when $pk_S = pk_S^*$ and $M = M^*$. In a case of (pk_S^*, M^*), the $\mathcal{DSO}$ oracle outputs $\perp$.

$\mathcal{VSO}$, $\mathcal{VPO}$, $\mathcal{DVO}$, $\mathcal{SDO}$, and $\mathcal{OPO}$ oracles: These oracles are straightforward as described in Section 4.3.

$\mathcal{F}$ then gives an access of these oracles to $\mathcal{A}$. It is assumed that $\mathcal{A}$ always makes a request for a hash of message M to the random oracle $\mathcal{HO}$ before it makes a query to the $\mathcal{SSO}$, $\mathcal{VSO}$, $\mathcal{DSO}$, $\mathcal{DVO}$, $\mathcal{SDO}$ or $\mathcal{OPO}$ oracles, or before it outputs a potential forgery, denoted by $(M^*, \rho_*, pk_S^*, pk_V^*)$. We also denote by q a polynomial

upper bound on the number of queries that $\mathcal{A}$ makes to the random oracle $\boldsymbol{\mathcal{HO}}$ and the $\boldsymbol{\mathcal{SKO}}$ oracle.

Finally, $\mathcal{A}$ outputs a forged designated verifier signature ρ_* on a message M^* with respect to pk_S^*, pk_V^*. It is noted that a message M^* may be submitted to the random oracle $\boldsymbol{\mathcal{HO}}$ before F outputs the forgery, however, a message M^* must never be submitted to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{DSO}}$ and $\boldsymbol{\mathcal{SDO}}$ oracles.

Let e denote the base of the natural logarithm. Now the probability of the following events that $\mathcal{F}$ does not abort during the simulation is considered.

- E_1: *$\mathcal{F}$ does not abort during the queries to the $\boldsymbol{\mathcal{SKO}}$ oracle.*

  The probability of this event is greater than $(1 - \frac{1}{q_{SP}})^{q_{SP}-1} \approx \frac{q_{SP}}{e\cdot(q_{SP}-1)}$.

- E_2: *$\mathcal{F}$ does not abort during the queries to the $\boldsymbol{\mathcal{SSO}}$ oracle.*

  The fact is that $\mathcal{A}$ needs at least one hash value and one signer secret to output a forgery, and hence, $q_{SS} \leq (q_H - 1) \cdot (q_{SP} - 1)$. Therefore, the probability of this event is greater than $(1 - \frac{1}{q_H \cdot q_{SP}})^{q_{SS}} = (1 - \frac{1}{q_H \cdot q_{SP}})^{(q_H-1)\cdot(q_{SP}-1)} \approx \frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^(q_H + q_{SP} - 1)$.

- E_3: *$\mathcal{F}$ does not abort during the queries to the $\boldsymbol{\mathcal{DSO}}$ oracle.*

  Similar to the E_2 event, which $q_{DS} \leq (q_H - 1) \cdot (q_{SP} - 1)$, the probability of this event is greater than $(1 - \frac{1}{q_H \cdot q_{SP}})^{q_{DS}} = (1 - \frac{1}{q_H \cdot q_{SP}})^{(q_H-1)\cdot(q_{SP}-1)} \approx \frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^(q_H + q_{SP} - 1)$.

Let $Succ^{CM-CPK-A}_{EUF-OT-UDVS} = \epsilon$ be the probability that $\mathcal{A}$ wins the game. The probability that $\mathcal{A}$ wins the above game and outputs with a message M^* and a signer's public parameter pk_S^* is $\frac{\epsilon}{q_H \cdot q_{SP} - max(q_D, q_{SS})} \leq \frac{\epsilon}{q_H + q_{SP} - 1}$ where $q_{SS} \leq (q_H - 1) \cdot (q_{SP} - 1)$ and $q_{DS} \leq (q_H - 1) \cdot (q_{SP} - 1)$. Note that q_H and q_{SP} are the maximum numbers of queries that $\mathcal{A}$ made to the random oracle $\boldsymbol{\mathcal{HO}}$ and the $\boldsymbol{\mathcal{SPO}}$ oracle, respectively. Putting the above probabilities together, the probability is resolved such that $\mathcal{F}$ does not abort during the simulation and $\mathcal{A}$ wins the game with M^*, pk_S^* is about $\frac{\epsilon}{q_H+q_{SP}-1} \cdot \frac{q_{SP}}{e\cdot(q_{SP}-1)} \cdot (\frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{q_H+q_{SP}-1})^2 = \frac{\epsilon}{q_H+q_{SP}-1} \cdot \frac{q_{SP}}{e^3\cdot(q_{SP}-1)} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{2(q_H+q_{SP}-1)} > \frac{\epsilon}{e^3(q_H+q_{SP}-1)}$.

From the above probability, it is obvious that the probability, which $\mathcal{F}$ successfully runs the above simulation and $\mathcal{A}$ wins the game with M^* and pk_S^*, is

non-negligible compared with the probability that $\mathcal{A}$ wins the game where $q > (q_H + q_{SP} - 1)$.

Hence, within non-negligible running time due to the forking lemma [PS00], $\mathcal{F}$ also obtains another set of forgery by rerunning the experiment with $\mathcal{A}$ working as follows.

- First, reset $\mathcal{A}$ to the initial state.
- Secondly, provide the same setting as same as the previous experiment except a new set of verifiers' public parameters is given.
- Finally, rerun the experiment with the same random tape as the first experiment.

At the end of the second experiment, $\mathcal{F}$, with non-negligible probability, outputs a forgery $(M^*, \rho_{**}, pk_S^*, pk_V^{**})$. Since the setting is same as the first experiment, $\mathcal{A}$ outputs M^*, pk_S^* with the same probability as in the first experiment. However, in the second experiment, $\mathcal{A}$ given with a new set of verifiers' public parameters; hence, $\mathcal{A}$ outputs ρ_{**}, pk_V^{**}, which are different from those in the first experiment.

Finally, $\mathcal{F}$ runs *Open* to obtain σ, which is $\sigma \leftarrow Open(\rho_*, \rho_{**}, pk_V^*, pk_V^{**}, pk_S^*, M^*)$. Since $X = g^a$ in pk_S^* and $H(\breve{M}) = g^b$, $\mathcal{F}$ then outputs $\sigma = H(\breve{M})^a = g^{b \cdot a}$ as an answer to the CDH problem with non-negligible probability as mentioned above.

The above simulation shows that a probability of success on attacking our OT-UDVS scheme by existential unforgeability under an adaptive chosen message and chosen public key attack is negligible since a probability of solving CDH problem is negligible. □

4.4.3 Non-transferability Privacy

Theorem 4.2 *In the random oracle model, the purposed one-time universal designated verifier scheme is existential non-transferable privacy against adaptively chosen message and chosen public key distinguisher* $\mathcal{A}_{ENT-OT-UDVS}^{CM-CPK-A}$.

Proof: To prove Theorem 4.2, it will be shown that the success probability of distinguisher $\mathcal{A}$ attacking our OT-UDVS scheme by running the existential non-transferable privacy game defined in Section 4.2.4 is non-negligible. The oracle and the existential non-transferable privacy game defined in Section 4.2.3 will first be constructed. Then it will be shown that both probabilities of the distribution of DVS

signature generated by a signature holder and a designated verifier are indeed equal. Finally, it will be concluded the indistinguishability of valid or simulated designated verifier signatures of the OT-UDVS scheme under the existential non-transferable privacy game.

In the following, it will be shown that there exists a simulator that runs on both signer and verifier and generates a designated verifier signature, which is indistinguishable whether a signer or a verifier indeed generated it.

First, the simulation will be constructed as follows. $\mathcal{F}$ constructs the oracles in the same way as the proof in Theorem 4.1 except that the random oracle is no longer required and $\mathcal{F}$ can arbitrarily generated a public-private key pair for $\mathcal{A}$. $\mathcal{F}$ gives an access to those oracles to $\mathcal{A}$. Note that for every private key corresponding to its queried public parameters, $\mathcal{F}$ keeps them secretly to itself.

Secondly, the distribution of the $\boldsymbol{\mathcal{DSO}}$ oracle is analysed. There are two uniformly random integers $r_1, r_2 \in \mathbb{Z}_p$ involved in the production of designated verifier signature beside the private key of the signer. With random integers $r_1, r_2 \in \mathbb{Z}_p$ and a private key of the signer sk_S, $\mathcal{F}$ randomly produces the designated verifier signature as follows. Let ρ_{DV} denote a designated verifier signature in the distribution of the $\boldsymbol{\mathcal{DSO}}$ oracle.

$$\begin{aligned}
\boldsymbol{\rho_1} &= g^{r_1}, \\
M' &= M||\boldsymbol{\rho_1}||pk_S, \\
T_V &= h(g_l^{r_2} Y^{\bar{h}(M')}), \\
h_V &= h(pk_S||pk_V||M||T_V), \\
R' &= H(M')^x \cdot H(M')^{r_1 \cdot h_V}, \\
\boldsymbol{\rho_2} &= H(M')^{x \cdot T_V} \cdot R', \\
\rho_{DV} &= (\boldsymbol{\rho_1}, \boldsymbol{\rho_2}, r_2).
\end{aligned}$$

Hence, if a designated verifier signature ρ_* is randomly chosen, then the probability that ρ_* is in the distribution of the $\boldsymbol{\mathcal{DSO}}$ oracle is $\mathbf{Pr}[\rho_{DV} = \rho_*] = \frac{1}{p^2}$.

Finally, the distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle is analysed. There are also two uniformly random integers $k, \bar{r}_2 \in \mathbb{Z}_p$ involved in the production of the designated verifier signature in addition to the private key of the verifier. With these random integers $k, \bar{r}_2$ and a private key of a verifier sk_V, $\mathcal{F}$ randomly produces the designated verifier signature as follows, with ρ_{DS} denoted a designated verifier signature in the

distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle.

$$\begin{aligned}
\bar{M} &= M||K||pk_S, \\
T_V &= h(g_l^{\bar{r_2}} Y^{\bar{h}(\bar{M})}), \\
h_V &= h(pk_S||pk_V||M||T_V), \\
\boldsymbol{\rho_1} &= (g^k \cdot X^{-T_V} \cdot X^{-1})^{\frac{1}{h_V}}, \\
M' &= M||\sigma_{(1,SH)}||pk_S, \\
r_2 &= \bar{r_2} + y \cdot \bar{h}(\bar{M}) - y \cdot \bar{h}(M'), \\
\boldsymbol{\rho_2} &= H(M')^k, \\
\rho_{DS} &= (\boldsymbol{\rho_1}, \boldsymbol{\rho_2}, r_2).
\end{aligned}$$

Therefore, if a designated verifier signature ρ_* is randomly chosen, then the probability that ρ_* is in the distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle is $\mathbf{Pr}[\rho_{DS} = \rho_*] = \frac{1}{p^2}$.

To conclude, due to the above probabilities, one cannot distinguish whether a randomly given valid universal designated verifier signature is produced by the $\boldsymbol{\mathcal{DSO}}$ oracle or the $\boldsymbol{\mathcal{SDO}}$ oracle. Thus, our OT-UDVS scheme satisfies the non-transferable privacy property. □

4.4.4 Single Designatability

Theorem 4.3 *Our one-time universal designated verifier scheme is existential single designatable under an adaptive chosen message and chosen public key attack if the hash function is collision resistant.*

Proof: Assume that the hash function h of our OT-UDVS scheme is a collision resistant hash function. We denote by $\mathcal{A}$ a forger and let $\mathcal{F}$ denote an adversary searching for a collision message-pair for hash function h through $\mathcal{A}$. Due to the completeness of an opened signature, the only designated verifier signatures pair $(\hat{\rho}, \tilde{\rho}\ )$ that can open is when $\hat{\sigma_1} = \tilde{\sigma_2}$, $\hat{\sigma_2} = \tilde{\sigma_2}$ and $\hat{r_2} \neq \hat{r_2}$ or $pk_{\widehat{V}} \neq pk_{\widetilde{V}}$. The collision of hash function h will occur if such an event happens.

From the above statement, the simulation can be constructed as follows. First, since $\mathcal{F}$ can arbitrarily generate a public-private key pair for $\mathcal{A}$, $\mathcal{F}$ constructs straightforwardly oracles as described in Section 4.2.5. $\mathcal{A}$ is given access to those oracles. At the end of the above queries, it is assumed that $\mathcal{A}$ outputs two designated verifier signatures $\hat{\rho}, \tilde{\rho}$ on a message M^* regarding to public parameters $pk_S^*, pk_{\widehat{V}}, pk_{\widetilde{V}}$. $\mathcal{F}$ pronounces that $\mathcal{A}$ wins the game if both signatures are accepted

by the $\boldsymbol{\mathcal{DVO}}$ oracle, and a private key corresponding to pk_S^* has never been revealed by the $\boldsymbol{\mathcal{SKO}}$ oracle and only one designated verifier signature on message M^* can be queried. $\mathcal{F}$ then computes $T_{\hat{V}} = h(\hat{g}_{\hat{l}}^{\hat{r}_2}\hat{Y}^{\hat{h}(M^*||\hat{\sigma}_1||pk_S^*)})$; $T_{\tilde{V}} = h(\tilde{g}_{\tilde{l}}^{\tilde{r}_2}\tilde{Y}^{\tilde{h}(M^*||\tilde{\sigma}_1||pk_S^*)})$. Next, $\mathcal{F}$ sets $\hat{M} = pk_S^*||pk_{\hat{V}}||M^*||T_{\hat{V}}$; $\tilde{M} = pk_S^*||pk_{\tilde{V}}||M^*||T_{\tilde{V}}$ and computes $h_{\hat{V}} = h_{\tilde{V}} = h(\hat{M}) = h(\tilde{M})$.

It is noted that both $T_{\hat{V}}$ and $h_{\hat{V}}$ are required to be equal to $T_{\tilde{V}}$ and $h_{\tilde{V}}$, respectively. This is because of $\hat{\sigma}_1 = \tilde{\sigma}_1$ and $\hat{\sigma}_2 = \tilde{\sigma}_2$ as we mentioned earlier. However, $T_{\hat{V}} = T_{\tilde{V}}$ is easy to achieve since $\mathcal{A}$ possessed the verifier secret keys and can arbitrarily compute $h(\hat{g}_{\hat{l}}^{\hat{r}_2}\hat{Y}^{\hat{h}(M^*||\hat{\sigma}_1||pk_S^*)}) = h(\tilde{g}_{\tilde{l}}^{\tilde{r}_2}\tilde{Y}^{\tilde{h}(M^*||\tilde{\sigma}_1||pk_S^*)})$. On the other hand, $h_{\hat{V}}$ is hard to make itself equivalent to $h_{\tilde{V}}$ since $\hat{M} \neq \tilde{M}$ in every case.

Hence, $\mathcal{F}$ outputs $\hat{M}$ and $\tilde{M}$ as messages that lead to a collision of hash value $h(\hat{M}) = h(\tilde{M})$. This completes the proof. □

4.5 Definition of Universal Designated Verifier Signature with Threshold-Signers Schemes (TS-UDVS)

4.5.1 Outline of TS-UDVS

Assume that every party does the registration with a certificate of authority CA to obtain certificates on their public parameters prior to communications with other parties. We denote $\mathfrak{LS}$ with a list of the entire signers such that $\mathfrak{LS} = \{pk_{S_i}\}$ where i is an index of the signer.

A threshold-signers universal designated verifier signature scheme Σ is a seven-tuple ($SKeyGen$, $Sign$, $Verify$, $VKeyGen$, $TDesignate$, $DVerify$, $DSimulate$). The definition of $SKeyGen$, $Sign$, $Verify$ and $VKeyGen$ is same as the definition in Section 3.3 where param is 1^ℓ. The definition of $TDesignate$, $DVerify$ and $DSimulate$ is described as follows.

Signature Threshold-Signers Designation ($TDesignate$): Denote by t a number of signers who a signature holder possessed their signatures and denote by n the total number of signers. This is a probabilistic algorithm that, given a verifier's public parameter pk_V, signers' public parameters $pk_{S_1}, ..., pk_{S_n}$, the signers' signatures $\sigma_1, ..., \sigma_t$, and a message M as input, $TDesignate$ returns

a designated verifier signature ρ. That is,

$$TDesignate(M, \sigma_1, ..., \sigma_t, pk_V, pk_{S_1}, ..., pk_{S_n}) \rightarrow \rho.$$

Designated Verifier Signature Verification ($DVerify$): This is a deterministic algorithm that, given a verifier's public parameter pk_V, signers' public parameters $pk_{S_1}, ..., pk_{S_n}$, a message M and a designated verifier signature ρ as input, $DVerify$ returns a verification decision $d \in \{Accept, Reject\}$. That is,

$$DVerify(M, \rho, pk_V, pk_{S_1}, ..., pk_{S_n}) \rightarrow \mathsf{d}.$$

Simulation of a Designated Verifier Signature ($DSimulate$): This is a probabilistic algorithm that, given a verifier's public parameter pk_V, a verifier's private key sk_V, signers' public parameters $pk_{S_1}, ..., pk_{S_n}$, and a message M as input, $DSimulate$ outputs a designated verifier signature ϱ such that

$$DVerify(M, \varrho, pk_V, sk_V, pk_{S_1}, ..., pk_{S_n}) \rightarrow Accept.$$

That is,

$$DSimulate(M, pk_V, sk_V, pk_{S_1}, ..., pk_{S_n}) \rightarrow \varrho.$$

Security notions for universal designated verifier signature with threshold-signers (TS-UDVS) schemes are described in the following subsections. They include completeness, unforgeability, non-transferable privacy and anonymity.

4.5.2 Completeness

For all $\ell \in \mathbb{N}$, all $(pk_S, sk_S) \in SKeyGen(1^\ell)$, all $(pk_V, sk_V) \in VKeyGen(1^\ell)$ and all messages M, a universal designated verifier signature with threshold-signers scheme must comply with the following properties:

Completeness of a Signature:

$$\forall \sigma \in Sign(M, sk_S, pk_S), \mathbf{Pr}[Verify(M, \sigma, pk_S) = Accept] = 1. \tag{4.6}$$

Completeness of a TS-UDVS:

$$\forall \rho \in TDesignate(M, \sigma_1, ..., \sigma_t, pk_V, pk_{S_1}, ..., pk_{S_n}),$$

$$\mathbf{Pr}[DVerify(M, \rho, pk_V, pk_{S_1}, ..., pk_{S_n}) = Accept] = 1. \tag{4.7}$$

Completeness of a Simulated TS-UDVS:

$$\forall \varrho \in DSimulate(M, pk_V, sk_V, pk_{S_1}, ..., pk_{S_n}),$$
$$\mathbf{Pr}[DVerify(M, \varrho, pk_V, pk_{S_1}, ..., pk_{S_n}) = Accept] = 1. \quad (4.8)$$

4.5.3 Unforgeability

In this chapter, when discussing the unforgeability property, we are referring to the "designated verifier unforgeability" in [SBWP03, HSMW06]. The unforgeability property in [HSMW06] provides security against existential unforgeability under an adaptive chosen message and chosen public key attack. It intentionally prevents an attacker corrupted with a signature holder from generating a designated verifier signature ρ_* on a new message M^*. Formally, this unforgeability provides an assurance that one with access to a signing oracle, designation oracle, simulated signature oracle, and verification oracles, and with the signer's public parameter pk_S, should be unable to produce a designated verifier signature on a new message even when arbitrarily choosing the verifier's public parameter pk_V and message M as input.

However, for TS-UDVS schemes, unforgeability has a slightly different notion from that in [SBWP03, HSMW06]. To provide security of unforgeability against insider corruption (up to $t-1$ signers) for TS-UDVS schemes, our unforgeability notion has adapted the notion of unforgeability in the ring signature schemes in [RST01, BKM06, LW04, SW07, TWC+04, LWW03, BSS02]. Intuitively, the unforgeability property of a TS-UDVS scheme provides security against existential unforgeability under an adaptive chosen message, chosen public key attack and insider corruption. It intentionally prevents an attacker corrupted with $(t-1)$ signers and a signature holder from generating a threshold-signers designated verifier signature ρ_* on a new M^*.

Here, our unforgeability provides assurance that, with access to a signing oracle, threshold-signers designation oracle, and simulated designated verifier signature oracle, and with signers' public parameters $pk_{S_1}, ...pk_{S_n}$, arbitrarily chosen verifier's public parameter pk_V^* and the knowledge of t'-signer secret keys $sk_{S_1^*}, ..., sk_{S_{t'}^*}$, one should not able to produce a designated verifier signature on a new arbitrarily chosen message M^*. Note that t is the threshold, t' is the number of colluded signers and $t' < t$. The unforgeability of TS-UDVS is formally modelled as follows.

First, the oracles are provided in order to model the ability of adversaries breaking the unforgeability of TS-UDVS schemes as follows. The definition of the $\mathcal{SPO}$,

$\mathcal{VPO}$ and $\mathcal{SKO}$ oracles can be found in Section 4.2.3. The definition of the $\mathcal{SSO}$ oracle can be found in Section 3.3.1. For the definition of the $\mathcal{TDO}$ and $\mathcal{SDO}$ oracles, there are described as follows.

$\mathcal{TDO}$ oracle: Let $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$ and $\mathfrak{TS} = \{pk_{S_j}\}$, where j is an index of each signer in a threshold t and $\mathfrak{TS} \subset \mathfrak{LS}$. At most q_{TD} times, $\mathcal{A}$ can make a query for a designated verifier signature ρ on its choice of message M under its choice of a group of signers' public parameters $\mathfrak{LS}$, a group of threshold signers' public parameters $\mathfrak{TS}$ and a verifier's public parameter pk_V. In response, $\mathcal{TDO}$ runs the $TDesignate$ algorithm to generate a designated verifier signature ρ on a message M corresponding with $\mathfrak{LS}, \mathfrak{TS}, pk_V$. $\mathcal{TDO}$ then returns ρ, M to $\mathcal{A}$.

$\mathcal{SDO}$ oracle: At most q_{SD} times, under its choice of signers' public parameters $pk_{S_1}, ..., pk_{S_n}$ and a verifier's public parameter pk_V, $\mathcal{A}$ can make a query for a (simulated) designated verifier signature ϱ on its choice of message M, where ϱ must indeed be generated by the verifier. In response, $\mathcal{SDO}$ runs the $DSimulate$ algorithm to generate a (simulated) designated verifier signature ϱ on a message M corresponding with $pk_{S_1}, ..., pk_{S_n}, pk_V$. $\mathcal{SDO}$ then returns ϱ, M to $\mathcal{A}$.

Let $CM\text{-}CPK\text{-}A$ be the adaptively chosen message, chosen public key attack and insider corruption. Let $EUF\text{-}TS\text{-}UDVS$ be the existential unforgeability of the TS-UDVS scheme. Let $\mathcal{A}^{CM-CPK-A}_{EUF-TS-UDVS}$ be the adaptively chosen message and chosen public key adversary and allow $\mathcal{F}$ to be a simulator. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential unforgeability of the TS-UDVS scheme: given a choice of messages M and an access to the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{VPO}$, $\mathcal{TDO}$, $\mathcal{SDO}$ and $\mathcal{SKO}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles. At the end of these queries, it is assumed that $\mathcal{A}$ outputs a forged signature ρ_* on a new message M^* with respect to the public parameters $pk^*_{S_1}, ..., pk^*_{S_n}, pk^*_V$. $\mathcal{A}$ wins the game if:

1. $Accept \leftarrow DVerify(M^*, \rho_*, pk^*_V, pk^*_{S_1}, ..., pk^*_{S_n})$.
2. pk^*_V has never been submitted as the input of a query for a private key to the $\mathcal{SKO}$ oracle.
3. At least $n - t'$ of the challenge signers' public parameters have never been submitted as the input of a query for a private key to the $\mathcal{SKO}$ oracle.

4. For each signer's public parameter, $\mathcal{A}$ never makes a request for a signature on input $M^*, pk^*_{S_i}$ to the $\boldsymbol{\mathcal{SSO}}$ oracle, where i is an index of submitted signer's public parameter.

5. $\mathcal{A}$ never makes a request for a designated verifier signature on input M^*, $pk^*_{S_1}, ..., pk^*_{S_n}$ to the $\boldsymbol{\mathcal{TDO}}$ oracle.

6. $\mathcal{A}$ never makes a request for a simulated designated verifier signature on input M^*, pk^*_V to the $\boldsymbol{\mathcal{SDO}}$ oracle.

Let $Succ^{CM-CPK-A}_{EUF-TS-UDVS}(.)$ be a success probability that $\mathcal{A}^{CM-CPK-A}_{EUF-TS-UDVS}$ wins the above game.

Definition 4.4 *The TS-UDVS scheme is ($\mathtt{t}$,q_H,q_{SP},q_{SS},q_{VP},q_{TD},q_{SD}, q_{SK},ϵ)-secure existential unforgeable under an adaptive chosen message, chosen public key attack and insider corruption if there is no PPT CM-CPK-A adversary $\mathcal{A}^{CM-CPK-A}_{EUF-TS-UDVS}$ such that the success probability $Succ^{CM-CPK-A}_{EUF-TS-UDVS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-CPK-A}_{EUF-TS-UDVS}$ runs in time at most $\mathtt{t}$, makes at most q_H, q_{SP}, q_{SS}, q_{VP}, q_{TD}, q_{SD}, and q_{SK} queries to the random oracles, the $\boldsymbol{\mathcal{SPO}}$ oracle, the $\boldsymbol{\mathcal{SSO}}$ oracle, the $\boldsymbol{\mathcal{VPO}}$ oracle, the $\boldsymbol{\mathcal{TDO}}$ oracle, the $\boldsymbol{\mathcal{SDO}}$ oracle, and the $\boldsymbol{\mathcal{SKO}}$ oracle, respectively.*

4.5.4 Non-transferable Privacy

Building on the non-transferable privacy property in [SBWP03, HSMW08, HSMW06], the non-transferable privacy property for TS-UDVS schemes requires that even one obtains many threshold-signers designated verifier signatures $\rho_1, ..., \rho_q$ on its choice of messages $M \in \{M_1, ..., M_q\}$ designated to the same or different verifiers, where $\rho_1, ..., \rho_q$ are generated by the same signature holder using the same set of signatures $\sigma_1, ..., \sigma_t$, it is hard to convince other party that a signer indeed generated a signature $\rho \in \{\rho_1, ..., \rho_q\}$ on a message $M \in \{M_1, ..., M_q\}$. This intentionally prevents a distinguisher from distinguishing a signer from a (simulated) threshold-signers designated verifier signature ρ_* on any new message M^*. The non-transferable privacy of TS-UDVS is defined as follows.

First, the oracles provided in order to model the ability of adversaries breaking the non-transferable privacy of TS-UDVS schemes are described in Section 4.5.3. Let $ENT\text{-}TS\text{-}UDVS$ denote the existential non-transferable privacy of TS-UDVS scheme. Let $\mathcal{A}^{CM-CPK-A}_{ENT-TS-UDVS}$ be the adaptively chosen message and chosen public

key distinguisher and let $\mathcal{F}$ be a simulator. The following experiment between $\mathcal{F}$ and $\mathcal{A}$ is prescribed to demonstrate the existential non-transferable privacy of the TS-UDVS scheme. The experiment is divided into two phases, as described as follows.

1. **Phase** 1**:** With any adaptive strategies, $\mathcal{A}$ arbitrarily sends queries to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VPO}}$, $\boldsymbol{\mathcal{TDO}}$, $\boldsymbol{\mathcal{SDO}}$ and $\boldsymbol{\mathcal{SKO}}$ oracles. The oracles respond as their design.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs $M^*, pk^*_{S_1}, ..., pk^*_{S_n}, pk^*_V$ such that:

   a. Given $pk^*_{S_1}, ..., pk^*_{S_n}$ and M^* as input, $\mathcal{A}$ never issues a request for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. Given $pk^*_{S_1}, ..., pk^*_{S_n}$ and M^* as input, $\mathcal{A}$ never issues a request for a designated verifier signature to the $\boldsymbol{\mathcal{TDO}}$ oracle.

   c. Given pk^*_V and M^* as input, $\mathcal{A}$ never issues a request for a designated verifier signature to the $\boldsymbol{\mathcal{SDO}}$ oracle.

   d. Given $pk^*_{S_1}, ..., pk^*_{S_n}$ as input, $\mathcal{A}$ never issues a request for a private key to the $\boldsymbol{\mathcal{SKO}}$ oracle.

   After this, $\mathcal{F}$ chooses a random bit $b \overset{\$}{\leftarrow} \{0, 1\}$. If $b = 1$ then, on input $pk^*_{S_1}, ..., pk^*_{S_n}$, pk^*_V and M^*, $\mathcal{F}$ makes a request for a designated verifier signature to the $\boldsymbol{\mathcal{TDO}}$ oracle and responds to $\mathcal{A}$ with ρ as an output from the $\boldsymbol{\mathcal{TDO}}$ oracle. Otherwise, on input $pk^*_{S_1}, ..., pk^*_{S_n}$, pk^*_V and M^*, $\mathcal{F}$ makes a request for a simulated designated verifier signature to the $\boldsymbol{\mathcal{SDO}}$ oracle and responds to $\mathcal{A}$ with ρ as an output from the $\boldsymbol{\mathcal{SDO}}$ oracle.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it wants. One condition must be met that $\mathcal{A}$ must have at least one set of the challenge $M^*, pk^*_{S_1}, ..., pk^*_{S_n}, pk^*_V$ such that

   a. $\mathcal{A}$ never submits a request for a signature on input $M^*, pk^*_{S_1}, ..., pk^*_{S_n}$ to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. $\mathcal{A}$ never submits a request for a designated verifier signature on input $M^*, pk^*_{S_1}, ...,pk^*_{S_n}, pk^*_V$ to the $\boldsymbol{\mathcal{TDO}}$ oracle.

c. $\mathcal{A}$ never submits a request for a designated verifier signature on input M^*, pk_V^* to the $\boldsymbol{\mathcal{SDO}}$ oracle.

d. $\mathcal{A}$ never submits any request for a private key $sk_{S_i}^*$ corresponding with $pk_{S_i}^*$ to the $\boldsymbol{\mathcal{SKO}}$ oracle, where i is the index and $i \in \{1, ..., n\}$.

4. **Guessing:** On the challenge $M^*, pk_{S_1}^*, ..., pk_{S_n}^*, pk_V^*$, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ_{ENT-TS-UDVS}^{CM-CPK-A}(.)$ be the success probability that $\mathcal{A}_{ENT-TS-UDVS}^{CM-CPK-A}$ wins the above game.

Definition 4.5 *We say that the TS-UDVS scheme is ($\mathtt{t}$,q_H,q_{SP},q_{SS},q_{VP},q_{TD},q_{SD}, q_{SK},ϵ)-secure existential non-transferable private under a chosen message and chosen public key attack if there is no PPT CM-CPK-A distinguisher $\mathcal{A}_{ENT-TS-UDVS}^{CM-CPK-A}$ such that the success probability $Succ_{ENT-TS-UDVS}^{CM-CPK-A}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']|$ $= \epsilon$ is non-negligible in ℓ, where $\mathcal{A}_{ENT-TS-UDVS}^{CM-CPK-A}$ runs in time at most $\mathtt{t}$, makes at most q_H, q_{SP}, q_{SS}, q_{VP}, q_{TD}, q_{SD}, and q_{SK} queries to the random oracles, the $\boldsymbol{\mathcal{SPO}}$ oracle, the $\boldsymbol{\mathcal{SSO}}$ oracle, the $\boldsymbol{\mathcal{VPO}}$ oracle, the $\boldsymbol{\mathcal{TDO}}$ oracle, the $\boldsymbol{\mathcal{SDO}}$ oracle, and the $\boldsymbol{\mathcal{SKO}}$ oracle, respectively.*

4.5.5 Anonymity

The motivation of the anonymity property from ring signature schemes [RST01, BKM06, LW04, SW07] and threshold ring signature schemes [TWC+04, LWW03, BSS02], is adopted and their notations is adapted to realize the security of anonymity against full key exposure for TS-UDVS schemes. The anonymity property for TS-UDVS schemes requires that even if one obtains all secret keys of both signers and verifiers, and reviews a polynomial number of designated verifier signatures $\rho_1, ..., \rho_q$ on its choice of a message M designated to the same or different verifiers, where $\rho_1, ..., \rho_q$ are generated by the same signature holder using the same set of signatures $\sigma_1, ..., \sigma_t$, it is hard to persuade the other party, which signers are in the list of the threshold signers who generated a designated signature $\rho \in \{\rho_1, ..., \rho_q\}$ on a message $M \in \{M_1, ..., M_q\}$. The anonymity of TS-UDVS is defined as follows.

The oracles constructed in order to model the ability of adversaries breaking the anonymity of TS-UDVS schemes are described in Section 4.5.3. We denote by $EA\text{-}TS\text{-}UDVS$ the existential anonymity against a full key exposure of a TS-UDVS

scheme. Let $\mathcal{A}^{CM-CPK-A}_{EA-TS-UDVS}$ be the adaptively chosen message and chosen public key distinguisher and let $\mathcal{F}$ be a simulator. The following experiment between $\mathcal{F}$ and $\mathcal{A}$ is prescribed to show the existential anonymity against a full key exposure of a TS-UDVS scheme.

1. **Learning:** With any adaptive strategies, $\mathcal{A}$ arbitrarily sends queries to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VPO}}$, $\boldsymbol{\mathcal{TDO}}$ and $\boldsymbol{\mathcal{SDO}}$ oracles. The oracles respond according to their design.

2. **Challenge:** Let $\mathfrak{LS}^* = \{pk^*_{S_1}, ..., pk^*_{S_n}\}$ and $\mathfrak{TS}^* = \{pk^*_{S_{j_1}}, ..., pk^*_{S_{j_t}}\}$, where $j_1, ..., j_t$ are indexes of signers in a threshold t and $\mathfrak{TS} \subset \mathfrak{LS}$. When $\mathcal{A}$ decides to challenge $\mathcal{F}$, it outputs $i_0, i_1, M^*, \mathfrak{LS}^*, pk^*_V$. In return, $\mathcal{F}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$. Given $\mathfrak{TS}^* : pk^*_{S_{i_b}} \in \mathfrak{TS}^*; pk^*_{S_{i_{\sim b}}} \notin \mathfrak{TS}^*$, $\mathfrak{LS}^*$, pk^*_V and M^* as input, $\mathcal{F}$ makes a request for a designated verifier signature to the $\boldsymbol{\mathcal{TDO}}$ oracle and responds to $\mathcal{A}$ with ρ as an output from the $\boldsymbol{\mathcal{TDO}}$ oracle.

3. **Guessing:** Now, $\mathcal{A}$ is given an access to the $\boldsymbol{\mathcal{SKO}}$ oracle. After this, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ^{CM-CPK-A}_{EA-TS-UDVS}(.)$ be the success probability that $\mathcal{A}^{CM-CPK-A}_{EA-TS-UDVS}$ wins the above game.

Definition 4.6 *We say that the TS-UDVS scheme is (*$\mathtt{t}$*,*q_H*,*q_{SP}*,*q_{SS}*,*q_{VP}*,*q_{TD}*,*q_{SD}*, *q_{SK}*,*ϵ*)-secure existential anonymous against a full key exposure attack if there is no PPT CM-CPK-A distinguisher* $\mathcal{A}^{CM-CPK-A}_{EA-TS-UDVS}$ *such that the success probability* $Succ^{CM-CPK-A}_{EA-TS-UDVS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon - t/n$ *is non-negligible in* ℓ*, where* t *is a threshold of* n *signers and* t/n *is a probability that* $\mathcal{A}^{CM-CPK-A}_{EA-TS-UDVS}$ *can guess correctly without any advantage,* $\mathcal{A}^{CM-CPK-A}_{EA-TS-UDVS}$ *runs in time at most* $\mathtt{t}$*, makes at most* q_H*,* q_{SP}*,* q_{SS}*,* q_{VP}*,* q_{TD}*,* q_{SD}*, and* q_{SK} *queries to the random oracles, the* $\boldsymbol{\mathcal{SPO}}$ *oracle, the* $\boldsymbol{\mathcal{SSO}}$ *oracle, the* $\boldsymbol{\mathcal{VPO}}$ *oracle, the* $\boldsymbol{\mathcal{TDO}}$ *oracle, the* $\boldsymbol{\mathcal{SDO}}$ *oracle, and the* $\boldsymbol{\mathcal{SKO}}$ *oracle, respectively.*

4.6 The Proposed TS-UDVS Scheme

In this section, our scheme is presented based on the concept outlined in the previous section. First, some notations are defined. Let $\mathbb{G}_1, \mathbb{G}_T$ be multiplicative groups of prime order p. This is denoted by $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ a computationally efficient

bilinear mapping function $\hat{e}$, which maps $\mathbb{G}_1$ to $\mathbb{G}_T$. Let us denote by $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ a random one-way function that maps any string to group $\mathbb{G}_1$ and by $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ a collision-resistant hash function. The scheme then works as follows.

SKeyGen: Given a security parameter ℓ as input, a signer S randomly chooses a prime $p = poly(1^\ell)$ and a random generator $g \in \mathbb{G}_1$. Let $\mathsf{param} = (p, \hat{e}, g, H, h)$ denote the system parameter. A private key and a public parameter of the signer are generated as follows. Choose a random integer $x \in \mathbb{Z}_p$. Let us denote by $X = g^x$ the public key of the signer. Hence, *SKeyGen* returns $pk_S = (\mathsf{param}, X)$ and $sk_S = x$ as a public parameter and a private key of the signer, respectively.

VKeyGen: Given a security parameter ℓ as input, a verifier V complies with a trapdoor commitment scheme's setup function $Setup(1^\ell)$ to generate α, g_α, $Y = g_\alpha^y, sk_V = y$. Let $\bar{h} : \{0,1\}^* \rightarrow \mathbb{Z}_\alpha^*$ denote a collision-resistant hash function selected by V. V keeps sk_V as a private key and then publishes $pk_V = (\mathsf{param} = (\alpha, g_\alpha, \bar{h}), Y)$ as its public parameter. Note that the reader should be reminded that both signer and verifier key generation uses the same security parameter ℓ and hence, $|\alpha| = |p|$ and $\alpha \approx p$.

Sign: Given a message M, pk_S and sk_S, S computes $\sigma = H(M)^x$ as a BLS short signature on message M.

Verify: Given pk_S, σ and a message M, a signature holder SH checks whether $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H(M), X)$ holds or not. If not, then it outputs `reject`. Otherwise, it outputs `accept`.

TDesignate: Let $\mathfrak{TS}$ be a set of signers where the signature holder holds their signatures and t be a threshold where $t = |\mathfrak{TS}|$. Let i represent the index of the signer in $\mathfrak{LS}$ where $\mathfrak{TS} \subset \mathfrak{LS}$. Given pk_V, $\sigma_1, ..., \sigma_t$, $pk_{S_1}, ..., pk_{S_n}$ and a message M, SH computes a designated verifier signature ρ on message M as follows.

- First, provide the simulated signature of the signers $pk_{S_i} \in \mathfrak{LS} \setminus \mathfrak{TS}$ as follows. Select random integers $z_i, c_i \in \mathbb{Z}_p^*$ and compute

$$Z_i \;=\; H(M)^{z_i},\; R_i = \hat{e}(Z_i, g)\hat{e}(H(M), X_i)^{c_i}.$$

- Secondly, for the signers $pk_{S_i} \in \mathfrak{TS}$, compute as follows. Select a random integer $r_i \in \mathbb{Z}_p^*$ and compute $R_i = \hat{e}(H(M), g)^{r_i}$.
- Next, let $\omega \stackrel{def}{=} R_1||...||R_n$. Then compute the first part of the designated verifier signature with a verifier's public parameter as follows. Select a random integer $r_V \in \mathbb{Z}_\alpha$ and compute

$$\begin{aligned} \Psi &\stackrel{def}{=} M||\omega||pk_{S_1}||...||pk_{S_n} \\ c_0 &= T_V = h(g_\alpha^{r_V} Y^{\bar{h}(\Psi)}). \end{aligned}$$

- Finally, from the Shamir's secret sharing technique [Sha79], let f be a polynomial such that it satisfies the following conditions:

$$deg(f) = n - t \bigwedge f(0) = c_0 \bigwedge \forall i \in \mathfrak{LS} \setminus \mathfrak{TS} : f(i) = c_i.$$

  Then, for every signer $i \in \mathfrak{TS}$, compute the rest of the designated verifier signature as follows.

$$c_i = f(i),\ Z_i = H(M)^{r_i} \cdot \sigma_i^{-c_i}.$$

Therefore, a designated verifier signature ρ is $(\mathfrak{LS}, f, Z_1, ..., Z_n, r_V)$. Output ρ as a designated verifier signature on message M.

DVerify: Given $pk_{S_1}, ..., pk_{S_n}$, pk_V, ρ and a message M as input, the designated verifier V first computes as follows.

$$\begin{aligned} \forall i \in \mathfrak{LS} \bigcup \{0\}, c_i &= f(i). \\ \omega &= \hat{e}(Z_1, g)\hat{e}(H(M), X_1)^{c_1}||...||\hat{e}(Z_n, g)\hat{e}(H(M), X_n)^{c_n}. \\ \Psi &= M||\omega||pk_{S_1}||...||pk_{S_n}. \end{aligned}$$

Then V checks whether $c_0 \stackrel{?}{=} h(g_\alpha^{r_V} Y^{\bar{h}(\Psi)})$ holds or not. If not, then it outputs `reject`. Otherwise, it outputs `accept`.

DSimulate: Given sk_V, pk_V, $pk_{S_1}, ..., pk_{S_n}$ and a message M as input, V computes as follows.

- First, randomly generate c_0 as follows: select a random integer $k', r'_V \in \mathbb{Z}_\alpha$ and compute $c_0 = h(g_\alpha^{r'_V} Y^{k'})$.

- Second, randomly select a polynomial f such that

$$deg(f) = n - t \bigwedge f(0) = c_0.$$

  Then, for every signer $i \in \mathfrak{LS}$, compute $c_i = f(i)$.

- Next, for each signer $pk_{S_i} \in \mathfrak{LS}$, compute the first part of the designated verifier signature as follows.

$$Z_i \overset{\$}{\leftarrow} \mathbb{G}_1,\ R_i = \hat{e}(Z_i, g)\hat{e}(H(M), X_i)^{c_i}.$$

- Finally, recompute r_V with the verifier private key as follows.

$$\Psi = M||\omega||pk_{S_1}||...||pk_{S_n},\ r_V = r'_V + y \cdot k' - y \cdot \bar{h}(\Psi).$$

Therefore, a simulated designated verifier signature by the verifier is $\rho = (\mathfrak{LS}, f, Z_1, ..., Z_n, r_V)$.

4.7 Security Analysis of TS-UDVS

4.7.1 Completeness

Completeness of a Signature: The signature verification is as follows.

$$\begin{aligned} \hat{e}(\sigma, g) &\overset{?}{=} \hat{e}(H(M), X) \\ \hat{e}(H(M)^x, g) &\overset{?}{=} \hat{e}(H(M), g^x) \\ \hat{e}(H(M),) &\overset{?}{=} \hat{e}(H(M), g^x) \end{aligned}$$

Completeness of a (Simulated) Designate Verifier Signature: Given a public parameter of the signers $\mathfrak{LS}$, a public parameter of the designated verifier pk_V, a private key of the designated verifier sk_V, a message M and a designated verifier signature ρ, one first computes as follows.

$$\begin{aligned} \forall i \in \mathfrak{LS} \bigcup \{0\}, c_i &= f(i). \\ \omega &= \hat{e}(Z_1, g)\hat{e}(H(M), X_1)^{c_1}||...||\hat{e}(Z_n, g)\hat{e}(H(M), X_n)^{c_n}. \\ \Psi &= M||\omega||pk_{S_1}||...||pk_{S_n}. \\ \text{Check } c_0 = h(g_\alpha^{r'_V} Y^{k'}) &\overset{?}{=} h(g_\alpha^{r_V} Y^{\bar{h}(\Psi)}) \\ h(g_\alpha^{r'_V} g_\alpha^{y \cdot k'}) &\overset{?}{=} h(g_\alpha^{r'_V + y \cdot k' - y \cdot \bar{h}(\Psi)} g_\alpha^{y \cdot \bar{h}(\Psi)}) \\ h(g_\alpha^{r'_V + y \cdot k'}) &\overset{?}{=} h(g_\alpha^{r'_V + y \cdot k'}). \end{aligned}$$

Hence, the above statements show that the simulated threshold-signers designated verifier signature does indeed hold. □

4.7.2 Unforgeability

Theorem 4.4 *Our universal designated verifier signature with threshold-signers scheme is existential unforgeability under an adaptive chosen message, chosen public key attack and insider corruption if the CDH assumption holds in the random oracle model.*

Proof: Suppose that there exists a forger $\mathcal{A}$, which runs the existential unforgeability game defined in Section 4.5.3, then we will show that there exists an adversary $\mathcal{F}$ that solves the CDH problem by using $\mathcal{A}$. Start with the construction of oracles as they are designed in Section 4.5.3. Then construct $\mathcal{F}$ and run it over $\mathcal{A}$ with the existential unforgeability game defined in Section 4.5.3. Next, summarise the success probability of the existential unforgeability game under an adaptive chosen message, chosen public key attack and insider corruption. Finally, from the existential unforgeability game and its success probability, a conclusion can be drawn that the success probability of solving the CDH problem is non-negligible if the success probability of the above game is non-negligible.

The oracles can be constructed and the existential unforgeability game can be run as follows. Given g, g^a and g^b as an instance of the CDH problem, $\mathcal{F}$ sets g^b as one of the answers for the hash query to the random oracle. Next, $\mathcal{F}$ sets $X^* = g^{x^*} = g^a$ in one of the signer's public parameter defined as pk_{S^*}. The aim is to obtain g^{ab} from running the existential unforgeability experiment. Assuming that there exists an algorithm managing the list of each query and such algorithm will be omitted. Let $\Psi = M||\sigma_1||pk_S$ and $\Omega = M^*||\sigma_1^*||pk_S$. From the above setting, it is easy for $\mathcal{F}$ to construct the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{VPO}$, $\mathcal{TDO}$, $\mathcal{SDO}$, $\mathcal{SKO}$ and the random oracles $\mathcal{HO}$ as follows.

$\mathcal{HO}$ oracle: Select $d \xleftarrow{\$} \{0,1\}$ such that the probability of $d = 1$ is $\frac{1}{q_H}$. If $d = 1$ then set $M = \Omega$ and $H(M) = g^b$ and return $H(M)$. Otherwise, $k \xleftarrow{\$} \mathbb{Z}_p$; $H(M) = g^k$ and return $H(M)$. Then $\mathcal{HO}$ keeps a pair of $H(M)$ and k in the list, which it is accessible only by $\mathcal{F}$. For a query for $h(\Psi)$, $\mathcal{HO}$ randomly selects $k_1 \xleftarrow{\$} \mathbb{Z}_p$; $h(\Psi) = k_1$ and return $h(\Psi)$.

$\mathcal{SPO}$ oracle: Let $\mathsf{param} = (p, \hat{e}, h, H, g)$ be the system parameter for each signer.

$\boldsymbol{\mathcal{SPO}}$ chooses $\dot{d} \stackrel{\$}{\leftarrow} \{0,1\}$ such that the probability of $\dot{d}=1$ is $\frac{1}{q_{SP}}$. If $\dot{d}=1$ then set $X^* = g^a$ and return $pk_S^* = (\mathsf{param}^*, X^*)$. Otherwise, $t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$; $X = g^t$ and then return $pk_S = (\mathsf{param}, X)$ and keep t as a private key.

$\boldsymbol{\mathcal{SKO}}$ oracle: the $\boldsymbol{\mathcal{SKO}}$ oracle responds to every query on input pk_S and pk_V with its corresponding private key. Expect for pk_S^*, $\boldsymbol{\mathcal{SKO}}$ outputs $\perp$.

$\boldsymbol{\mathcal{SSO}}$ oracle: Let $r_1 \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. Given pk_S and M as input, $\boldsymbol{\mathcal{SSO}}$ outputs $\sigma = H(M)^t$ for every query except when $pk_S = pk_S^*$ and $M = M^*$. In the case of (pk_S^*, M^*), it outputs $\perp$.

$\boldsymbol{\mathcal{TDO}}$ oracle: Let t_i be a secret of i-th signer in $\mathfrak{LS}$. Given $\mathfrak{LS}$, $\mathfrak{TS}$, pk_V and M as input, where $pk_S^* \notin \mathfrak{TS}$ and $M \neq \Omega$, $\boldsymbol{\mathcal{TDO}}$ obtains each signer's private key t_i of the public parameter in $\mathfrak{TS}$ from $\boldsymbol{\mathcal{SPO}}$. Then $\boldsymbol{\mathcal{TDO}}$ computes a threshold-signers designated verifier signature as described in $Sign$ and $TDesignate$ in Section 4.6. In the case of $pk_S^* \in \mathfrak{TS}$ and $M \neq \Omega$, $\mathcal{F}$ obtains a random integer k associated with $H(M)$ from a list of $H(M)$ and k maintained by $\boldsymbol{\mathcal{HO}}$. $\mathcal{F}$ then gives it to $\boldsymbol{\mathcal{TDO}}$. $\boldsymbol{\mathcal{TDO}}$ computes $\sigma_i = X_i^k$, where i is an index of each signer in $\mathfrak{TS}$. Then $\boldsymbol{\mathcal{TDO}}$ computes a threshold-signers designated verifier signature as described in $TDesignate$ in Section 4.6. In the case of $pk_S^* \in \mathfrak{TS}$ and $M = \Omega$, $\boldsymbol{\mathcal{TDO}}$ outputs $\perp$.

$\boldsymbol{\mathcal{VPO}}$ and $\boldsymbol{\mathcal{SDO}}$ oracles: These oracles are straightforward as described in Section 4.5.3.

An access to the above oracles is provided to $\mathcal{A}$. Assume that a hash of message M from the random oracle $\boldsymbol{\mathcal{HO}}$ is always queried before $\mathcal{A}$ makes queries to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{TDO}}$ and $\boldsymbol{\mathcal{SDO}}$ oracles, or before it outputs a potential forgery, denoted by $(M^*, \rho_*, \mathfrak{LS}^*, pk_V^*)$.

In the end, after processing an adaptive strategy with the above oracles, $\mathcal{A}$ outputs a forged threshold-signers designated verifier signature ρ_* on a message M^* with respect to $\mathfrak{LS}^*, pk_V^*$. $\mathcal{A}$ wins the game if a message M^* is never submitted to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{TDO}}$ and $\boldsymbol{\mathcal{SDO}}$ oracles and at least $n+1-t$ signers' secret keys in $\mathfrak{LS}$ have never been queried to the $\boldsymbol{\mathcal{SKO}}$ oracle. After the above experiment, we obtain a valid threshold-signers designated verifier signature ρ_* on a message M^* with respect to $\mathfrak{LS}^*, pk_V^*$. From the above signature, the probability of success will be shown in the next paragraph and then, after running the second experiment, how

to obtain the original signature $\sigma^* = H(\Omega)^{x^*} = (g^b)^a = g^{ab}$ will be demonstrated, which is also an answer for the CDH problem.

Let e denote the base of the natural logarithm and q be a polynomial upper bound on the number of queries that $\mathcal{A}$ makes to the $\boldsymbol{\mathcal{HO}}$ and $\boldsymbol{\mathcal{SKO}}$ oracles. Now the probability of events such that $\mathcal{F}$ does not abort during the simulation is analysed as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{SKO}}$.*

  The probability of this event is greater than $(1 - \frac{1}{q_{SP}})^{q_{SP}-1} \approx \frac{q_{SP}}{e\cdot(q_{SP}-1)}$.

- E_2: *$\mathcal{F}$ does not abort when issuing queries to the $\boldsymbol{\mathcal{SSO}}$.*

  The fact is that $\mathcal{A}$ needs at least one hash value and one signer secret to output a forgery, and hence, $q_{SS} \leq (q_H - 1) \cdot (q_{SP} - 1)$. Therefore, the probability of this event is greater than $(1 - \frac{1}{q_H \cdot q_{SP}})^{q_{SS}} = (1 - \frac{1}{q_H \cdot q_{SP}})^{(q_H-1)\cdot(q_{SP}-1)} \approx \frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{(q_H+q_{SP}-1)}$.

- E_3: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{TDO}}$.*

  Similar to the E_2 event, where $q_{TD} \leq (q_H - 1) \cdot (q_{SP} - 1)$, the probability of this event is greater than $(1 - \frac{1}{q_H \cdot q_{SP}})^{q_{TD}} = (1 - \frac{1}{q_H \cdot q_{SP}})^{(q_H-1)\cdot(q_{SP}-1)} \approx \frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{(q_H+q_{SP}-1)}$.

Let $Succ^{CM-CPK-A}_{EUF-TS-UDVS} = \epsilon$ be the success probability that $\mathcal{A}$ wins the game.

The probability that $\mathcal{A}$ wins the above game and outputs a message M^* and a signer's public parameter pk^*_S is $\frac{\epsilon}{q_H \cdot q_{SP} - max(q_{TD}, q_{SS})} \leq \frac{\epsilon}{q_H + q_{SP} - 1}$ where $q_{SS} \leq (q_H - 1) \cdot (q_{SP} - 1)$, $q_{TD} \leq (q_H - 1) \cdot (q_{SP} - 1)$, and q_H and q_{SP} are the maximum number of queries that $\mathcal{A}$ made to the random oracle and the $\boldsymbol{\mathcal{SPO}}$ oracle, respectively. Putting the above probabilities together, the probability can be resolved such that $\mathcal{F}$ does not abort during the simulation and $\mathcal{A}$ wins the game with M^*, pk^*_S is about $\frac{\epsilon}{q_H+q_{SP}-1} \cdot \frac{q_{SP}}{e\cdot(q_{SP}-1)} \cdot (\frac{1}{e} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{q_H+q_{SP}-1})^2 = \frac{\epsilon}{q_H+q_{SP}-1} \cdot \frac{q_{SP}}{e^3\cdot(q_{SP}-1)} \cdot (\frac{q_H \cdot q_{SP}}{q_H \cdot q_{SP}-1})^{2(q_H+q_{SP}-1)} > \frac{\epsilon}{e^3(q_H+q_{SP}-1)}$.

From the above probability, it is obvious that the probability, where $\mathcal{F}$ successfully runs the above simulation and $\mathcal{A}$ wins the game with $M^* = \Omega$ and $pk^*_S \in \mathfrak{LG}^*$, is non-negligible compared with the probability that $\mathcal{A}$ wins the game where $q > (q_H + q_{Sp} - 1)$.

Hence, with a non-negligible running time due to the forking lemma [PS00, BN06], $\mathcal{F}$ also obtains another set of forgeries by rerunning the experiment with $\mathcal{A}$ as follows.

- First, reset $\mathcal{A}$ to the initial state.
- Second, provide the same setting as in the previous experiment but with a new set of verifiers' public parameters.
- Finally, rerun the experiment with the same random tape as the first experiment.

At the end of the second experiment, $\mathcal{F}$, with non-negligible probability, outputs a forgery $(M^*, \rho_{**}, \mathfrak{LS}^*, pk_V^{**})$. Since the setting is the same as the first experiment, $\mathcal{A}$ outputs $M^*, \mathfrak{LS}^*$ such that $pk_S^* \in \mathfrak{LS}^*$ with the same probability as in the first experiment. However, in the second experiment, $\mathcal{A}$ is given with a new set of verifiers' public parameters hence, $\mathcal{A}$ outputs ρ_{**}, pk_V^{**}, which are different outputs from those in the first experiment.

From the above outputs by $\mathcal{A}$, $\mathcal{F}$ obtains $Z^* = H(\Omega)^{r^*} \cdot \sigma_*^{-c^*}$ and $Z^{**} = H(\Omega)^{r^{**}} \cdot \sigma_{**}^{-c^{**}}$ from ρ_*, ρ_{**}, where both Z^* and Z^{**} are associated with pk_S^* and $H(\Omega)$. Since $r^* = r^{**}$, $\mathcal{F}$ computes $\sigma^* = (Z^*/Z^{**})^{1/(c^{**}-c^*)}$. In fact, $\sigma^* = H(\Omega)^{x^*} = (g^b)^a = g^{ab}$. Therefore, $\mathcal{F}$ outputs σ^* as an output for the CDH problem with non-negligible probability as mentioned above. The above simulation shows that the probability of success in attacking our TS-UDVS scheme by existential unforgeability under an adaptive chosen message, chosen public key attack and insider corruption is negligible since the probability of solving the CDH problem is negligible.

4.7.3 Non-transferable Privacy

Theorem 4.5 *In the random oracle model, the proposed universal designated verifier signature with threshold-signers scheme offers existential non-transferable privacy against adaptively chosen message and chosen public key distinguisher* $\mathcal{A}_{ENT-TS-UDVS}^{CM-CPK-A}$.

Proof: Theorem 4.5 is proved by running the existential non-transferable privacy game defined in Section 4.5.4 and showing that the success probability of distinguisher $\mathcal{A}$ in that game attacking our TS-UDVS scheme is negligible. This is begun with the construction of oracles and the existential non-transferable privacy game defined in Section 4.5.4. Then we show that both probabilities of the distribution of the DVS signature generated by the signature holder and designated verifier are equal. Finally, the indistinguishability of both valid and simulated designated verifier signatures in our TS-UDVS scheme will be concluded.

The following simulation shows that, running on both the signer and the verifier, a simulator $\mathcal{F}$ generates a designated verifier signature, which is indistinguishable whether a signer or a verifier generated it. In the first step, since $\mathcal{F}$ can arbitrarily generate a public-private key pair for $\mathcal{A}$, $\mathcal{F}$ constructs straightforward oracles as described in Section 4.5.4. $\mathcal{A}$ is given access to those oracles. Note that for every private key corresponding to its queried public parameters, $\mathcal{F}$ keeps them secretly to itself.

The distribution of the $\boldsymbol{\mathcal{TDO}}$ oracle is then analysed. There is one random integer involved in the production of the signature for each signer, $n-t$ random integers involved in the production of a polynomial function and one random integer involved in the production of the designated verifier signature related to the verifier. Therefore, there are in total $2n-t+1$ uniformly random numbers used in the generation of the designated verifier signature besides the secret keys of the signers. Let ρ_{DV} denote a designated verifier signature in the distribution of the $\boldsymbol{\mathcal{TDO}}$ oracle. With the above random integers and secret keys of the signer $sk_{S_1}, ..., sk_{S_t}$, $\mathcal{F}$ randomly produces the designated verifier signature as described in the $TDesignate$ algorithm in Section 4.6. Hence, since each random integer is selected from $\mathbb{Z}_p^*$ or $\mathbb{G}_1$, if a designated verifier signature ρ_* is randomly chosen, then the probability that ρ_* is in the distribution of the $\boldsymbol{\mathcal{TDO}}$ oracle is $\mathbf{Pr}[\rho_{DV} = \rho_*] = \frac{1}{p^{(2n-t+1)}}$.

Next, the distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle is analysed. There is one random integer involved in the production of the signature for each signer, $n-t$ random integers involved in the production of a polynomial function. There are also other two random integers (k', r'_V) involved in the production of the designated verifier signature related to the verifier; however, these two integers work to achieve one output, which is r_V. Hence, these are viewed together as one random variable. Therefore, there are in total $2n-t+1$ uniformly random numbers used in the generation of the simulated designated verifier signature, in addition to the private key of the verifier. Let ρ_{DS} denote a designated verifier signature in the distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle. With the above random integers and secret keys of the signer $sk_{S_1}, ..., sk_{S_t}$, $\mathcal{F}$ randomly produces the designated verifier signature as described in the $DSimulate$ algorithm in Section 4.6.

More precisely, since each random integer is selected from $\mathbb{Z}_p^*$ or $\mathbb{G}_1$, if a designated verifier signature ρ_* is randomly chosen, then the probability that ρ_* is in the distribution of the $\boldsymbol{\mathcal{SDO}}$ oracle is $\mathbf{Pr}[\rho_{DS} = \rho_*] = \frac{1}{p^{(2n-t+1)}}$.

Finally, the above probabilities claim that one cannot distinguish whether a randomly given valid universal designated verifier signature is generated by the $\mathcal{TDO}$ or $\mathcal{SDO}$ oracles. Hence, our TS-UDVS scheme satisfies the non-transferable privacy property. □

4.7.4 Anonymity

Theorem 4.6 *With probability at most $t/n + \epsilon$, where ϵ is negligible, our universal designated verifier signature with threshold-signers scheme offers anonymity against a full key exposure.*

Proof: Theorem 4.6 is proven by showing that the success probability of distinguisher $\mathcal{A}$ attacking the TS-UDVS scheme when running anonymity against the full key exposure game defined in Section 4.5.5 is negligible, when it excludes t/n. The following simulation shows that, running on signers, a simulator $\mathcal{F}$ generates a designated verifier signature that is indistinguishable in which signer is in a list of threshold signers. In the first step, since $\mathcal{F}$ can arbitrarily generate a public-private key pair for $\mathcal{A}$, $\mathcal{F}$ constructs straightforward oracles as described in Section 4.5.5. Except for the $\mathcal{VPO}$ oracle, since $\mathcal{A}$ has taken over control of the verifier, $\mathcal{A}$ can arbitrarily run $VKeyGen$ to generate a public-private key pair for the verifier by itself. Then, $\mathcal{A}$ is given access to those oracles and is run with anonymity against the full key exposure game defined in Section 4.5.5. Note that for every private key corresponding to its queried public parameters, $\mathcal{A}$ can arbitrarily issue a request for the signer's private key to the $\mathcal{SKO}$ oracle.

Discussion:

First, the polynomial f in the $TDesignate$ algorithm in Section 4.6 uniquely outputs c_0 and c_i, where $i \in \mathfrak{TS}$. c_0 and c_i are uniquely generated by the random oracle and random tapes consumed by $\mathcal{F}$. Therefore, the polynomial f can be considered as a random function selected from the entire polynomials over $GF(p)$ with degree $n-t$. Hence, the distribution of c_i, where $i \in \mathfrak{TS}$, is also uniform over $GF(p)$. Second, for each Z_j, where $j \in \mathfrak{LS}$, a random variable (either z_j or r_j) is independently chosen and uniformly distributed over $GF(p)$. Therefore, Z_j is uniformly distributed over $GF(p)$. Finally, it can be seen that, for a fixed message M and a fixed set of signers' public keys $\mathfrak{LS}$, there are p^n possible solutions for $\mathcal{F}$ to output $(Z_1, ..., Z_n)$. The possible solutions above are uniformly and independently distributed; hence, it does not matter whether $\mathcal{A}$ possesses unbounded computing resources and all the secret

keys, and how many participanting signers (t) there are to generate signatures. To identify any one of the participant signers, advantage over random guessing is negligible. □

4.8 Conclusion

In this chapter, the notion of one-time universal designated verifier signatures (OT-UDVS) and the notion of universal designated verifier signatures with threshold-signers (TS-UDVS) were introduced. The notion of one-time universal designated verifier signatures allows a signer to limit his/her signature to be used only one time to compute a designated verifier signature. If a signature holder computes more than one designated verifier signature from the signer's signature then the signer's signature will be revealed. A definition of one-time universal designated verifier signature scheme and its security model were presented. A concrete construction of one-time universal designated verifier signature scheme that is secure in our model was given. The notion of universal designated verifier signatures with threshold-signers allow a signature holder to provide the anonymity for the signer(s) and the signature(s) that he has in his possession. Moreover, the privacy between the signer(s) and the designated verifier is also provided in this notion. In other words, a designated verifier signature constructed by the signature(s) from a signer (or t signers) in the list of n signers convinces only the designated verifier and the designated verifier only know that at least one signature (or t signatures) of n possible signatures is (are) valid. A definition of universal designated verifier signatures with threshold-signers schemes and its security model were given. A concrete construction of universal designated verifier signatures with threshold-signers scheme that is secure in our model is also presented.

Chapter 5

Policy-controlled Signatures Scheme and Its Applications

In this chapter, a new primitive algorithm called a "policy-controlled signature" is described. Part of this chapter appeared in *ICICS 2009* [TSM09b]. Its applications, which are the "universal policy-controlled signature" and the "multi-level controlled signature", are also provided in this chapter.

5.1 Introduction

The principle of policy-based cryptography was introduced by Bagga and Molva in [BM05]. In the policy-based signature scheme in [BM05], a signer is allowed to sign a message correctly if and only if, the signer satisfies an assigned policy. However, what if the situations were reversed? So a signer is allowed to assign the policy such that only a verifier satisfying the signer's policy can verifier a signature. In a policy-controlled signature scheme, a verifier is allowed to verify a policy-controlled signature if and only if, the verifier satisfies an assigned policy. For instance, we are dealing with some sensitive resources or messages that need to specify limiting multiple verifiers to gain access to the authenticity of the message. Moreover, in some cases, the authenticity of these messages requires privacy, which means it must be kept between the signer and the verifiers. In other words, only a verifier that satisfies the necessary condition or policy can verify the authenticity of these messages. The other parties should not be convinced about the authenticity of these messages. Let us describe more about the applications in the following scenario. Alice is a CEO of a company. She would like to inform the board members about issues regarding the future of the company. However, this message should not be revealed as an official document created by herself for the public. Hence, a signature on this message should be verifiable only by the board members. In the above case, a designated

verifier signature scheme could be applied where the number of board members is small. Alice could generate signatures on a message for each board member. However, this solution is not efficient, and what if Alice also wants to inform the major share holders and directors? In this case a designated verifier signature is not an appropriate solution. By limiting the group of people who can verify authentication of a message, a signature on a message should not be verifiable by any other people outside this authorised group. Furthermore, the right of verification of the message by the authorised group is limited, so the authorised verifier should not be able to relay this conviction or convince any other third party outside the authorised group. Another example is a single sign-on system. A single sign-on system is an access control system for multiple web service systems or application service systems. Generally, a user log onto the identity identification server and can access web services or application services that are registered with the identity identification server according to the user's privileges. A single sign-on system provides the advantage that a user can access many resources once his/her identity is authenticated. There are many ways to construct a single sign-on system. However, when the number of services and users is huge, the problem of the size of tokens or credentials also increases linearly. To cater for such a system, a policy-based cryptography is a good choice to consider. The policy-based cryptography can be applied to group services with the similar requirements. Consider the following scenario. Let SIIS be a single sign-on identity identification server. Alice would like to obtain a token for these services, so she identifies herself to SIIS. After SIIS successfully identifies Alice, it produces a token (it can be a signature on the time stamp) and gives it to Alice. Alice runs an interactive proof to the service with this token. Finally, Alice can access the service. If we apply this policy to control access, then a token is supposed to be verifiable only by a service that possesses the right credential for the policy. The above is a typical scenario where policy-controlled signature schemes are useful primitive schemes. It can be argued that we can achieve the above solution by signing a message with a regular signature scheme, and then encrypt it with policy-based encryption scheme introduced in [BM05]. However, a verifier can decrypt the ciphertext and then distribute the signature; hence, the signature is publicly verifiable. Therefore the solution above violates the requirements as stated above. To extend the application of policy-controlled signatures, we propose using universal policy-controlled signature schemes (UPCS). Now, a signer signs on a message and

gives a signature to a policy signer. Then the policy signer computes a (universal) policy-controlled signature from a signature generated by a signer. With the policy assigned by the policy signer, this (universal) policy-controlled signature can be verified only by a verifier who obtains sufficient credentials to satisfy the policy. In addition, this (universal) policy-controlled signature also represents a proof that the policy signer has obtained a signature on a message that has been generated by a signer. Let us extend the first scenario as follows. Alice, a CEO of a company, assigns Peter as an internal auditor to audit the financial account of each department in her company. Alice may want this matter to be private so only the involved parties can verify that Alice authorises Peter to do the account audition. Alice signs a message that authorised Peter as the internal auditor and give it to Peter. Since the company contains many departments, Peter may want show this message and signature to each department one at a time. A policy-controlled signature scheme can be applied to the above case. However, Alice may not know which policy she should apply first in order to let Peter correctly auditing every department since the management structure of each department may be different. Hence, this solution cannot be resolved efficiently and correctly. The simple solution is allowing Peter to assign the policy for each department where it is appropriated. The above scenario can be simply solved with universal policy-controlled signature schemes.

Another extension of the policy-controlled signature described in this chapter is the multi-level controlled signature. Multi-level controlled signatures allow a signer to assign the level of security in order to verify the signer's signature. Let us extend the first scenario as follows. Alice is a CEO of an organisation with n subbranch. She wants to send a message about a change of management in the company to all the employees who hold a position higher than or equal to manager of each subbranch. Since the organisation may contain many levels of management, if Alice applies this with policy controlled signatures then she needs to assign the following policy: "POLICY= board members of the main branch or board members of the first subbranch or .... or manager of the n-th subbranch". With multi-level controlled signature schemes, Alice can limit the level of position of the verifiers who are able to verify her signature on a message. Hence, other employees in positions lower than manager should not be able to verify her signature. Let the security level define as the level of the security that is assigned in the system and let the i-security level be the i-th level of the security in the system. For instance, there are 13 levels of the security assigned by the system. The 11-security level is the security at the

eleventh level from 13 levels of the security. In policy-controlled signature, a signer can assigned the policy such that it works in the same way as multi-level controlled signature. For example, a signer assigns the 11-security level to be the minimum security level from 13 security levels. A signer can assign the policy "POLICY=(11-security level or 12-security level or 13-security level)" for policy-controlled signature to work as multi-level controlled signature. However, what if the number of level is increased? Let say the assigned minimum security is at the 11-security level from 100 security levels. Then, the policy for this security level will be "POL=(11-security level or 12-security level,..., or 100-security level)", which is very long compared to a multi-level policy assigned in the multi-level controlled signatures that is "POLICY= more than or equal 11-security level". The notion of multi-level controlled signatures eliminates the unnecessary chain of attributes in the policy when it can be assign as the security level.

5.1.1 Related Work

In Financial Cryptography Conference 2005, Bagga and Molva [BM05] proposed a novel notion called "Policy-based Cryptography". The notion of policy-based cryptography includes policy-based encryption schemes and policy-based signature schemes. Now, a signer (or a receiver, respectively) can sign on a message (or decrypt the ciphertext, respectively) if he/she satisfies the assigned policy. The notion of policy-based encryption provides the authorisation and the integrity of the message. The notion of policy-based signature ensures the message's integrity, authenticity and the non-repudiation of the signer. In [BM06], an improved construction of policy-based encryption was proposed by Bagga and Molva. In this improved policy-based encryption scheme, a user's public key is involved in the process of generating a user's credentials. This improvement is designed to prevent a collision attack, where attackers can obtain multiple sets of credentials that satisfy the assigned policies in order to create a new set of credentials. Hence, both a set of credentials and a user's public key are needed in order to decrypt the ciphertext.

Another related notion is the notion of Hierarchical Identity based Encryption/Signature. The Hierarchical Identity based Encryption (HIBE) system [GS02, HL02, BBG05, BW06] is a concept that unites between a Hierarchy system and an Identity-based Encryption (IBE) system [Sha84, BF01], where an identity at level k of the hierarchical system can issue a private key for its descendant identity, but it

cannot decrypt a message on behalf of other identities except its descendants. The Hierarchical Identity based Signature (HIBS) scheme [GS02, CHYC04, HL02] is a natural conversion from the HIBE scheme. Similar to the HIBE, the ancestor identity of the hierarchical system can issue a private key for its descendant identities. However, it cannot sign a message on behalf of other identities except its descendants. The purpose of HIBE systems is to reduce the bottleneck in a large network, where the PKG of the IBE system is applied, and to limit the scope of key escrow. The HIBS, however, is similar to policy-based signature schemes in that it only provides the integrity of message as well as the authenticity and non-repudiation of a signer, but it does not provide the authorisation for the verifier.

5.1.2 Our Contributions

In this chapter, we introduce the notion of policy-controlled signature (PCS) schemes. Our notion allows a receiver to verify the authenticity of a signed message if and only if, the receiver satisfies the policy specified by the sender (or signer). We formalise this notion and define its security model and requirements. Furthermore, we provide a concrete construction that is proven secure in our model. We also extend the notion of policy-controlled signature to universal policy-controlled signature (UPCS) and multi-level controlled signature (MLCS).

Chapter Organisation

This chapter is organised as follows. In the next section, we will review some preliminaries that will be used throughout this Chapter. The definition of PCS and its security notions will be presented in Section 5.2. Next, our concrete scheme of PCS and its security proof will be provided in Section 5.3 and 5.4. In Section 5.5, we give a definition of universal policy controlled signature (UPCS). Next, in Section 5.6 and 5.7, we present our concrete scheme of UPCS and its security proof. In Section 5.8, we introduce the definition of MLCS and its security notions. Next, the first concrete scheme, together with its security proof will be provided in Section 5.9 and 5.10. Then, in the Section 5.11 and 5.12, we will provide the second concrete scheme and its security proof. Finally, the conclusion of the chapter will be presented in the last section.

5.2 Definition of Policy-controlled Signature Scheme (PCS)

In this section, we give a definition of policy-controlled signature (PCS) schemes that allow a signer to limit the verification of his/her signature by using a policy. In other words, only a verifier that satisfies the policy specified by the signer can verify the policy-controlled signature. We provide an outline of our PCS scheme as follows.

5.2.1 Outline of PCS

Let TA denote a trusted authority that issues credentials associated with policies. Let CA denote a certificate authority that generates system parameter and certifies public keys for all parties. There are two main players in a policy-controlled signature scheme, namely, a signer and a verifier. A signer S generates a signature that can be verified *only* when a verifier V holds a credential satisfying the policy. V holds credentials issued by TA.

Let A denote an assertion issued by TA. Each assertion A may be a hash value of some certain statements, such as "CIA agent". We define $\mathbb{P}$ to be a policy that contains a set of assertions $\mathbb{P} = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ where i, j, k are indexes. In general, a policy $\mathbb{P}$ can be represented in the disjunctive normal form (DNF) or the conjunctive normal form (CNF), or any combination of both forms. The policy $\mathbb{P}$ is in the DNF when $a = 1$ and in CNF when $\forall i, \forall j : a_{i,j} = 1$. For example, a policy in DNF is as follows: $\mathbb{P}$ = "$(A_{1,1,1} \bigwedge A_{1,1,2}) \bigvee (A_{1,2,1} \bigwedge A_{1,2,2}) \bigvee A_{1,3,1} \bigvee (A_{1,4,1} \bigwedge A_{1,4,2})$". For simplicity, let $\mathbb{C}_{i,j,k}$ be a credential for an assertion and let $\mathbb{B} = [\{\mathbb{C}_{1,1,1}, ..., \mathbb{C}_{a,1,a_{i,1}}\}, ..., \{\mathbb{C}_{1,a_i,1}, ..., \mathbb{C}_{a,a_i,a_{i,j}}\}]$ be a set of the entire possible set of credentials that satisfy the policy $\mathbb{P}$. Note that i, j, k are indexes for assertions associated with the credentials. Let $\{\mathbb{C}_{i,j,k}\} = \mathbb{C}_{1,j,1}, ..., \mathbb{C}_{a,j,a_{i,j}}$ be a set of credentials, which it may or may not be a set of credentials in $\mathbb{B}$. Let $\{\mathbb{C}\} = \mathbb{C}_{1,1,1}, ..., \mathbb{C}_{a,a_i,a_{i,j}}$ be the entire credentials, where i, j are indexes.

Without losing generality, we assume that all parties must comply with the registration protocol with a certificate authority CA to obtain a certificate on their respective public keys. A policy-controlled signature scheme Σ is a 6-tuple ($Setup$, $TKeyGen$, $SKeyGen$, $CreGen$, $PSign$, $PVerify$), which is described as follows.

System Parameter Generation ($Setup$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, outputs the system parameter param. That is,

$$Setup(1^{\ell}) \rightarrow \mathsf{param}.$$

TA Key Generator ($TKeyGen$):

This is a probabilistic algorithm that, given the system parameter param as input, outputs the private key (sk_{TA}) and the public parameter (pk_{TA}) of a trusted authority. That is,

$$TKeyGen(\mathsf{param}) \rightarrow (pk_{TA}, sk_{TA}).$$

Signer Key Generator ($SKeyGen$):

This is a probabilistic algorithm that, given a system parameter param and a public key of the trusted authority pk_{TA} as input, outputs the private key (sk_S) and the public parameter (pk_S) of the signer. That is,

$$SKeyGen(\mathsf{param}, pk_{TA}) \rightarrow (pk_S, sk_S).$$

Verifier Credential Generator ($CreGen$):

This is a probabilistic algorithm that, given the system parameter param, the TA's private key and the policy $\mathbb{P}$ as input, outputs verifier credential strings $\{\mathbb{C}_{i,j,k}\}$ where i, j, k are indexes of credential strings. That is,

$$CreGen(\mathsf{param}, sk_{TA}, \mathbb{P}) \rightarrow \{\mathbb{C}_{i,j,k}\}.$$

Policy-controlled Signature Signing ($PSign$):

This is a probabilistic algorithm that, given the system parameter param, the trust authority's public key pk_{TA}, the signer's private key sk_S, the signer's public key pk_S, a message M and the policy $\mathbb{P}$ as input, outputs the signer's signature δ. That is,

$$PSign(\mathsf{param}, M, sk_S, pk_S, pk_{TA}, \mathbb{P}) \rightarrow \delta.$$

Policy-controlled Signature Verification ($PVerify$):

This is a deterministic algorithm that, given the system parameter param, the trust authority's public key pk_{TA}, the signer's public key pk_S, the policy $\mathbb{P}$, a set of credentials $\{\mathbb{C}_{i,j,k}\} \in \mathbb{B}$, a message M and a signature δ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$PVerify(\mathsf{param}, M, \delta, pk_{TA}, pk_S, \mathbb{P}, \{\mathbb{C}_{i,j,k}\}) \rightarrow \mathsf{d}.$$

5.2.2 Unforgeability

The unforgeability property in our model aims to provide security against existential unforgeability under an adaptive chosen message and credentials exposure attack. It intentionally prevents an attacker, who accesses to credential queries, to generate a policy-controlled signature δ^* on a new message M^*. Formally, this unforgeability provides assurance that someone with an access to the signing oracle, the credential oracle and the signer's public key pk_S, should be unable to produce a policy-controlled signature on a new message M^*, even if it arbitrarily chooses a policy $\mathbb{P}$, a message M and the entire credentials $\{\mathbb{C}\}$ as input.

We denote by *CM-A* the adaptively chosen message and credentials exposure. We also denote by *EUF-PCS* the existential unforgeable of a PCS scheme. Let $\mathcal{A}^{CM-A}_{EUF-PCS}$ be the adaptively chosen message and credentials exposure adversary and let $\mathcal{F}$ be a simulator. We denote that $\mathbb{B}^*$ is the entire possible set of credentials of a policy $\mathbb{P}^*$. Let *st* be the state of information that $\mathcal{A}$ obtains during the learning phase. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential unforgeability of a PCS scheme.

Let $\mathfrak{PS}$ be an algorithm that maintains the list of queried policy-controlled signatures and $\mathfrak{QS}$ be an algorithm that maintains the list of queried credentials. The oracles are provided in order to model the ability of adversaries to break the unforgeability of a PCS scheme as shown below.

$\mathcal{SSO}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a signature δ on its choice of a message M. As a response, $\mathcal{SSO}$ runs the *PSign* algorithm to generate a signature δ on a message M corresponding with pk_{TA}, pk_S and $\mathbb{P}$. $\mathcal{SSO}$ then returns δ, M to $\mathcal{A}$. After that, $\mathcal{SSO}$ keeps a record in the $\mathfrak{PS}$, which is $\mathfrak{PS} \leftarrow \mathfrak{PS}(\delta, M, \mathbb{P})$.

$\mathcal{VCO}$ oracle: At most q_{VC} times, $\mathcal{A}$ can make a query for the credential $\mathbb{C}_i$ corresponding to the assertion A_i in the policy $\mathbb{P}$. As a response, $\mathcal{VCO}$ replies to $\mathcal{A}$ with corresponding credentials $\{\mathbb{C}_{i,j,k}\}$. After that, $\mathcal{VCO}$ keeps a record in the $\mathfrak{QS}$, which is $\mathfrak{QS} \leftarrow \mathfrak{QS}(\{\mathbb{C}_{i,j,k}\})$.

$\mathcal{VSO}$ oracle: At most q_{VS} times, $\mathcal{A}$ can make a query for the verification of a signature δ to $\mathcal{VSO}$ with a signature δ as input. As a response, $\mathcal{VSO}$ returns with a decision d, which is *Accept* or *Reject* corresponding to a validation of signature δ.

We begin the experiment $\text{Expt}^{\mathcal{A}^{CMA}_{EUF\text{-}PCS}}(\ell)$ as follows: given a choice of messages M and access to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VCO}}$ and $\boldsymbol{\mathcal{VSO}}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles in an adaptive way. At the end of these queries, we assume that $\mathcal{A}$ outputs a forged signature δ^* on a new message M^* with respect to the public key pk_S and a policy $\mathbb{P}^*$. We denote that $\mathbb{B}^*$ is the entire possible set of credentials of a policy $\mathbb{P}^*$. We say that $\mathcal{A}$ wins the game if:

1. $Accept \leftarrow PVerify(M^*, \delta^*, pk_S, \mathbb{P}^*, \{\mathbb{C}_{i,j,k}\} \in \mathbb{B}^*)$.
2. $M^*, \mathbb{P}^* \notin \mathfrak{PS}$.

Let $Succ^{CM-A}_{EUF-PCS}(.)$ be the success probability of that $\mathcal{A}^{CM-A}_{EUF-PCS}$ wins the above game.

Definition 5.1 *We say that a PCS scheme is ($\mathtt{t}$,q_H,q_{SS},q_{VC},ϵ)-secure existential unforgeability under a chosen message and credentials exposure attack if there is no PPT adversary $\mathcal{A}^{CM-A}_{EUF-PCS}$ such that the success probability $Succ^{CM-A}_{EUF-PCS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-A}_{EUF-PCS}$ runs in time at most $\mathtt{t}$, makes at most q_H queries to the random oracle, and at most q_{SS} and q_{VC} queries to queries $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$, respectively.*

5.2.3 Coalition-resistance

In this section, we will discuss the coalition-resistance property of PCS schemes. This coalition-resistance property aims to prevent an attacker as a group of corrupted credential holders (verifiers) from verifying a policy-controlled signature δ^* on a message M^* with a policy $\mathbb{P}$ where the attacker does not have enough credentials to satisfy the policy $\mathbb{P}$. The condition "verifier does not have enough credentials to satisfy the policy $\mathbb{P}$" is elaborated as follows.
At least one set of credentials of assertions A in the policy $\mathbb{P} = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ is not given to the verifier, such that the verifier does not have sufficient credentials to verify a policy-controlled signature on a message M with the policy $\mathbb{P}$. For example:

1. In the case of $a = 1; a_i = 1; a_{i,j} > 1$, the verifier does not have one (or more) credential $\mathbb{C}_l$ of assertions $A_1, ..., A_{a_{i,j}}$.

2. In the case of $a = 1; a_i > 1; a_{i,j} > 1$, the verifier does not have one set of credential $\mathbb{C}_{l,1}, ..., \mathbb{C}_{l,a_i}$ of assertions $[A_{1,j,1}, ..., A_{1,j,a_{i,j}}]_{1 \leq j \leq a_i}$ that satisfies the first case.

3. In the case of $a > 1; a_i > 1; a_{i,j} > 1$, the verifier has one set of credential (e.g., $[\mathbb{C}_{l,j,1}, ..., \mathbb{C}_{l,j,a_{i,j}}]_{1 \leq j \leq a_i}$) of assertions $[A_{i,j,1}, ..., A_{i,j,a_{i,j}}]_{1 \leq i \leq a, 1 \leq j \leq a_i}$ that does not satisfy the second case.

Formally, the coalition-resistance property provides assurance that someone with access to the signing oracle, the credential oracle, and the signer's public parameter pk_S should be unable to distinguish a valid signature out of two policy-controlled signatures on a message M^* even by arbitrarily choosing a policy $\mathbb{P}^*$, a message M^* and the entire credentials $\{\mathbb{C}\}$ except one set of credentials that does not satisfy the policy $\mathbb{P}^*$ as input. This intentionally prevents a distinguisher from distinguishing a valid signature from a (simulated) invalid signature on any message M with a new policy $\mathbb{P}^*$.

Let *CRI-PCS* denote the existential coalition-resistance property of a PCS scheme. Let $\mathcal{A}_{CRI-PCS}^{CMP-A}$ be an adaptively chosen message and chosen policy attack and let $\mathcal{F}$ be a simulator. The experiment between $\mathcal{F}$ and $\mathcal{A}$ describes the existential coalition-resistance property of a PCS scheme as follows.

First, the oracles provided in order to model the abilities of adversary to break the coalition-resistance property of a PCS scheme are described in Section 5.2.2.

The experiment is divided into two phases. We run them as follows.

1. **Phase** 1**:** With any adaptive strategies, $\mathcal{A}$ arbitrarily sends a request for query to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles. The oracles respond as per their design.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{P}^* = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ such that:

   a. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issue a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle, however, $\mathcal{A}$ does not make sufficient requests for credentials to satisfy the policy $\mathbb{P}^*$, as mentioned clearly above.

   Next, $\mathcal{F}$ chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$ then, given a policy $\mathbb{P}^*$, a signer's public key pk_S and a message M^* as input, $\mathcal{F}$ makes a request for a

policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle and responds to $\mathcal{A}$ with δ^* as an output from the $\boldsymbol{\mathcal{SSO}}$ oracle. Otherwise, given a policy $\mathbb{P}^*$, a signer's public key pk_S, a message M^*, a valid policy-controlled signature δ^* on a message M^* with a policy $\mathbb{P}^*$ and a set of credentials $\{\mathbb{C}_{i,j,k}\} \in \mathbb{B}^*$ as input, $\mathcal{F}$ computes a (simulated) invalid policy-controlled signature δ^* and responds to $\mathcal{A}$ with δ^*.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it wants, on one condition, that $\mathcal{A}$ must have at least one set of challenges $M^*, \mathbb{P}^*, \delta^*$ such that

   a. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issue a request of credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle; however, $\mathcal{A}$ must not have enough credentials to satisfy the policy $\mathbb{P}^*$ and to verify δ^*, as mentioned clearly above.

4. **Guessing:** On the challenge $M^*, \mathbb{P}^*, \delta^*$, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ^{CMP-A}_{CRI-PCS}(.)$ be the success probability of $\mathcal{A}^{CMP-A}_{CRI-PCS}$ winning the above game.

Definition 5.2 *We say that PCS scheme is (*$\mathtt{t}$*,*q_H*,*q_{SS}*,*q_{VC}*,*ϵ*)-secure existential coalition-resistant under a chosen message and chosen policy attack if there is no PPT distinguisher* $\mathcal{A}^{CMP-A}_{CRI-PCS}$ *such that the success probability* $Succ^{CMP-A}_{CRI-PCS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}^{CMP-A}_{CRI-PCS}$ *runs in time at most* $\mathtt{t}$*, make at most* q_H *queries to the random oracle, and at most* q_{SS}*, and* q_{VC} *queries to the* $\boldsymbol{\mathcal{SSO}}$ *and* $\boldsymbol{\mathcal{VCO}}$ *oracles, respectively.*

5.2.4 Invisibility

In this section, we will elaborate the invisibility property of PCS schemes. Intuitionally, the invisibility property aims to prevent an attacker who does not have any credentials to satisfy a policy $\mathbb{P}$ from verifying a policy-controlled signature δ on a message M with respect to a policy $\mathbb{P}$. Formally, the invisibility property provides assurance that someone, with an access to the signing oracle, the verification oracle and the signer's public key pk_S, should be unable to distinguish a valid

policy-controlled signature on a message M^* from an invalid one, even if it arbitrary chooses a policy $\mathbb{P}^*$ and a message M^* as input.

Let $INV\text{-}PCS$ denote the existential invisibility privacy of a PCS scheme. Let $\mathcal{A}_{INV-PCS}^{CMP-A}$ be an adaptively chosen message and chosen policy distinguisher and let $\mathcal{F}$ be a simulator. The experiment between $\mathcal{F}$ and $\mathcal{A}$ describes as the existential invisibility privacy of a PCS scheme as follows.

First, the oracles provided in order to model the ability of adversaries to break the invisibility privacy of a PCS scheme are described in Section 5.2.2.

The experiment is divided into two phases. We run them as follows.

1. **Phase** 1**:** With any adaptive strategies, $\mathcal{A}$ arbitrarily sends query requests to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VSO}}$ oracles. The oracles respond as per their design.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{P}^* = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ such that, given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle. Next, $\mathcal{F}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$ then, given a policy $\mathbb{P}^*$ as input, a signer's public key pk_S and a message M^* as input, $\mathcal{F}$ makes a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle and responds to $\mathcal{A}$ with δ^* as an output from the $\boldsymbol{\mathcal{SSO}}$ oracle. Otherwise, given a policy $\mathbb{P}^*$, a signer's public key pk_S, a message M^* and a valid policy-controlled signature δ^* on message M^* with a policy $\mathbb{P}^*$ as input, $\mathcal{F}$ computes a (simulated) invalid policy-controlled signature δ^* and responds to $\mathcal{A}$ with δ^*.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it want. On one condition, that $\mathcal{A}$ must have at least one set of challenges $M^*, \mathbb{P}^*, \delta^*$ such that:

   a. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. Given $\mathbb{P}^*$, M^* and δ^* as input, $\mathcal{A}$ never issues a request for the verification of the policy-controlled signature δ^* to the $\boldsymbol{\mathcal{VSO}}$ oracle.

4. **Guessing:** On the challenge $M^*, \mathbb{P}^*, \delta^*$, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ_{INV-PCS}^{CMP-A}(.)$ be the success probability of $\mathcal{A}_{INV-PCS}^{CMP-A}$ winning the above game.

Definition 5.3 *We say that a PCS scheme is (*$\mathtt{t}$*,*q_H*,*q_{SS}*,*q_{VS}*,*ϵ*)-secure existential invisible under a chosen message and chosen policy attack if there is no PPT distinguisher* $\mathcal{A}_{INV-PCS}^{CMP-A}$ *such that the success probability* $Succ_{INV-PCS}^{CMP-A}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}_{INV-PCS}^{CMP-A}$ *runs in time at most* $\mathtt{t}$*, makes at most* q_H *queries to the random oracle, and at most* q_{SS}*, and* q_{VS} *queries to the* $\boldsymbol{\mathcal{SSO}}$ *and* $\boldsymbol{\mathcal{VSO}}$ *oracles, respectively.*

Theorem 5.1 *The invisibility of policy-controlled signature schemes implies the coalition-resistance property of policy-controlled signature schemes.*

Proof: Assuming that an invisibility adversary $\mathcal{A}_I$ solves the invisibility of a PCS scheme, we will show that an adversary $\mathcal{A}_{CR}$ can solve the coalition-resistance property of a PCS scheme by using $\mathcal{A}_I$. Let $\mathcal{S}$ be a simulator and then let $\mathcal{S}$ run the existential coalition-resistance game defined earlier with $\mathcal{A}_{CR}$. Meanwhile, $\mathcal{A}_{CR}$ runs the existential invisibility game defined earlier with $\mathcal{A}_I$. On accessing the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles constructed by $\mathcal{S}$, $\mathcal{A}_{CR}$ passes the $\boldsymbol{\mathcal{SSO}}$ oracle from $\mathcal{S}$ to $\mathcal{A}_I$. Then, by using the $\boldsymbol{\mathcal{VCO}}$ oracle from $\mathcal{S}$, $\mathcal{A}_{CR}$ constructs the $\boldsymbol{\mathcal{VSO}}$ oracle for $\mathcal{A}_I$ by making queries to the $\boldsymbol{\mathcal{VCO}}$ oracle for credentials to verify a signature queried by $\mathcal{A}_I$. At the end of phase 2, $\mathcal{A}_I$ outputs for a challenge with M^* and $\mathbb{P}^*$ to $\mathcal{A}_{CR}$. $\mathcal{A}_{CR}$ then relays this challenge to $\mathcal{S}$. $\mathcal{S}$ responds with δ^*. $\mathcal{A}_{CR}$ returns δ^* to $\mathcal{A}_I$. Finally, $\mathcal{A}_I$ outputs a decision b' and gives it to $\mathcal{A}_{CR}$. Then $\mathcal{A}_{CR}$ returns b' to $\mathcal{S}$.

From the above experiment, it is clearly shown that if $\mathcal{A}_I$ can solve the invisibility property of a PCS scheme, then $\mathcal{A}_{CR}$ can solve the coalition-resistance property of a PCS scheme via $\mathcal{A}_I$. Hence, the coalition-resistance property is a stronger model of the invisibility property in the notion of PCS scheme. □

5.3 The Proposed PCS Scheme

Prior to presenting our concrete construction of policy-controlled signature schemes, we will first describe the idea and intuition behind our construction as clarification. Intuitively, we can achieve a policy-controlled signature scheme by combining the idea of policy-based encryption schemes [BM05], a general signature scheme and a designated verifier signature scheme. First, we combine a designated verifier signature on a message into a policy-based ciphertext such that only a verifier who has satisfied the policy can verify the authenticity of the signature. This will constitute the part of policy-controlled signatures that we refer to as "the encrypted designated

verifier signature". Then, a signature scheme is used to sign the concatenation of the policy and the encrypted designated verifier signature. The encrypted designated verifier signature and the signature from the above construction constitute a policy-controlled signature. The purpose of the above construction is to ensure the authentication of the signer such that the verifier is convinced that the signer has actually generated this policy-controlled signature. The signature can be publicly verifiable; however, one cannot be convinced that the signature is indeed associated with a message unless one has the credentials satisfying the policy to verify the encrypted designated verifier signature. Hence, without revealing the verifier's private information (the credentials associated with the policy), the verifier should not be able to convince another party that a signer generated the policy-controlled signature.

5.3.1 The General Construction

In this section, we present our concrete construction of PCS schemes. Let $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be three distinct random one-way functions that map any string to group $\mathbb{G}_1$ and let $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. We denote by $\mathbb{G}_1$ and $\mathbb{G}_T$ groups of prime order p. Assume that there exists an efficient computationally bilinear mapping function $\hat{e}$, which maps $\mathbb{G}_1$ to $\mathbb{G}_T$. The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The scheme is described as follows.

Setup: Given a security parameter ℓ as input, a trusted third party randomly chooses a prime $p \approx poly(1^\ell)$. Select a random generator $g \in \mathbb{G}_1$ and a bilinear mapping function $\hat{e}$. Select hash functions $H_0(.), H_1(.), H_2(.), h(.)$. We denote by $\mathsf{param} = (p, \hat{e}, g, H_0, H_1, H_2, h)$ the system parameter. Then, *Setup* returns param.

TKeyGen: Given a system parameter param as input, a trusted authority TA randomly generates a private key sk_{TA} and a public key pk_{TA} as follows: select two random integers $\mu, \gamma \in \mathbb{Z}_p$. Let $U = g^\mu; W = g^\gamma$ denote a public key. Therefore, *TKeyGen* returns $sk_{TA} = (\mu, \gamma)$ as a private key of the trusted authority and $pk_{TA} = (U, W)$ as a public key of the trusted authority.

SKeyGen: Given a system parameter param and a public key of the trusted authority pk_{TA} as input, a signer S randomly generates a private key sk_S and a public

key pk_S as follows: select a random integer $x \in \mathbb{Z}_p$. Let $X = g^x; \hat{X} = W^x$ denote a public key. Therefore, $SKeyGen$ returns $sk_S = x$ as a private key of the signer and $pk_S = (X, \hat{X})$ as a public key of the signer.

CreGen: Let P be a statement in the policy, e.g., $P =$'CIA agent'. An assertion A of P is computed as follows: $A = H_2(P)$. Given a system parameter param as input, the trusted authority's public key pk_{TA}, the trusted authority's private key sk_{TA} and a set of assertions $A_1, ..., A_n$ that the verifier is allowed to obtain, a trusted authority TA randomly generates each verifier's credential string $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$ where i is an index of credentials as follows. TA randomly selects $\nu_i \in \mathbb{Z}_p^*$ and computes each credential $\mathbb{V}_i = U^{1/\nu_i}$; $\mathbb{R}_i = g^{(\mu\gamma)/\nu_i} A_i^{\mu}$; $\mathbb{G}_i = g^{\nu_i}$ and then returns $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$ to the verifier as a credential of assertion A_i. The verifier checks the validity of $\mathbb{C}_i$ as follows.

$$\begin{aligned} \hat{e}(\mathbb{R}_i, g) &\stackrel{?}{=} \hat{e}(A_i, U)\hat{e}(W, \mathbb{V}_i), \\ \hat{e}(\mathbb{V}_i, \mathbb{G}_i) &\stackrel{?}{=} \hat{e}(U, g). \end{aligned}$$

PSign: Given param, pk_{TA}, sk_S, pk_S, $\mathbb{P} = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ and a message M, S computes a policy-controlled signature δ on a message M as follows.

$$\begin{aligned} r, t_1, ..., t_a &\stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ t = \oplus_{i=1}^{a} t_i,\ \boldsymbol{\delta_1} = g^r, \\ \boldsymbol{\delta_2} &= X^r,\ \boldsymbol{\delta_3} = \hat{X}^r, \\ \Psi &= M||\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}, \end{aligned}$$

for $i = 1$ to a, for $j = 1$ to a_i:

$$R_{i,j} = t_i \oplus h(H_0(\Psi)||i||j||\hat{e}((\prod_{k=1}^{a_{i,j}} A_{i,j,k})^{r \cdot x}, U)).$$

Then compute

$$\begin{aligned} \Omega &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1 \le i \le a}, \\ \boldsymbol{\delta_4} &= H_1(\Omega)^x. \end{aligned}$$

The policy-controlled signature on a message M is
$\delta = (H_0(\Psi), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, [R_{i,1}, ..., R_{i,a_i}]_{1 \le i \le a})$.

PVerify: Let $\{\mathbb{C}_{i,j,k}\} = \mathbb{C}_{1,j,1}, ..., \mathbb{C}_{a,j,a_{i,j}}$ be a set of credentials in $\mathbb{B}$ that the verifier possesses. Given pk_S, pk_{TA}, pk_V, $\{\mathbb{C}_{i,j,k}\} \subset \mathbb{B}$, $\mathbb{P}$, δ and a message M, a verifier V first checks whether

$$\hat{e}(\boldsymbol{\delta_2}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_1}, X),\ \hat{e}(\boldsymbol{\delta_3}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, W)$$

holds or not. If not, then V outputs *Reject*. Otherwise, V computes as follows: for $i = 1$ to a:

$$\hat{t}_i = R_{i,j} \oplus h(H_0(\Psi)||i||j||(\prod_{k=1}^{a_{i,j}}(\hat{e}(\mathbb{R}_{i,j,k}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j,k}, \boldsymbol{\delta_3})^{-1}))).$$

Next, compute

$$\hat{t} = \oplus_{i=1}^{a}\hat{t}_i,\ \Lambda = M||\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\hat{t}||\hat{t}_1||...||\hat{t}_a||pk_S||pk_{TA}||\mathbb{P}.$$

Then, V checks whether $H_0(\Lambda) \stackrel{?}{=} H_0(\Psi)$, $\hat{e}(\boldsymbol{\delta_4}, g) \stackrel{?}{=} \hat{e}(H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\hat{t}||\hat{t}_1||...||\hat{t}_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1\leq i\leq a}), X)$ holds or not. If not, then it outputs *Reject*. Otherwise, it outputs *Accept*.

5.4 Security Analysis

5.4.1 Unforgeability

Theorem 5.2 *Our policy-controlled signature scheme is existential unforgeable under an adaptive chosen message and credentials exposure attack if the CDH assumption holds in the random oracle model.*

Proof: Suppose that there exists a forger $\mathcal{A}$, which runs the existential unforgeability game defined in Section 5.2.2, then we will show there exists an adversary $\mathcal{F}$ that solves the CDH problem by using $\mathcal{A}$. Intuitively, the proof begins with the construction of oracles defined in Section 5.2.2. We then construct a simulator $\mathcal{F}$ and run the existential unforgeability game defined in Section 5.2.2 with a forgery $\mathcal{A}$. Finally, we analyse the success probability of the existential unforgeability game under an adaptive chosen message and credentials exposure attack and show that this success probability is reducible to the CDH problem in the random oracle model.

We begin with the construction of oracles and run the existential unforgeability game as follows. Given g, g^a and g^b as an instance of the CDH problem, $\mathcal{F}$ sets g^b as one of the answers for the hash query to the random oracle. Next, $\mathcal{F}$ chooses

a random integer $\mu, \gamma \in \mathbb{Z}_p$ and sets $U = g^{\mu}; W = g^{\gamma}$ as a public key of TA. Then, $\mathcal{F}$ sets $X = g^{a}; \hat{X} = X^{\gamma}$ as the signer's public key pk_S. We note that the algorithms managing the lists of each query are simple to construct, and therefore such algorithms will be omitted. From the above setting, it is easy for $\mathcal{F}$ to construct the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VCO}}$ and the random oracle $\boldsymbol{\mathcal{HO}}$ as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: Let M be a message that is an input for the hash value to the $\boldsymbol{\mathcal{HO}}$ oracle. If it is a request for a hash value of $H_1(M)$, $\boldsymbol{\mathcal{HO}}$ selects $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d} = 1$ is $\frac{1}{q_H}$. If $\mathsf{d} = 1$ then set $H_1(M) = g^{b}$ and return $H_1(M)$. Otherwise, $\bar{l} \xleftarrow{\$} \mathbb{Z}_p$; $H_1(M) = g^{\bar{l}}$ and return $H_1(M)$. For $H_0(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_1 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_0(M) = g^{l_1}$. For $H_2(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_2 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_2(M) = g^{l_2}$. For $h(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_3 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(M) = l_3$. Then $\boldsymbol{\mathcal{HO}}$ keeps $\bar{l}, l_1, l_2, l_3$ in the list and this list can be accessed only by $\mathcal{F}$. Note that $\boldsymbol{\mathcal{HO}}$ manages the duplicated hash value of the list by repeating the process such that the output of $\boldsymbol{\mathcal{HO}}$ behaves like a result from the random oracle.

$\boldsymbol{\mathcal{VCO}}$ oracle: $\boldsymbol{\mathcal{VCO}}$ runs $CreGen$ to generate the credential $\mathbb{C}_i$ of assertion A_i and then returns $\mathbb{C}_i$

$\boldsymbol{\mathcal{SSO}}$ oracle: Given $\mathbb{P} = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ and message M as input, $\boldsymbol{\mathcal{SSO}}$ computes a policy-controlled signature as follows.

$$\begin{aligned} r, t_1, ..., t_a &\xleftarrow{\$} \mathbb{Z}_p,\ t = \oplus_{i=1}^{a} t_i,\ \boldsymbol{\delta_1} = g^{r},\ \boldsymbol{\delta_2} = X^{r},\ \boldsymbol{\delta_3} = \hat{X}^{r}, \\ \Psi &= M||\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}, \end{aligned}$$

for $i = 1$ to a, for $j = 1$ to a_i:

$$R_{i,j} = \hat{t}_i \oplus h(H_0(\Psi)||i||j||(\prod_{k=1}^{a_{i,j}}(\hat{e}(\mathbb{R}_{i,j,k}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j,k}, \boldsymbol{\delta_3})^{-1}))).$$

Then, having access to the list of $\bar{l}, l_1, l_2, l_3$, $\mathcal{F}$ checks whether $H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\hat{t}||\hat{t}_1||...||\hat{t}_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1\le i\le a}) \stackrel{?}{=} g^{b})$, if not, then $\mathcal{F}$ gives $\bar{l}$ to $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{SSO}}$ computes $\boldsymbol{\delta_4} = X^{\bar{l}}$. Otherwise, output $\perp$. Finally, $\boldsymbol{\mathcal{SSO}}$ returns a policy-controlled signature on message M, which is $\delta = (H_0(\Psi), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, [R_{i,1}, ..., R_{i,a_i}]_{1\le i\le a})$.

$\mathcal{A}$ is given an access to these oracles. Assume that a hash of a string or a message from the random oracle $\boldsymbol{\mathcal{HO}}$ is always queried before $\mathcal{A}$ makes a query to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, or before it outputs a potential forgery, denoted by $M^*, \delta^*, \mathbb{P}^*$.

After executing an adaptive strategy with these oracles, $\mathcal{A}$ outputs a forgery δ^* on a message M^* with respect to $\mathbb{P}^*$. $\mathcal{A}$ wins the game if a policy-controlled signature δ^* on message M^* is valid and is not an output from the $\boldsymbol{\mathcal{SSO}}$ oracle.

Let $Succ^{CM-A}_{EUF-PCS} = \epsilon$ be the probability that $\mathcal{A}$ wins the game. We denote by e the base of the natural logarithm and let $q \geq q_H$ be a polynomial upper bound on the number of queries that $\mathcal{A}$ makes to the $\boldsymbol{\mathcal{HO}}$ oracle. As mentioned above, $\mathcal{A}$ always make a query request to $\boldsymbol{\mathcal{HO}}$ before making any requests to the $\boldsymbol{\mathcal{SSO}}$ oracle; hence, $q_H \geq q_{SS}$. Therefore, we can analyse the success probability that $\mathcal{A}$ outputs a signature δ^* on message M^*, where $\boldsymbol{\delta^*_4} = H_1(\Omega)^x = (g^b)^x$, and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{SSO}}$ oracles.* The probability of this event is $(1 - \frac{1}{q_H})^{q_{SS}}$. The fact is that $\mathcal{A}$ needs to reserve at least one request for a hash value to output $\boldsymbol{\delta^*_4}$, which is part of the forgery. Therefore, the upper bound for the $\boldsymbol{\mathcal{SSO}}$ oracle is $q_H - 1$ and then the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

- E_2: *$\mathcal{F}$ does not abort after $\mathcal{A}$ output δ^*.* $\mathcal{A}$ needs to reserve at least one request for a hash value to output $\boldsymbol{\delta^*_4}$, which is a part of forgery. However, if

$$H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1 \leq i \leq a})$$

$$= H_1(\Omega) \neq g^b$$

  for $\boldsymbol{\delta^*_4}$, then $\mathcal{F}$ aborts the simulation. Hence, the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

The probability that $\mathcal{A}$ wins the above game and outputs a signature δ^* on a message M^*, where $\boldsymbol{\delta^*_4} = H_1(\Omega)^x = (g^b)^x$, is $\Pr[Succ^{CM-A}_{EUF-PCS}] \cdot \Pr[Succ^{CM-A}_{EUF-PCS} | E_1 | E_2] \geq \epsilon(\frac{q_H}{e \cdot (q_H - 1)})^2$. From the above outputs by $\mathcal{A}$, $\mathcal{F}$ obtains $\boldsymbol{\delta^*_4} = H_1(\Omega)^x$. Since $H_1(\Omega) = g^b$ and $x = a$, $\mathcal{F}$ returns $\boldsymbol{\delta^*_4} = g^{ab}$ as an output for the CDH problem with non-negligible probability as mentioned above. □

5.4.2 Coalition-resistance

Theorem 5.3 *In the random oracle model, the proposed policy-controlled signature scheme is existential coalition-resistant against adaptively chosen message and chosen policy attack $\mathcal{A}^{CMP-A}_{CRI-PCS}$ attack if the DBDH assumption is hold.*

Proof: Suppose that there exists a forger $\mathcal{A}$, which runs the existential coalition-resistance game defined in Section 5.2.3, then we will show that there exists an adversary $\mathcal{F}$ that answers the DBDH problem by using $\mathcal{A}$ as a tool. We shall start with the construction oracles as they are designed in Section 5.2.3. Then we construct $\mathcal{F}$ and run it over $\mathcal{A}$ with the existential coalition-resistance game defined in Section 5.2.3. Next, we summarise the success probability of the existential coalition-resistance game under an adaptive chosen message and chosen policy attack. Finally, from the existential coalition-resistance game and its success probability, we can draw the conclusion that the success probability of solving the DBDH problem is non-negligible if the success probability of the above game is non-negligible.

We construct the oracles and run the existential unforgeability game as follows. Given g, g^a, g^b, g^c and Z as an instance of the DBDH problem, $\mathcal{F}$ sets g^a as one of the answers for the hash query to the random oracle. Next, $\mathcal{F}$ randomly selects x, γ and sets $U = g^b; W = g^\gamma$ as the TA public key. Then, $\mathcal{F}$ sets $X = g^x; \hat{X} = g^{x \cdot \gamma}$ as the signer's public key. Assume that there are algorithms managing the lists of each query and that such algorithms are easy to construct and will be omitted. From the above setting, it is easy for $\mathcal{F}$ to construct the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VCO}}$ and the random oracle $\boldsymbol{\mathcal{HO}}$ as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: Let M be a message that is an input for the hash value to the $\boldsymbol{\mathcal{HO}}$ oracle. If it is a request for a hash value of $H_2(M)$, $\boldsymbol{\mathcal{HO}}$ selects $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d} = 1$ is $\frac{1}{q_H}$. If $\mathsf{d} = 1$ then set $H_2(M) = g^a$ and return $H_2(M)$. Otherwise, $\bar{l} \xleftarrow{\$} \mathbb{Z}_p$; $H_1(M) = g^{\bar{l}}$ and return $H_2(M)$. For $H_0(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_1 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_0(M) = g^{l_1}$. For $H_1(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_2 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_2(M) = g^{l_2}$. For $h(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_3 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(M) = l_3$. Then $\boldsymbol{\mathcal{HO}}$ keeps $\bar{l}, l_1, l_2, l_3$ in a list and this list can be accessed only by $\mathcal{F}$. Note that $\boldsymbol{\mathcal{HO}}$ manages the duplicated hash value of the list by repeating the process such that the output of $\boldsymbol{\mathcal{HO}}$ behaves like a result from the random oracle.

$\boldsymbol{\mathcal{VCO}}$ oracle: Given A_i as input, $\boldsymbol{\mathcal{VCO}}$ accesses to the $\boldsymbol{\mathcal{HO}}$ oracle for a matching pair of $P_i, A_i = H_2(P_i)$; if $\boldsymbol{\mathcal{HO}}$ returns $A_i = H_2(P_i) = g^a$ then output $\perp$. Otherwise, $\mathcal{F}$ accesses the list in $\boldsymbol{\mathcal{HO}}$ and returns $\bar{l_i} : A_i = g^{\bar{l_i}}$ to $\boldsymbol{\mathcal{VCO}}$. Then $\boldsymbol{\mathcal{VCO}}$ selects a random integer $\nu \in \mathbb{Z}_p$ and computes $\mathbb{V}_i = U^{1/\nu_i}$; $\mathbb{R}_i = U^{\gamma/\nu_i} U^{\bar{l_i}}$; $\mathbb{G}_i = g^{\nu_i}$ as a credential of A_i. $\boldsymbol{\mathcal{VCO}}$ returns $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$.

$\mathcal{SSO}$ oracle: Given $\mathbb{P} = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$ and a message M as input, $\mathcal{SSO}$ computes a policy-controlled signature as follows.

$$\begin{aligned} r, t_1, ..., t_a &\xleftarrow{\$} \mathbb{Z}_p,\ t = \oplus_{i=1}^{a} t_i,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = X^r,\ \boldsymbol{\delta_3} = \hat{X}^r, \\ \Psi &= M||\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}, \end{aligned}$$

for $i = 1$ to a, for $j = 1$ to a_i:

$$R_{i,j} = t_i \oplus h(H_0(\Psi)||i||j||((\prod_{k=1}^{a_{i,j}} A_{i,j,k})^{r \cdot x}, U)).$$

Next, $\mathcal{SSO}$ computes

$$\boldsymbol{\delta_4} = H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1 \le i \le a})^x.$$

Then $\mathcal{SSO}$ returns a policy-controlled signature on message M, which is $\delta = (H_0(\Psi), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, [R_{i,1}, ..., R_{i,a_i}]_{1 \le i \le a})$.

Access to these oracles is given to $\mathcal{A}$. Let us presume that a hash of message M from the $\mathcal{HO}$ oracle is always queried before $\mathcal{A}$ makes a query request to the $\mathcal{SSO}$ and $\mathcal{VCO}$ oracles, or before it outputs a decision bit b. Now, we run an experiment as defined in Section 5.2.3 as follows.

1. **Phase** 1**:** $\mathcal{A}$ arbitrarily sends queries to the $\mathcal{SSO}$ and $\mathcal{VCO}$ oracles. The oracles respond as above.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{P}^* = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}} A_{i,j,k}]]$. $\mathcal{F}$ aborts the game if

   1. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ issued a request for a policy-controlled signature to the $\mathcal{SSO}$ oracle.

   2. $\mathcal{A}$ does not have sufficient credentials to satisfy the policy $\mathbb{P}^*$.

   Next, $\mathcal{F}$ computes a response as follows.

$$\begin{aligned} r, t_1, ..., t_a &\xleftarrow{\$} \mathbb{Z}_p,\ t = \oplus_{i=1}^{a} t_i,\ \boldsymbol{\delta^*_1} = g^{c \cdot r},\ \boldsymbol{\delta^*_2} = g^{c \cdot x \cdot r},\ \boldsymbol{\delta^*_3} = g^{c \cdot \gamma \cdot x \cdot r}, \\ \Psi &= M^*||\boldsymbol{\delta^*_1}||\boldsymbol{\delta^*_2}||\boldsymbol{\delta^*_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}, \end{aligned}$$

   For $i = 1$ to a, for $j = 1$ to a_i: if $A_{i^*,j^*,k^*} = g^a$ then compute as follows.

$$R_{i,j} = t_i \oplus h(H_0(\Psi)||i||j||\hat{e}((\prod_{k=1, k \neq k^*}^{a_{i,j}} g^{c \cdot \bar{l}_{i,j,k}})^{r \cdot x}, U) Z^{r \cdot x}).$$

Otherwise, compute as follows.

$$R_{i,j} \quad = \quad t_i \oplus h(H_0(\Psi)||i||j||\hat{e}((\prod_{k=1}^{a_{i,j}} g^{c\cdot\bar{l}_{i,j,k}})^{r\cdot x}, U)).$$

Then $\mathcal{F}$ computes

$$\boldsymbol{\delta^*_4} = H_1(\boldsymbol{\delta^*_1}||\boldsymbol{\delta^*_2}||\boldsymbol{\delta^*_3}||t||t_1||...||t_a||pk_S||pk_{TA}||\mathbb{P}||[R_{i,1}||...||R_{i,a_i}]_{1\leq i\leq a})^x.$$

Next, $\mathcal{F}$ returns a policy-controlled signature on a message M, which is $\delta^* = (H_0(\Psi), \boldsymbol{\delta^*_1}, \boldsymbol{\delta^*_2}, \boldsymbol{\delta^*_3}, \boldsymbol{\delta^*_4}, [R_{i,1}, ..., R_{i,a_i}]_{1\leq i\leq a})$.

3. **Phase 2:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it requests, on one condition, that $\mathcal{A}$ must have at least one set of challenges $M^*, \mathbb{P}^*, \delta^*$ such that

   a. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issued a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle; however, $\mathcal{A}$ does not have sufficient credentials to satisfy the policy $\mathbb{P}^*$.

4. **Guessing:** On the challenge $M^*, \mathbb{P}^*$, $\mathcal{A}$ finally outputs a guess b'.

From the above experiment, we can solve the DBDH problem when $\mathcal{A}$ wins or aborts the experiment with the condition that $\mathcal{A}$ picks the challenge $M^*, \mathbb{P}^*, \delta^*$, which is inserted with Z and g^c, and a credential for $A_{i,j,k^*} = g^a$ is never queried. Since $A_{i,j,k^*} = g^a$ contained in $\alpha_{i,j}$, $\mathcal{A}$ can use it to check whether

$$\begin{aligned}
\hat{e}((\prod_{k=1,k\neq k^*}^{a_{i,j}} g^{c\cdot\bar{l}_{i,j,k}})^{r\cdot x}, U)Z^{r\cdot x} &\stackrel{?}{=} \prod_{k=1}^{a_{i,j}}(\hat{e}(\mathbb{R}_{i,j,k}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j,k}, \boldsymbol{\delta_3})^{-1}) \\
\hat{e}((\prod_{k=1,k\neq k^*}^{a_{i,j}} g^{c\cdot\bar{l}_{i,j,k}})^{r\cdot x}, U)Z^{r\cdot x} &\stackrel{?}{=} \prod_{k=1,k\neq k^*}^{a_{i,j}}(\hat{e}(\mathbb{R}_{i,j,k}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j,k}, \boldsymbol{\delta_3})^{-1}) \cdot \\
&\quad \hat{e}(\mathbb{R}_{i,j,k^*}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j,k^*}, \boldsymbol{\delta_3})^{-1} \\
Z^{r\cdot x} &\stackrel{?}{=} \hat{e}(\mathbb{R}_{i,j,k^*}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j,k^*}, \boldsymbol{\delta_3})^{-1} \\
Z^{r\cdot x} &\stackrel{?}{=} \hat{e}(U^{\gamma/\nu_{i,j,k^*}} A^b_{i,j,k^*}, g^{c\cdot x\cdot r}) \cdot \hat{e}(U^{1/\nu_{i,j,k^*}}, g^{c\cdot\gamma\cdot x\cdot r})^{-1} \\
Z^{r\cdot x} &\stackrel{?}{=} \hat{e}(A^b_{i,j,k^*}, g^{c\cdot x\cdot r}) \\
Z^{r\cdot x} &\stackrel{?}{=} \hat{e}((g^a)^b, g^{c\cdot x\cdot r}) \\
Z^{r\cdot x} &\stackrel{?}{=} \hat{e}(g, g)^{a\cdot b\cdot c\cdot x\cdot r}
\end{aligned}$$

holds or not. $\mathcal{A}$ will not abort the game if the above holds.

Let $\mathcal{A}$ win the game with an advantage $Succ^{CM-A}_{CRI-PCS} = \epsilon$. We denote by e the base of the natural logarithm and let $q \geq q_H$ be a polynomial upper bound of queries that $\mathcal{A}$ makes to the $\boldsymbol{\mathcal{HO}}$ oracle. Note that $q << p$. As mentioned above, $\mathcal{A}$ always makes a query request to $\boldsymbol{\mathcal{HO}}$ before it makes any requests to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles; hence, $q_H \geq q_{VC}$ and $q_H \geq q_{SS}$. Therefore, we can analyse the probability that $\mathcal{A}$'s guess is correct and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{VCO}}$ oracles.* The probability of this event is $(1 - \frac{1}{q_H})^{q_{VC}}$. The fact is that $\mathcal{A}$ needs to reserve at least one request for a credential of $A^*_{i,j,k}$ and one request for a hash value of $A^*_{i,j,k} = H_2(P^*_{i,j,k})$, which is part of the policy $\mathbb{P}^*$. Therefore, the upper bound for the $\boldsymbol{\mathcal{VCO}}$ oracle is $q_H - 1$ and then the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

- E_2: *$\mathcal{F}$ does not abort after* Phase 1 *and* Phase 2. Since we have assumed that $\mathcal{A}$ follows the experiment and outputs a guess with a valid challenge $(M^*, \mathbb{P}^*, \delta^*)$, then the probability of this event is 1.

The probability that $\mathcal{A}$ wins the above game and it outputs a correct guess $b' = b$ is $\Pr[Succ^{CM-A}_{CRI-PCS}] \cdot \Pr[Succ^{CM-A}_{CRI-PCS} | E_1 | E_2] \geq \epsilon \frac{q_H}{e \cdot (q_H - 1)}$.

Let ϵ' be an advantage in solving the DBDH problem. From the above game, $\mathcal{F}$ outputs a guess for the DBDH problem with $\mathcal{A}$'s guess. However, we note that $\mathcal{A}$ can choose a challenge policy $\mathbb{P}$. Hence, $\mathcal{A}$ can try to guess from one of the credentials that $\mathcal{A}$ does not make a request for a credential to the $\boldsymbol{\mathcal{VCO}}$ oracle. Thus, there is a event where $\mathcal{A}$'s guess is not a correct guess for the DBDH problem, which is when A_{i^*,j^*,k^*} is not chosen by $\mathcal{A}$. The probability for this event is $\frac{1}{a \cdot a_i}$. Therefore, the advantage that $\mathcal{F}$ can output a correct guess for the DBDH problem by using $\mathcal{A}$ is $\epsilon' \geq \frac{1}{a \cdot a_i} \epsilon \frac{q_H}{e \cdot (q_H - 1)}$. Hence, the probability that $\mathcal{A}$ breaks the existential coalition-resistance property of a PCS scheme against adaptively chosen message and chosen policy attack is $\epsilon \leq \epsilon' a \cdot a_i \cdot e \cdot (q_H - 1)/q_H$. Since $a \cdot a_i \leq q_H << q$, the analysis of the above advantages shows that the success of breaking the existential coalition-resistance property of a PCS scheme is non-negligible if the probability of breaking the DBDH problem is non-negligible. □

5.5 Definition of Universal Policy-controlled Signature Scheme (UPCS)

In this section, we give a definition of universal policy-controlled signature (UPCS) schemes that allow a policy signer (as a signature holder) using a policy to limit the verification of the signature generated by a signer. In other words, only a verifier that satisfies the policy specified by the policy signer can verify the policy-controlled signature that generated from a signature signed by a signer. We mentioned the practical scenario of universal policy-controlled signature schemes in Section 5.1. We provide an outline of our UPCS scheme as follows.

5.5.1 Outline of UPCS

Let TA denote a trusted authority that issues credentials associated with policies. Let CA denote a certificate authority that generates system parameter and certifies public keys for all parties. There are three major players in a universal policy-controlled signature scheme, namely a signer(s), a policy signer (who also acts as signature holder) and a verifier. The role of the signer S is to generate an ordinary signature for a policy signer. The policy signer is a party that generates a policy-controlled signature. This (universal) policy-controlled signature is used to prove that (1) a signature holder holds an ordinary signature generated by the signer, and (2) the policy-controlled signature is indeed generated by the signature holder. In addition, *only* a verifier V, who holds a credential satisfying the policy directed by a signature holder, can verify the (universal) policy-controlled signature. Credentials held by a verifier V are issued by TA.

We denote by A an assertion issued by TA. Each assertion A may be a hash value of certain statements, such as "Priority Club Platinum Member". Without loss of generality, we redefine $\mathbb{P}$ for UPCS to be a policy only in the disjunctive normal form (DNF), which contains a set of assertions $\mathbb{P} = \bigvee_{i=1}^{a}[\bigwedge_{j=1}^{a_i} A_{i,j}]$ where i, j are indexes. Let $\mathbb{C}_{i,j}$ denote a credential for an assertion and let $\mathbb{B} = [\{\mathbb{C}_{1,1}, ..., \mathbb{C}_{a,1},\}, ..., \{\mathbb{C}_{1,a_i}, ..., \mathbb{C}_{a,a_i}\}]$ denote a set of the entire possible set of credentials that satisfy the policy $\mathbb{P}$ where i, j are indexes for assertions associated with the credentials. For simplicity, let $\{\mathbb{C}_i\} = \mathbb{C}_{i,1}, ..., \mathbb{C}_{i,a_i}$ denote a set of credentials, which may or may not be a set of credentials in $\mathbb{B}$. Let $\{\mathbb{C}\} = \mathbb{C}_{1,1}, ..., \mathbb{C}_{a,a_i}$ be the entire credentials, where i is an index. Let us assume that all parties will comply with the registration

protocol with a certificate authority CA to obtain a certificate on their respective public keys.

A universal policy-controlled signature scheme Σ is a 8-tuple ($Setup$, $TKeyGen$, $SKeyGen$, $PKeyGen$, $CreGen$, $Sign$, $Verify$, $PSign$,$PVerify$). The definition of $Setup$, $TKeyGen$, $SKeyGen$ and $CreGen$ is same as the definition in Section 5.2. The definition of $PKeyGen$, $Sign$, $Verify$, $PSign$ and $PVerify$ is described as follows.

Policy Signer Key Generator ($PKeyGen$):

This is a probabilistic algorithm that, given the system parameter param and the public key of the trusted authority pk_{TA} as input, outputs strings (sk_P, pk_P) where they denote the private key and the public key of a signer, respectively. That is,

$$PKeyGen(\mathsf{param}, pk_{TA}) \rightarrow (pk_P, sk_P).$$

Signature Signing ($Sign$):

This is a probabilistic algorithm that, given the system parameter param, the signer's private key sk_S, the signer's public key pk_S and a message M as input, outputs a signer's signature σ. That is,

$$Sign(\mathsf{param}, M, sk_S, pk_S) \rightarrow \sigma.$$

Signature Verification ($Verify$):

This is a deterministic algorithm that, given the system parameter param, the signer's public key pk_S, a message M and a signature σ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(\mathsf{param}, M, \sigma, pk_S) \rightarrow \mathsf{d}.$$

Policy-controlled Signature Signing ($PSign$):

This is a probabilistic algorithm that, given the system parameter param, the trust authority's public key pk_{TA}, the signer's public key pk_S, the policy signer's private key sk_P, the policy signer's public key pk_P, a signer's signature σ on a message M and the policy $\mathbb{P}$ as input, outputs a universal policy-controlled signature δ. That is,

$$PSign(\mathsf{param}, M, \sigma, sk_P, pk_S, pk_P, pk_{TA}, \mathbb{P}) \rightarrow \delta.$$

Policy-controlled Signature Verification ($PVerify$):

This is a deterministic algorithm that, given the system parameter param, the trust authority's public key pk_{TA}, the signer's public key pk_S, the policy signer's public key pk_P, the policy $\mathbb{P}$, a set of credentials $\{\mathbb{C}_{i,j,k}\} \in \mathbb{B}$, a message M and a universal policy-controlled signature δ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(\mathsf{param}, M, \delta, pk_{TA}, pk_S, pk_P, \mathbb{P}, \{\mathbb{C}_{i,j,k}\}) \rightarrow \mathsf{d}.$$

5.5.2 Unforgeability

The unforgeability property of UPCS is divided into two parts. The first model aims to ensure security against existential unforgeability under adaptive chosen message, signer's private key exposure and credentials exposure attack. It intentionally prevents an attacker, who accesses the credential oracle and the signer's private key, from generating a policy-controlled signature σ_* on a new message M^*. The second model aims to provide security against existential unforgeability under adaptive chosen message, chosen policy signer's private key exposure and credentials exposure attack. The second model intentionally prevents an attacker, who accesses to the credential oracle and the policy signer's private key, from generating a policy-controlled signature σ_* on a new message M^*. The purpose of the above models is to provide a fair unforgeability property of UPCS for both the original signer and the policy signer.

First, the oracles are provided in order to model the ability of adversaries to break the unforgeability of a UPCS scheme as described below. Let $\mathfrak{SL}$, $\mathfrak{PL}$, $\mathfrak{QR}$, $\mathfrak{SS}$, $\mathfrak{PS}$ and $\mathfrak{QS}$ be algorithms that maintain the list of the signer's public-private key pair, the list of the policy signer's public-private key pairs, the list of queried public-private key pairs, the list of queried signatures, the list of queried policy-controlled signatures and the list of queried credentials, respectively.

SPO* oracle:** At most q_{SP} times, $\mathcal{A}$ can make a query for a new public key of the signer to ***SPO. As a response, given the security parameter param, ***SPO*** runs the $SKeyGen$ algorithm to generate a public-private key of the signer (pk_S, pk_S). ***SPO*** then returns pk_S to $\mathcal{A}$. After that, ***SPO*** keeps a record in the $\mathfrak{SL}$, which is $\mathfrak{SL} \leftarrow \mathfrak{SL}(pk_S, sk_S)$.

$\mathcal{SKO}$ oracle: At most q_{SK} times, $\mathcal{A}$ can make a query for the signer's private key sk_S of the chosen the signer's public key pk_S to $\mathcal{SKO}$. As a response, $\mathcal{SKO}$ matches the signer's public key pk_S in the list $\mathfrak{SL}$ to obtain the signer's private key sk_S Then, $\mathcal{SKO}$ returns sk_S to $\mathcal{A}$. After that, $\mathcal{SKO}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_S, sk_S)$.

$\mathcal{PPO}$ oracle: At most q_{PP} times, $\mathcal{A}$ can make a query for a new public key of the signer to $\mathcal{PPO}$. As a response, given the security parameter param, $\mathcal{PPO}$ runs the $PKeyGen$ algorithm to generate a public-private key of the signer (pk_P, pk_P). $\mathcal{PPO}$ then returns pk_P to $\mathcal{A}$. After that, $\mathcal{PPO}$ keeps a record in the $\mathfrak{PL}$, which is $\mathfrak{PL} \leftarrow \mathfrak{PL}(pk_P, sk_P)$.

$\mathcal{PKO}$ oracle: At most q_{PK} times, $\mathcal{A}$ can make a query for the policy signer's private key sk_P to $\mathcal{PKO}$. As a response, $\mathcal{PKO}$ matches the signer's public key pk_P in the list $\mathfrak{PL}$ to obtain the signer's private key sk_P Then, $\mathcal{PKO}$ returns sk_P to $\mathcal{A}$. After that, $\mathcal{PKO}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_P, sk_P)$.

$\mathcal{SSO}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a signature σ on its choice of a message M. As a response, $\mathcal{SSO}$ runs the $Sign$ algorithm to generate a signature σ on a message M corresponding with pk_{TA}, pk_S and $\mathbb{P}$. $\mathcal{SSO}$ then returns σ, M to $\mathcal{A}$. After that, $\mathcal{SSO}$ keeps the record in the $\mathfrak{SS}$, which is $\mathfrak{SS} \leftarrow \mathfrak{SS}(\sigma, M, \mathbb{P})$.

$\mathcal{PSO}$ oracle: At most q_{PS} times, $\mathcal{A}$ can make a query for a policy-controlled signature δ on its choice of a message M, a policy signer's public key pk_P, a signer's public key pk_P, a signature σ and a policy $\mathbb{P}$ as input to $\mathcal{PSO}$. As a response, $\mathcal{PSO}$ runs the $PSign$ algorithm to generate a policy-controlled signature δ on a message M corresponding with pk_{TA}, pk_S, pk_P and $\mathbb{P}$. $\mathcal{PSO}$ then returns the policy-controlled signature δ to $\mathcal{A}$. After that, $\mathcal{PSO}$ keeps a record in the $\mathfrak{PS}$, which is $\mathfrak{PS} \leftarrow \mathfrak{PS}(\delta, \sigma, M, \mathbb{P}, pk_S, pk_P)$.

Note that, for the definition of the $\mathcal{VCO}$ oracle, the reader may refer to Section 5.2.2. Next, we discuss the existential unforgeability of UPCS on the policy signer's side. Let CM-SK-A be the adaptively chosen message, signer's private key exposure and credentials exposure. We also denote by EUF-$UPCS$ the existential unforgeability of UPCS schemes. Let $\mathcal{A}^{CM-SK-A}_{EUF-UPCS}$ be the adaptively chosen message,

the signer's private key exposure and credentials exposure adversary and let $\mathcal{F}$ be a simulator. We denote that $\mathbb{B}^*$ is the entire possible set of credentials of a policy $\mathbb{P}^*$. Let st be the state of information that $\mathcal{A}$ obtains during the learning phase. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential unforgeability $\text{Expt}^{\mathcal{A}^{CM\text{-}SK\text{-}A}_{EUF\text{-}UPCS}}(\ell)$ of UPCS on the policy signer's side: given a choice of messages M and access to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SKO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles in an adaptive way. At the end of the above queries, we assume that $\mathcal{A}$ outputs a forged policy-controlled signature δ^* on a new message M^* with respect to the signer's public key pk_S, the policy signer's public key pk_P and a policy $\mathbb{P}^*$. We denote that $\mathbb{B}^*$ is the entire possible set of credentials of a policy $\mathbb{P}^*$. We say that $\mathcal{A}$ wins the game if:

1. $Accept \leftarrow Verify(M^*, \delta^*, pk_S, pk_P, pk_{TA}, \mathbb{P}^*, \{\mathbb{C}_{i,j,k}\} \in \mathbb{B}^*)$.
2. $\sigma^*, M^*, \mathbb{P}^* \notin \mathfrak{GS}$.

Let $Succ^{CM-SK-A}_{EUF-UPCS}(.)$ be the success probability of $\mathcal{A}^{CM-SK-A}_{EUF-UPCS}$ winning the above game.

Definition 5.4 *We say that a UPCS scheme is ($\mathtt{t}$,q_H,q_{SS},q_{PS},q_{VC},ϵ)-secure existential unforgeable under an adaptive chosen message, a signer's private key exposure and credentials exposure attack if there is no PPT adversary $\mathcal{A}^{CM-SK-A}_{EUF-UPCS}$ such that the success probability $Succ^{CM-SK-A}_{EUF-UPCS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-SK-A}_{EUF-UPCS}$ runs in time at most $\mathtt{t}$, makes at most q_H queries to the random oracle, and at most q_{SS}, q_{PS}, and q_{VC} queries to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, respectively.*

Finally, we discuss the existential unforgeability of UPCS on the signer's side. We denote by $CM\text{-}PSK\text{-}A$ the adaptively chosen message, policy signer's private key exposure and credentials exposure. Let $\mathcal{A}^{CM-PSK-A}_{EUF-UPCS}$ be the adaptively chosen message, the policy signer's private key exposure and credentials exposure adversary. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential unforgeability $\text{Expt}^{\mathcal{A}^{CM\text{-}SK\text{-}A}_{EUF\text{-}UPCS}}(\ell)$ of UPCS on the signer's side: given a choice of messages M and access to the $\boldsymbol{\mathcal{PPO}}$, $\boldsymbol{\mathcal{PKO}}$, $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles in an adaptive way. At the end of the above queries, we assume that $\mathcal{A}$ outputs a forged policy-controlled signature δ^* on a new message M^* with respect to the signer's public key pk_S, the policy signer's public key pk_P

and a policy $\mathbb{P}^*$. We denote that $\mathbb{B}^*$ is the entire possible set of credentials of a policy $\mathbb{P}^*$. We say that $\mathcal{A}$ wins the game if:

1. $Accept \leftarrow Verify(M^*, \delta^*, pk_S, pk_P, pk_{TA}, \mathbb{P}^*, \{\mathbb{C}_{i,j,k}\} \in \mathbb{B}^*)$.
2. $\sigma^*, M^*, \mathbb{P}^* \notin \mathfrak{PS}$.

Let $Succ^{CM-PSK-A}_{EUF-UPCS}(.)$ be the success probability of $\mathcal{A}^{CM-PSK-A}_{EUF-UPCS}$ winning the above game.

Definition 5.5 *We say that a UPCS scheme is ($\mathtt{t}$,q_H,q_{SS},q_{PS},q_{VC},ϵ)-secure existential unforgeable under an adaptive chosen message, a policy signer's private key exposure and credentials exposure attack if there is no PPT adversary $\mathcal{A}^{CM-PSK-A}_{EUF-UPCS}$ such that the success probability $Succ^{CM-PSK-A}_{EUF-UPCS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM-PSK-A}_{EUF-UPCS}$ runs in time at most $\mathtt{t}$, makes at most q_H queries to the random oracle, and at most q_{SS} and q_{VC} queries to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, respectively.*

5.5.3 Coalition-resistance

In this section, we will discuss the coalition-resistance property of UPCS schemes. The coalition-resistance property of UPCS schemes aims to prevent an attacker as a group of corrupted credential holders (verifiers) from verifying a policy-controlled signature σ_* on a message M^* with a policy $\mathbb{P}$, where an attacker does not have sufficient credentials to satisfy the policy $\mathbb{P}$. In fact, the coalition-resistance property of UPCS schemes is the same as the coalition-resistance property of PCS schemes. However, the security model is different. We describe the security model for the coalition-resistance property of UPCS schemes as follows. Let $CRI\text{-}UPCS$ denote the existential coalition-resistance property of UPCS schemes. Let $\mathcal{A}^{CMP-A}_{CRI-UPCS}$ be the adaptively chosen message and chosen policy attack. Let $\mathcal{F}$ be a simulator. Let $\mathfrak{SS}$ and $\mathfrak{PS}$ be algorithms that maintain the list of queried signatures and the list of queried policy-controlled signatures, respectively. Let $\mathfrak{QS}$ be an algorithm that maintains the list of queried credentials. The following experiment between $\mathcal{F}$ and $\mathcal{A}$ describes the existential coalition-resistance property of UPCS schemes. First, the oracles are provided in order to model the ability of adversaries breaking the coalition-resistance property of UPCS schemes which are the $\boldsymbol{\mathcal{VCO}}$, $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{SSO}}$ oracles The definition of the $\boldsymbol{\mathcal{VCO}}$ oracle may refer to Section 5.2.2. For the definition of the $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{SSO}}$ oracle, the reader may refer to Section 5.5.2.

The experiment is divided into two phases. We run them as follows.

1. **Phase** 1**:** With any adaptive strategies, $\mathcal{A}$ arbitrarily sends query requests to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles. The oracles respond as per their design.

2. **Challenge:** $\mathcal{A}$ decides to challenge and outputs σ^*, M^* and $\mathbb{P}^*$ such that:

   a. Given σ^*, $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{PSO}}$ oracle.

   b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issue a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle; however, $\mathcal{A}$ does not make sufficient requests for credentials to satisfy the policy $\mathbb{P}^*$.

   c. $Accept \leftarrow Verify(\mathsf{param}, M^*, \sigma^*, pk_S)$.

   Next, $\mathcal{F}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$ then, given a signature σ^*, a policy $\mathbb{P}^*$, a signer's public key pk_S and a message M^* as input, $\mathcal{F}$ makes a request for a policy-controlled signature to the $\boldsymbol{\mathcal{PSO}}$ oracle and responds $\mathcal{A}$ with δ^* as an output from the $\boldsymbol{\mathcal{PSO}}$ oracle. Otherwise, given a signature σ^*, a policy $\mathbb{P}^*$, a signer's public key pk_S, a policy signer's public key pk_P, a message M^*, a valid policy-controlled signature $\overline{\delta}$ on message M^* with a policy $\mathbb{P}^*$ and a set of credentials $\{\mathbb{C}_{i,j,k}\} \in \mathbb{B}^*$ as input, $\mathcal{F}$ computes a (simulated) invalid policy-controlled signature δ^* and responds to $\mathcal{A}$ with δ^*.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it wants, on one condition, that $\mathcal{A}$ must have at least one set of challenges $M^*, \mathbb{P}^*, \sigma^*, \delta^*$ such that

   a. Given σ^*, $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{PSO}}$ oracle.

   b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issue a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle, however, $\mathcal{A}$ does not have sufficient credentials to satisfy the policy $\mathbb{P}^*$ and to verify σ^*.

   c. $Accept \leftarrow Verify(\mathsf{param}, M^*, \sigma^*, pk_S)$.

4. **Guessing:** On the challenge $M^*, \mathbb{P}^*, \sigma^*, \delta^*$, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ^{CMP-A}_{CRI-UPCS}(.)$ be the success probability of $\mathcal{A}^{CMP-A}_{CRI-UPCS}$ winning the above game.

Definition 5.6 *We say that a UPCS scheme is (*$\mathtt{t}$*,* q_H*,* q_{SS}*,* q_{PS}*,* q_{VC}*,* ϵ*)-secure existential coalition-resistant under a chosen message and chosen policy attack if there is no PPT distinguisher* $\mathcal{A}^{CMP-A}_{CRI-UPCS}$ *such that the success probability* $Succ^{CMP-A}_{CRI-UPCS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}^{CMP-A}_{CRI-UPCS}$ *runs in time at most* $\mathtt{t}$*, makes at most* q_H *queries to the random oracle, and at most* q_{SS}*,*q_{PS} *and* q_{VC} *queries to the* $\boldsymbol{\mathcal{SSO}}$*,* $\boldsymbol{\mathcal{PSO}}$ *and* $\boldsymbol{\mathcal{VCO}}$ *oracles, respectively.*

5.6 The Proposed UPCS Scheme

In this section, we present our concrete construction of UPCS schemes. Let $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$; $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be three distinct random one-way functions that map any string to group $\mathbb{G}_1$ and let $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. We denote by $\mathbb{G}_1$ and $\mathbb{G}_T$ two groups of prime order p. Assume that there exists an efficient computationally bilinear mapping function $\hat{e}$, which maps $\mathbb{G}_1$ to $\mathbb{G}_T$. The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The scheme is described as follows.

Setup: Given a security parameter ℓ as input, a trusted third party randomly chooses a prime $p = poly(1^\ell)$. Select a random generator $g \in \mathbb{G}_1$ and a bilinear mapping function $\hat{e}$. Select hash functions $H_0(.), H_1(.), H_2(.), h(.)$. We denote by $\mathsf{param} = (p, \hat{e}, g, H_0, H_1, H_2, h)$ the system parameter. Then, *Setup* returns param.

$TKeyGen$: Given a system parameter param as input, a trusted authority TA randomly generates a private key sk_{TA} and a public key pk_{TA} as follows: select random integers $\mu, \gamma \in \mathbb{Z}_p$. Let $U = g^\mu; W = g^\gamma$ denote a public key. Therefore, $TKeyGen$ returns $sk_{TA} = (\mu, \gamma)$ as a private key of the trusted authority and $pk_{TA} = (U, W)$ as a public key of the trusted authority.

$SKeyGen$: Given a system parameter param as input, a signer S randomly generates a private key sk_S and a public key pk_S as follows: select a random integer $s \in \mathbb{Z}_p$. Let $\mathbb{S} = g^s$ denote a public key. Therefore, $SKeyGen$ returns $sk_S = s$ as a private key of the signer and $pk_S = \mathbb{S}$ as a public key of the signer.

$PKeyGen$: Given a system parameter param as input, a policy signer P randomly generates a private key sk_P and a public key pk_P as follows: select a random integer $x \in \mathbb{Z}_p$. Let $X = g^x; \hat{X} = W^x$ denote a public key. Therefore,

SKeyGen returns $sk_P = x$ as a private key of the policy signer and $pk_P = (X, \hat{X})$ as a public key of the policy signer.

CreGen: Let P be a statement in the policy, e.g., P ="Manager". An assertion A of P is computed as follows: $A = H_2(\mathsf{P})$. Given a system parameter param, the trusted authority's public key pk_{TA}, the trusted authority's private key sk_{TA} and a set of assertions $A_1, ..., A_n$ that the verifier is allowed to obtain as input, a trusted authority TA randomly generates each verifier's credential string $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$ where i is an index of credentials as follows. TA randomly selects $\nu_i \in \mathbb{Z}_p^*$ and computes each credential $\mathbb{V}_i = U^{1/\nu_i}$; $\mathbb{R}_i = g^{(\mu\gamma)/\nu_i} A_i^{\mu}$; $\mathbb{G}_i = g^{\nu_i}$ and then returns $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$ to the verifier as a credential of assertion A_i. The verifier checks the validity of $\mathbb{C}_i$ as follows.

$$\begin{aligned} \hat{e}(\mathbb{R}_i, g) &\stackrel{?}{=} \hat{e}(A_i, U)\hat{e}(W, \mathbb{V}_i), \\ \hat{e}(\mathbb{V}_i, \mathbb{G}_i) &\stackrel{?}{=} \hat{e}(U, g). \end{aligned}$$

Sign: Given a message M, pk_S and sk_S, S computes $\sigma = H_3(M)^s$ as a BLS short signature on message M.

Verify: Given pk_S, σ and a message M, a signature holder (who is also a policy signer) P checks whether $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H_2(M), \mathbb{S})$ holds or not. If not, then it outputs *Reject*. Otherwise, it outputs *Accept*.

PSign: Let σ be a signature from the signer S. Given param, pk_{TA}, sk_P, pk_P, pk_S, $\mathbb{P} = \bigvee_{i=1}^{a}[\bigwedge_{j=1}^{a_i} A_{i,j}]$ and a message M as input, *PSign* computes a universal policy-controlled signature δ on a message M as follows.

$$\begin{aligned} r_1, r_2, r_3, t &\stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^{r_1},\ \boldsymbol{\delta_2} = X^{r_1},\ \boldsymbol{\delta_3} = \hat{X}^{r_1}, \\ \Psi &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}, \\ \boldsymbol{\delta_4} &= \hat{e}(H_0(\Psi), g)^{r_2},\ \boldsymbol{\delta_5} = \hat{e}(H_0(\Psi), g)^{r_3},\ \Omega = \Psi||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}, \end{aligned}$$

for $i = 1$ to a:

$$R_i = t \oplus h(H_0(\Omega)||i||\hat{e}((\prod_{j=1}^{a_i} A_{i,j})^{r_1 \cdot x}, U)).$$

Then compute

$$\begin{aligned}
\mathbb{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i,\\
\boldsymbol{\delta_6} &= H_0(\Psi)^{r_2}\sigma^{h(\mathbb{M})},\ \boldsymbol{\delta_7} = r_3 + r_2 \cdot h(\mathbb{M}),\\
\mathcal{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i,\\
\boldsymbol{\delta_8} &= H_1(\mathcal{M})^x.
\end{aligned}$$

The universal policy-controlled signature on a message M is
$\delta = (H_0(\Omega), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8}, R_1, ..., R_i)$.

PVerify: Let $\{\mathbb{C}_i\} = \mathbb{C}_{i,1}, ..., \mathbb{C}_{i,a_i}$ be a set of credentials in $\mathbb{B}$ that the verifier possesses. Given pk_S, pk_{TA}, pk_V, $\{\mathbb{C}_i\} \subset \mathbb{B}$, $\mathbb{P}$, δ and a message M, a verifier V first checks whether $\hat{e}(\boldsymbol{\delta_2}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_1}, X)$, $\hat{e}(\boldsymbol{\delta_3}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, W)$ hold or not. If not, then V outputs *Reject*. Otherwise, V computes as follows.

$$\hat{t} = R_i \oplus h(H_0(\Omega)||i||(\prod_{j=1}^{a_i}(\hat{e}(\mathbb{R}_{i,j}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j}, \boldsymbol{\delta_3})^{-1}))).$$

Next, let $\overline{\Psi} = \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}$; $\overline{\Omega} = \overline{\Psi}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}$ and then V checks whether $H_0(\Omega) \stackrel{?}{=} H_0(\overline{\Omega})$. After that, compute

$$\begin{aligned}
\mathbb{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\hat{t}||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i,\\
\mathcal{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||\hat{t}||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i.
\end{aligned}$$

Then, V checks whether $\hat{e}(\boldsymbol{\delta_6}, g) \stackrel{?}{=} \boldsymbol{\delta_4} \cdot \hat{e}(H_3(M), \mathbb{S})^{h(\mathbb{M})}$, $\hat{e}(H_0(\Psi), g)^{\boldsymbol{\delta_7}} \stackrel{?}{=} \boldsymbol{\delta_5} \cdot \boldsymbol{\delta_4}^{h(\mathbb{M})}$, $\hat{e}(\boldsymbol{\delta_8}, g) \stackrel{?}{=} \hat{e}(H_1(\mathcal{M}), X)$. hold or not. If not, then it outputs *Reject*. Otherwise, it outputs *Accept*.

5.7 Security Analysis of UPCS Scheme

5.7.1 Unforgeability: Policy Signer

Theorem 5.4 *Our universal policy-controlled signature scheme is existential unforgeable under an adaptive chosen message, a signer's private key exposure and credentials exposure attack if the CDH assumption holds in the random oracle model.*

Proof: In the following proof, we will show that if there exists a forger $\mathcal{A}$, which runs the existential unforgeability game on the policy signer's side defined in Section 5.5.2,

then we can construct an algorithm $\mathcal{F}$ to solve the CDH problem by using $\mathcal{A}$. First, we illustrate the concept of the proof. The proof will begin with the construction of oracles defined in Section 5.5.2. Next, we will construct a simulator $\mathcal{F}$ and run the existential unforgeability game defined in Section 5.5.2 with a forgery $\mathcal{A}$. Finally, we will analyse the success probability of the existential unforgeability game under an adaptive chosen message, a signer's private key exposure and credentials exposure attack and show that this success probability is reducible to the CDH problem in the random oracle model.

The construction of oracles for the existential unforgeability game describe as follows. Given g, g^a and g^b as an instance of the CDH problem, $\mathcal{F}$ sets g^b as one of the answers for the hash query to the random oracle. Next, $\mathcal{F}$ chooses a random integer $\mu, \gamma \in \mathbb{Z}_p$ and sets $U = g^{\mu}; W = g^{\gamma}$ as a public key of TA. Then, $\mathcal{F}$ sets $X = g^a; \hat{X} = X^{\gamma}$ as the signer's public key pk_S. We note that the algorithms managing the lists of each oracle are simple to construct, and such algorithms will be omitted. From the above setting, it is easy for $\mathcal{F}$ to construct the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SKO}}$,$\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{PSO}}$, $\boldsymbol{\mathcal{VCO}}$, $\boldsymbol{\mathcal{VSO}}$ and $\boldsymbol{\mathcal{HO}}$ oracles as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: Let M be a message that is an input for the hash value to the $\boldsymbol{\mathcal{HO}}$ oracle. If it is a request for a hash value of $H_1(M)$, $\boldsymbol{\mathcal{HO}}$ selects $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d} = 1$ is $\frac{1}{q_H}$. If $\mathsf{d} = 1$ then set $H_1(M) = g^b$ and return $H_1(M)$. Otherwise, $\bar{l} \xleftarrow{\$} \mathbb{Z}_p$; $H_1(M) = g^{\bar{l}}$ and return $H_1(M)$. For $H_0(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_1 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_0(M) = g^{l_1}$. For $H_2(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_2 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_2(M) = g^{l_2}$. For $H_3(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_3 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_3(M) = g^{l_3}$. For $h(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_4 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(M) = l_4$. Then $\boldsymbol{\mathcal{HO}}$ keeps $\bar{l}, l_1, l_2, l_3, l_4$ in the list and this list can be accessed only by $\mathcal{F}$. Note that $\boldsymbol{\mathcal{HO}}$ manages the duplicated hash value of the list by repeating the process such that the output of $\boldsymbol{\mathcal{HO}}$ behaves like a result from the random oracle.

$\boldsymbol{\mathcal{SPO}}$ oracle: $\boldsymbol{\mathcal{SPO}}$ runs $SKeyGen$ to generate the signer's public-private key pair (pk_S, sk_S) and then returns pk_S. After that, $\boldsymbol{\mathcal{SPO}}$ updates (pk_S, sk_S) to the list $\mathfrak{SL}$.

$\boldsymbol{\mathcal{SKO}}$ oracle: Given pk as input, $\boldsymbol{\mathcal{SKO}}$ obtains sk_S from the list $\mathfrak{SL}$ and then returns sk_S. After that, $\boldsymbol{\mathcal{SKO}}$ updates (pk_S, sk_S) to the list $\mathfrak{QR}$.

$\mathcal{VCO}$ oracle: $\mathcal{VCO}$ runs $CreGen$ to generate the credential $\mathbb{C}_i$ of assertion A_i and then returns $\mathbb{C}_i$.

$\mathcal{SSO}$ oracle: Given a message M and a signer's public key pk as input, $\mathcal{SSO}$ accesses the list $\mathfrak{SL}$ to obtain sk_S and runs $Sing$ to generate the signature σ on a message M. Next, $\mathcal{SSO}$ returns σ.

$\mathcal{PSO}$ oracle: Given $\mathbb{P}$, a signature σ, a signer's public key pk_S and a message M as input, $\mathcal{PSO}$ computes a policy-controlled signature as follows.

$$\begin{aligned} r_1, r_2, r_3, t &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^{r_1},\ \boldsymbol{\delta_2} = X^{r_1},\ \boldsymbol{\delta_3} = \hat{X}^{r_1}, \\ \Psi &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}, \\ \boldsymbol{\delta_4} &= \hat{e}(H_0(\Psi), g)^{r_2},\ \boldsymbol{\delta_5} = \hat{e}(H_0(\Psi), g)^{r_3},\ \Omega = \Psi||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}, \end{aligned}$$

for $i = 1$ to a:

$$R_i = t \oplus h(H_0(\Omega)||i||(\prod_{k=1}^{a_i}(\hat{e}(\mathbb{R}_{i,j}, \boldsymbol{\delta_2})\hat{e}(\mathbb{V}_{i,j}, \boldsymbol{\delta_3})^{-1}))).$$

Then compute

$$\begin{aligned} \mathbb{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i, \\ \boldsymbol{\delta_6} &= H_0(\Psi)^{r_2}\sigma^{h(\mathbb{M})},\ \boldsymbol{\delta_7} = r_3 + r_2 \cdot h(\mathbb{M}), \\ \mathcal{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i. \end{aligned}$$

Next, having access to the list of $\bar{l}, l_1, l_2, l_3, l_4, \mathcal{F}$ checks whether $H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}$ $||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||\hat{t}||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i) \stackrel{?}{=} g^b)$. If not, then $\mathcal{F}$ gives $\bar{l}$ to $\mathcal{PSO}$ and $\mathcal{PSO}$ computes $\boldsymbol{\delta_8} = X^{\bar{l}}$. Otherwise, it outputs $\perp$. Finally, $\mathcal{PSO}$ returns a policy-controlled signature on message M, which is $\delta = (H_0(\Omega), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8}, R_1, ..., R_i)$.

$\mathcal{A}$ is given an access to these oracles. Assume that a hash of a string or a message from the random oracle $\mathcal{HO}$ is always queried before $\mathcal{A}$ makes a query to the $\mathcal{SSO}$, $\mathcal{PSO}$ and $\mathcal{VCO}$ oracles, or before it outputs a potential forgery, denoted by $M^*, \delta^*, \mathbb{P}^*, pk_S^*$.

With an adaptive strategy, $\mathcal{A}$ executes the above oracles and outputs a forgery δ^* on a message M^* with respect to $\mathbb{P}^*$ and pk_S^*. We say that $\mathcal{A}$ wins the game if a policy-controlled signature δ^* on message M^* with respect to $\mathbb{P}^*$ and pk_S^* is valid and is not an output from the $\mathcal{PSO}$ oracle.

We denote by $Succ^{CM-SK-A}_{EUF-UPCS} = \epsilon$ the probability that $\mathcal{A}$ wins the game. Let e be the base of the natural logarithm and let $q \geq q_H$ be a polynomial upper bound on the number of queries that $\mathcal{A}$ makes to the $\boldsymbol{\mathcal{HO}}$ oracle. As mentioned above, $\mathcal{A}$ always make a query request to the $\boldsymbol{\mathcal{HO}}$ oracle before making any requests to the $\boldsymbol{\mathcal{PSO}}$ oracle; hence, $q_H \geq q_{PS}$. Therefore, we can analyse the success probability that $\mathcal{A}$ outputs a policy-controlled signature δ^* on message M^* with respect to $\mathbb{P}^*$ and pk^*_S, where $\boldsymbol{\delta^*_8} = H_1(\mathcal{M})^x = (g^b)^x$, and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{SSO}}$ oracle.* The probability of this event is $(1 - \frac{1}{q_H})^{q_{PS}}$. The fact is that $\mathcal{A}$ needs to reserve at least one request for a hash value to output $\boldsymbol{\delta^*_8}$, which is a part of forgery. Therefore, the upper bound for the $\boldsymbol{\mathcal{PSO}}$ oracle is $q_H - 1$ and then the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

- E_2: *$\mathcal{F}$ does not abort after $\mathcal{A}$ output δ^*.* $\mathcal{A}$ needs to reserve at least one request for a hash value to output $\boldsymbol{\delta^*_8}$, which is a part of forgery. However, if

$$H_1(\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i)$$

$$= H_1(\mathcal{M}) \neq g^b$$

  for $\boldsymbol{\delta^*_4}$, then $\mathcal{F}$ aborts the simulation. Hence, the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

The probability that $\mathcal{A}$ wins the above game and outputs a policy-controlled signature δ^* on a message M^*, where $\boldsymbol{\delta^*_8} = H_1(\mathcal{M})^x = (g^b)^x$, is

$$\Pr[Succ^{CM-A}_{EUF-PCS}] \cdot \Pr[Succ^{CM-A}_{EUF-PCS}|E_1|E_2] \geq \epsilon(\frac{q_H}{e \cdot (q_H - 1)})^2.$$

$\mathcal{F}$ obtains $\boldsymbol{\delta^*_8} = H_1(\mathcal{M})^x$ from the above outputs. Since $H_1(\mathcal{M}) = g^b$ and $x = a$, $\mathcal{F}$ returns $\boldsymbol{\delta^*_8} = g^{ab}$ as an output for the CDH problem with non-negligible probability as mentioned above. □

5.7.2 Unforgeability: Signer

Theorem 5.5 *Our universal policy-controlled signature scheme is existential unforgeable under an adaptive chosen message, a policy signer's private key exposure and credentials exposure attack if the BLS scheme is existential unforgeability under an adaptive chosen message attack in the random oracle model.*

Proof: In the following proof, we will show that if there exists a forger $\mathcal{A}$, which runs the existential unforgeability game on the signer's side defined in Section 5.5.2, then we can construct an algorithm $\mathcal{F}$ to attack the unforgeability of the BLS scheme by using $\mathcal{A}$. Let $\mathcal{J}$ be the challenger of the unforgeability of the BLS scheme. We construct the proof in a similar way to the proof for Theorem 5.4. First, the construction of the $\boldsymbol{\mathcal{HO}}$, $\boldsymbol{\mathcal{PPO}}$, $\boldsymbol{\mathcal{PKO}}$ and $\boldsymbol{\mathcal{SSO}}$ oracles is given as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: Let M be a message that is an input for the hash value to the $\boldsymbol{\mathcal{HO}}$ oracle. For $H_0(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $\bar{l} \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_0(M) = g^{\bar{l}}$. For $H_1(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_1 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_1(M) = g^{l_1}$. For $H_2(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_2 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_2(M) = g^{l_2}$. For $h(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_3 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(M) = l_3$. For $H_3(M)$, $\boldsymbol{\mathcal{HO}}$ forwards the queries to the challenger $\mathcal{J}$. Then $\boldsymbol{\mathcal{HO}}$ keeps $\bar{l}, l_1, l_2, l_3$ in the list and this list can be accessed only by $\mathcal{F}$. Note that $\boldsymbol{\mathcal{HO}}$ manages the duplicated hash value of the list by repeating the process such that the output of $\boldsymbol{\mathcal{HO}}$ behaves like a result from the random oracle.

$\boldsymbol{\mathcal{PPO}}$ oracle: $\boldsymbol{\mathcal{PPO}}$ runs $PKeyGen$ to generate the policy signer's public-private key pair (pk_P, sk_P) and then returns pk_P. After that, $\boldsymbol{\mathcal{PPO}}$ updates (pk_P, sk_P) to the list $\mathfrak{SL}$.

$\boldsymbol{\mathcal{PKO}}$ oracle: Given pk_P as input, $\boldsymbol{\mathcal{PKO}}$ obtains sk_P from the list $\mathfrak{SL}$ and then returns sk_P. After that, $\boldsymbol{\mathcal{PKO}}$ updates (pk_P, sk_P) to the list $\mathfrak{QR}$.

$\boldsymbol{\mathcal{VCO}}$ oracle: $\boldsymbol{\mathcal{VCO}}$ runs $CreGen$ to generate the credential $\mathbb{C}_i$ of assertion A_i and then returns $\mathbb{C}_i$.

$\boldsymbol{\mathcal{SSO}}$ oracle: Given a message M as input, $\boldsymbol{\mathcal{SSO}}$ forwards the queries to the challenger $\mathcal{J}$. When $\mathcal{J}$ replies with σ, $\boldsymbol{\mathcal{SSO}}$ returns σ. After that, $\boldsymbol{\mathcal{SSO}}$ updates (σ, M) to the list $\mathfrak{SS}$.

$\mathcal{A}$ is given an access to these oracles. Assume that a hash of a string or a message from the random oracle $\boldsymbol{\mathcal{HO}}$ is always queried before $\mathcal{A}$ makes a query to the $\boldsymbol{\mathcal{SSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles, or before it outputs a potential forgery, denoted by $M^*, \delta^*, \mathbb{P}^*, pk_S$.

With an adaptive strategy, $\mathcal{A}$ executes the above oracles and outputs a forgery δ^* on a message M^* with respect to $\mathbb{P}^*$ and pk_S. We say that $\mathcal{A}$ wins the game if

a policy-controlled signature δ^* on message M^* with respect to $\mathbb{P}^*$ and pk_S is valid and $\mathcal{A}$ never makes a request for a signature σ^* to $\boldsymbol{\mathcal{SSO}}$ on input of a message M^*.

With a trivial probability, $\mathcal{F}$ runs $\mathcal{A}$ on the same setting but with a different hash function $h'(.)$. $\mathcal{F}$ obtains the second policy-controlled signature δ' and computes $\sigma = (\boldsymbol{\delta^*_6} \cdot \delta'^{-1}_6)^{1/(h(\mathbb{M})-h'(\mathbb{M}))}$. Finally, $\mathcal{F}$ outputs a BLS signature σ on a new message M^* and gives it to $\mathcal{J}$. □

5.7.3 Coalition-resistance

Theorem 5.6 *In the random oracle model, the proposed universal policy-controlled signature scheme is existential coalition-resistant against an adaptively chosen message and chosen policy attack $\mathcal{A}^{CMP-A}_{CRI-UPCS}$ if the DBDH assumption is held.*

Proof: In the following proof, we will show that if there exists a forger $\mathcal{A}$, which runs and wins the existential coalition-resistance game defined in Section 5.5.3, then there exists an adversary $\mathcal{F}$ that answers the DBDH problem by using $\mathcal{A}$ as a tool. The guidelines for this proof are similar to the proof for Theorem 5.3. We start by constructing the oracles as follows: given g, g^a, g^b, g^c and Z as an instance of the DBDH problem, $\mathcal{F}$ sets g^a as one of the answers for the hash query to the random oracle. Next, $\mathcal{F}$ randomly selects x, γ and sets $U = g^b; W = g^\gamma$ as the TA public key. Then, $\mathcal{F}$ sets $X = g^x; \hat{X} = g^{x \cdot \gamma}$ as the signer's public key. Assume that there exists an algorithm managing the list of each query and such an algorithm will be omitted. From the above setting, $\mathcal{F}$ constructs the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{PSO}}$, $\boldsymbol{\mathcal{VCO}}$ and random oracle $\boldsymbol{\mathcal{HO}}$ as follows.

$\boldsymbol{\mathcal{HO}}$ oracle: Let M be a message that is an input for the hash value to the $\boldsymbol{\mathcal{HO}}$ oracle. If it is a request for a hash value of $H_2(M)$, $\boldsymbol{\mathcal{HO}}$ selects $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d} = 1$ is $\frac{1}{q_H}$. If $\mathsf{d} = 1$ then set $H_2(M) = g^a$ and return $H_2(M)$. Otherwise, $\bar{l} \xleftarrow{\$} \mathbb{Z}_p$; $H_1(M) = g^{\bar{l}}$ and return $H_2(M)$. For $H_0(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_1 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_0(M) = g^{l_1}$. For $H_1(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_2 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_2(M) = g^{l_2}$. For $H_3(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_3 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $H_3(M) = g^{l_3}$. For $h(M)$, $\boldsymbol{\mathcal{HO}}$ chooses $l_4 \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(M) = l_4$. Then $\boldsymbol{\mathcal{HO}}$ keeps $\bar{l}, l_1, l_2, l_3, l_4$ in a list and this list can be accessed only by $\mathcal{F}$. Note that $\boldsymbol{\mathcal{HO}}$ manages the duplicated hash value of the list by repeating the process such that the output of $\boldsymbol{\mathcal{HO}}$ behaves like a result from the random oracle.

$\mathcal{VCO}$ oracle: Given A_i as input, $\mathcal{VCO}$ accesses to the $\mathcal{HO}$ oracle for a matching pair of $P_i, A_i = H_2(P_i)$, if the $\mathcal{HO}$ oracle returns $A_i = H_2(P_i) = g^a$ then $\mathcal{VCO}$ outputs $\perp$. Otherwise, $\mathcal{F}$ accesses the list in the $\mathcal{HO}$ oracle and returns $\bar{l}_i : A_i = g^{\bar{l}_i}$ to the $\mathcal{VCO}$ oracle. Then the $\mathcal{VCO}$ oracle selects a random integer $\nu \in \mathbb{Z}_p$ and computes $\mathbb{V}_i = U^{1/\nu_i}$; $\mathbb{R}_i = U^{\gamma/\nu_i} U^{\bar{l}_i}$; $\mathbb{G}_i = g^{\nu_i}$ as a credential of A_i. The $\mathcal{VCO}$ oracle returns $\mathbb{C}_i = (\mathbb{V}_i, \mathbb{R}_i, \mathbb{G}_i)$.

$\mathcal{SSO}$ oracle: Given a message M as input, $\mathcal{SSO}$, with access to sk_S, runs $Sing$ to generate the signature σ on message M and then returns σ.

$\mathcal{PSO}$ oracle: Given $\mathbb{P}$, a signature σ, a signer's public key pk_S and a message M as input, $\mathcal{PSO}$ computes a policy-controlled signature as follows.

$$\begin{aligned} r_1, r_2, r_3, t &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^{r_1},\ \boldsymbol{\delta_2} = X^{r_1},\ \boldsymbol{\delta_3} = \hat{X}^{r_1}, \\ \Psi &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}, \\ \boldsymbol{\delta_4} &= \hat{e}(H_0(\Psi), g)^{r_2},\ \boldsymbol{\delta_5} = \hat{e}(H_0(\Psi), g)^{r_3},\ \Omega = \Psi||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}, \end{aligned}$$

for $i = 1$ to a:

$$R_i = t \oplus h(H_0(\Omega)||i||\hat{e}((\prod_{j=1}^{a_i} A_{i,j})^{r_1 \cdot x}, U)).$$

Then compute

$$\begin{aligned} \mathbb{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i, \\ \boldsymbol{\delta_6} &= H_0(\Psi)^{r_2}\sigma^{h(\mathbb{M})},\ \boldsymbol{\delta_7} = r_3 + r_2 \cdot h(\mathbb{M}), \\ \mathcal{M} &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\boldsymbol{\delta_5}||\boldsymbol{\delta_6}||\boldsymbol{\delta_7}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}||R_1||...||R_i, \\ \boldsymbol{\delta_8} &= H_1(\mathcal{M})^x. \end{aligned}$$

Finally, $\mathcal{PSO}$ returns a policy-controlled signature on message M, which is $\delta = (H_0(\Omega), \boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8}, R_1, ..., R_i)$.

$\mathcal{A}$ is given an access to these oracles. Assume that a hash of message M from the $\mathcal{HO}$ oracle is always queried before $\mathcal{A}$ makes a query request to the $\mathcal{SSO}$, $\mathcal{PSO}$ and $\mathcal{VCO}$ oracles, or before it outputs a decision bit b.

Now, we run an experiment as defined in Section 5.5.3 as follows.

1. **Phase 1:** $\mathcal{A}$ arbitrarily sends query requests to the $\mathcal{SSO}$, $\mathcal{PSO}$ and $\mathcal{VCO}$ oracles. The oracles respond as above.

2. **Challenge:** After the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{P}^*$. The challenge set M^*, σ^* and $\mathbb{P}^*$ is valid if the following conditions hold.

   1. Given $\mathbb{P}^*$ and M^* as input, $\mathcal{A}$ issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{PSO}}$ oracle.

   2. $\mathcal{A}$ does not have sufficient credentials to satisfy the policy $\mathbb{P}^*$.

   Next, $\mathcal{F}$ computes a response as follows.

$$\begin{aligned} r_1, r_2, r_3, t &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta^*_1} = g^{r_1},\ \boldsymbol{\delta^*_2} = X^{r_1},\ \boldsymbol{\delta^*_3} = \hat{X}^{r_1}, \\ \Psi &= \boldsymbol{\delta^*_1}||\boldsymbol{\delta^*_2}||\boldsymbol{\delta^*_3}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}^*, \\ \boldsymbol{\delta^*_4} &= \hat{e}(H_0(\Psi), g)^{r_2},\ \boldsymbol{\delta^*_5} = \hat{e}(H_0(\Psi), g)^{r_3},\ \Omega = \Psi||\boldsymbol{\delta^*_4}||\boldsymbol{\delta^*_5}, \end{aligned}$$

   for $i = 1$ to a: if $A_{i^*,j^*} = g^a$ then compute as follows.

$$R_i = t \oplus h(H_0(\Omega)||i||\hat{e}((\prod_{j=1, j\neq j^*}^{a_i} g^{c\cdot\bar{l}_{i,j}})^{r\cdot x}, U)Z^{r\cdot x}).$$

   Otherwise, compute as follows.

$$R_i = t \oplus h(H_0(\Omega)||i||\hat{e}((\prod_{j=1}^{a_i} g^{c\cdot\bar{l}_{i,j}})^{r\cdot x}, U)).$$

   Then compute

$$\begin{aligned} \mathbb{M} &= \boldsymbol{\delta^*_1}||\boldsymbol{\delta^*_2}||\boldsymbol{\delta^*_3}||\boldsymbol{\delta^*_4}||\boldsymbol{\delta^*_5}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}^*||R_1||...||R_i, \\ \boldsymbol{\delta^*_6} &= H_0(\Psi)^{r_2}\sigma^{*h(\mathbb{M})},\ \boldsymbol{\delta^*_7} = r_3 + r_2 \cdot h(\mathbb{M}), \\ \mathcal{M} &= \boldsymbol{\delta^*_1}||\boldsymbol{\delta^*_2}||\boldsymbol{\delta^*_3}||\boldsymbol{\delta^*_4}||\boldsymbol{\delta^*_5}||\boldsymbol{\delta^*_6}||\boldsymbol{\delta^*_7}||t||pk_P||pk_S||pk_{TA}||\mathbb{P}^*||R_1||...||R_i, \\ \boldsymbol{\delta^*_8} &= H_1(\mathcal{M})^x. \end{aligned}$$

   Finally, $\mathcal{F}$ returns a policy-controlled signature on message M, which is

$$\delta^* = (H_0(\Omega), \boldsymbol{\delta^*_1}, \boldsymbol{\delta^*_2}, \boldsymbol{\delta^*_3}, \boldsymbol{\delta^*_4}, \boldsymbol{\delta^*_5}, \boldsymbol{\delta^*_6}, \boldsymbol{\delta^*_7}, \boldsymbol{\delta^*_8}, R_1, ..., R_i).$$

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can return to *Phase* 1 or *Challenge* as many times as it requests, on one condition, that $\mathcal{A}$ must have at least one set of challenges (which is a set of M^*, $\mathbb{P}^*$, σ^* and δ^*) such that

   a. Given $\mathbb{P}^*$ as input, σ^* and M^*, $\mathcal{A}$ never issues a request for a policy-controlled signature to the $\boldsymbol{\mathcal{PSO}}$ oracle.

b. Given $\mathbb{P}^*$ as input, $\mathcal{A}$ can issues a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle, however, $\mathcal{A}$ does not have sufficient credentials to satisfy the policy $\mathbb{P}^*$.

4. **Guessing:** With a challenge set of M^*, $\mathbb{P}^*$, σ^* and δ^*, $\mathcal{A}$ finally outputs a guess b'.

From the above experiment, we can solve the DBDH problem when $\mathcal{A}$ wins or aborts the experiment on a condition that $\mathcal{A}$ picks the challenge $M^*, \mathbb{P}^*, \sigma^*$, and δ^*, which is inserted with Z and g^c, and a credential for $A_{i,j^*} = g^a$ was never queried. Since $A_{i,j^*} = g^a$ contained in α_i, $\mathcal{A}$ can use it to check whether

$$\hat{e}((\prod_{j=1, j\neq j^*}^{a_i} g^{c\cdot\bar{l}_{i,j}})^{r\cdot x}, U)Z^{r\cdot x} \stackrel{?}{=} \prod_{j=1}^{a_i}(\hat{e}(\mathbb{R}_{i,j}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j}, \boldsymbol{\delta_3})^{-1})$$

holds or not. $\mathcal{A}$ will not abort the game if the above holds.

To link the experiment to the DBDH problem, the following shows that $\mathcal{A}$ will not abort if $Z = \hat{e}(g,g)^{a\cdot b\cdot c}$.

$$\begin{aligned}
\hat{e}((\prod_{j=1, j\neq j^*}^{a_i} g^{c\cdot\bar{l}_{i,j}})^{r\cdot x}, U)Z^{r\cdot x} &= \prod_{j=1}^{a_i}(\hat{e}(\mathbb{R}_{i,j}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j}, \boldsymbol{\delta_3})^{-1}) \\
\hat{e}((\prod_{j=1, j\neq j^*}^{a_i} g^{c\cdot\bar{l}_{i,j}})^{r\cdot x}, U)Z^{r\cdot x} &= \prod_{j=1, j\neq j^*}^{a_i}(\hat{e}(\mathbb{R}_{i,j}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j}, \boldsymbol{\delta_3})^{-1}) \cdot \\
&\quad \hat{e}(\mathbb{R}_{i,j^*}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j^*}, \boldsymbol{\delta_3})^{-1} \\
Z^{r\cdot x} &= \hat{e}(\mathbb{R}_{i,j^*}, \boldsymbol{\delta_2}) \cdot \hat{e}(\mathbb{V}_{i,j^*}, \boldsymbol{\delta_3})^{-1} \\
Z^{r\cdot x} &= \hat{e}(U^{\gamma/\nu_{i,j^*}} A_{i,j^*}^b, g^{c\cdot x\cdot r}) \cdot \hat{e}(U^{1/\nu_{i,j^*}}, g^{c\cdot\gamma\cdot x\cdot r})^{-1} \\
Z^{r\cdot x} &= \hat{e}(A_{i,j^*}^b, g^{c\cdot x\cdot r}) \\
Z^{r\cdot x} &= \hat{e}((g^a)^b, g^{c\cdot x\cdot r}) \\
Z^{r\cdot x} &= \hat{e}(g,g)^{a\cdot b\cdot c\cdot x\cdot r} \\
Z &= \hat{e}(g,g)^{a\cdot b\cdot c}.
\end{aligned}$$

Suppose $\mathcal{A}$ wins the game with the probability $Succ_{CRI-UPCS}^{CMP-A} = \epsilon$. We denote by e the base of the natural logarithm and let $q \geq q_H$ be a polynomial upper bound of queries that $\mathcal{A}$ makes to the $\boldsymbol{\mathcal{HO}}$ oracle. Note that $q << p$. As mentioned above, $\mathcal{A}$ always makes a query request to the $\boldsymbol{\mathcal{HO}}$ oracle before it makes any requests to the $\boldsymbol{\mathcal{PSO}}$ and $\boldsymbol{\mathcal{VCO}}$ oracles; hence, $q_H \geq q_{VC}$ and $q_H \geq q_{PS}$. Therefore, we can analyse the probability that $\mathcal{A}$'s guess is correct and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{VCO}}$ oracle.* The probability of this event is $(1 - \frac{1}{q_H})^{q_{VC}}$. The fact is that $\mathcal{A}$ needs to reserve at least one request for a credential of $A^*_{i,j}$ and one request for a hash value of $A^*_{i,j} = H_2(P^*_{i,j})$, which is a part of policy $\mathbb{P}^*$. Therefore, the upper bound for the $\boldsymbol{\mathcal{VCO}}$ oracle is $q_H - 1$ and then the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.
- E_2: *$\mathcal{F}$ does not abort after* Phase 1 *and* Phase 2. Since we have assumed that $\mathcal{A}$ follows the experiment and outputs a guess with a valid challenge $(M^*, \mathbb{P}^*, \sigma^*, \delta^*)$, then the probability of this event is 1.

The probability that $\mathcal{A}$ wins the above game and outputs a correct guess $b' = b$ is $\Pr[Succ^{CMP-A}_{CRI-UPCS}] \cdot \Pr[Succ^{CMP-A}_{CRI-UPCS} | E_1 | E_2] \geq \epsilon \frac{q_H}{e \cdot (q_H - 1)}$. Let ϵ' be an advantage in solving the DBDH problem. From the above game, $\mathcal{F}$ outputs a guess for the DBDH problem with $\mathcal{A}$'s guess. However, we note that $\mathcal{A}$ can choose a challenge policy $\mathbb{P}^*$ and try to guess from one of the credentials that $\mathcal{A}$ does not make a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle. Thus, there is a event that $\mathcal{A}$'s guess is not a correct guess for the DBDH problem, which is when $A^*_{i,j}$ is not chosen by $\mathcal{A}$. The probability for this event is $\frac{1}{a}$. Therefore, the probability that $\mathcal{F}$ can output a correct guess for the DBDH problem by using $\mathcal{A}$ is $\epsilon' \geq \frac{1}{a} \epsilon \frac{q_H}{e \cdot (q_H - 1)}$. Hence, the probability that $\mathcal{A}$ can break the existential coalition-resistance property of the UPCS scheme against adaptively chosen message and chosen policy attack is $\epsilon \leq \epsilon' a \cdot e \cdot (q_H - 1)/q_H$. Since $a \leq q_H << q$, the analysis of the above probabilities shows that success breaking the existential coalition-resistance property of the proposed PCS scheme is non-negligible if the probability of breaking the DBDH problem is non-negligible. □

5.8 Definition of Multi-level Controlled Signature Scheme (MLCS)

In this section, we give a definition of multi-level controlled signature (MLCS) schemes that allow only verifiers, who hold a credential for a certain security level specified by a signer, to verify the authenticity of the signed message. In other words, only a verifier that satisfies the security level specified by the signer can verify the multi-level controlled signature that generated by a signer. The practical scenario

of multi-level controlled signature schemes is mentioned in Section 5.1. We provide an outline of our MLCS scheme as follows.

5.8.1 Outline of MLCS

Let TA be a trusted authority that issues credentials associated with a security level in a multi-level security system. In MLCS schemes, there are three main players, which are a signer, a verifier and a trusted authority TA. A signer S generates a signature that can be verified *only* by a verifier V who holds a credential satisfying the multi-level security policy. TA is responsible for issuing a credential for V. Let A_V denote a security level in the multi-level security policy. We define $\mathbb{ML}$ to be a multi-level security policy, which contains a policy that indicates a level of security clearance of the verifier. Without losing generality, we assume that the order of the security levels increases, for example, a higher number means a higher security level[1]. For instance, $\mathbb{ML} =$ "$A_V > n$" where n is the number indicating the security level. Generally, we can use another type of index or symbol to indicate the security level. A multi-level controlled signature scheme Σ is a 6-tuple ($Setup$, $TKeyGen$, $SKeyGen$, $CreGen$, $Sign$, $Verify$), which is described as follows.

System Parameter Generation ($Setup$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, outputs the system parameter param. That is,

$$Setup(1^{\ell}) \rightarrow \mathsf{param}.$$

TA Key Generator ($TKeyGen$):

This is a probabilistic algorithm that, given the system parameter param as input, outputs the private key (sk_{TA}) and the public parameter (pk_{TA}) of a trusted authority. That is,

$$TKeyGen(\mathsf{param}) \rightarrow (pk_{TA}, sk_{TA}).$$

Signer Key Generator ($SKeyGen$):

This is a probabilistic algorithm that, given a system parameter param and a public key of the trusted authority pk_{TA} as input, outputs the private key (sk_S) and the public parameter (pk_S) of a signer. That is,

$$SKeyGen(\mathsf{param}, pk_{TA}) \rightarrow (pk_S, sk_S).$$

[1]We note that for a decreasing order of security levels, our scheme can be slightly modified.

Verifier Credential Generator ($CreGen$):

This is a probabilistic algorithm that, given the system parameter param, the TA's private key, and an assertion A_V indicated a security level of a verifier as input, outputs a credential for verifier $\mathbb{C}$. That is,

$$CreGen(\mathsf{param}, sk_{TA}, A_V) \rightarrow \mathbb{C}.$$

Multi-level Controlled Signature Signing ($Sign$):

This is a probabilistic algorithm that, given the system parameter param, the trusted authority's public key pk_{TA}, the signer's private key sk_S, the signer's public key pk_S, a message M and the multi-level security policy $\mathbb{ML}$ as input, outputs a signer's signature δ. That is,

$$Sign(\mathsf{param}, M, sk_S, pk_S, pk_{TA}, \mathbb{ML}) \rightarrow \delta.$$

Multi-level Controlled Signature Verification ($Verify$):

This is a deterministic algorithm that, given the system parameter param, the trusted authority's public key pk_{TA}, the signer's public key pk_S, the multi-level security policy $\mathbb{ML}$, a credential $\mathbb{C}$, a message M and a signature δ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(\mathsf{param}, M, \delta, pk_{TA}, pk_S, \mathbb{ML}, \mathbb{C}) \rightarrow \mathsf{d}.$$

5.8.2 Unforgeability

The unforgeability property of MLCS schemes is to prevent an attacker that has access to the credential oracle from generating a multi-level controlled signature δ^* on a new message M^*. Formally, the unforgeability in this model provides assurance that someone, with access to the $\boldsymbol{\mathcal{SSO}}$ oracle, the $\boldsymbol{\mathcal{VCO}}$ oracle, the signer's public key pk_S and the trusted authority's public key, should be unable to produce a multi-level controlled signature on a new message M^* even if it arbitrarily chooses a multi-level security policy $\mathbb{ML}$, a message M and the entire credentials as input. Let CM-A denote the adaptive chosen message and credentials exposure attack and let EUF-$MLCS$ denote the existential unforgeability of a MLCS scheme. Let $\mathcal{A}$ be the adaptively chosen message and credentials exposure adversary that attack the unforgeability of a MLCS scheme. The experiment between the adversary $\mathcal{A}$ and a simulator $\mathcal{F}$ models the security against existential unforgeability under an adaptive chosen message and credentials exposure attack as described below.

First, the oracles are provided in order to model the ability of adversaries to break the unforgeability of a MLCS scheme as shown below.

$\mathcal{SSO}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a signature δ on its choice of a message M. As a response, $\mathcal{SSO}$ runs the $Sign$ algorithm to generate a signature δ on a message M corresponding with pk_{TA}, pk_S and $\mathbb{ML}$. $\mathcal{SSO}$ then returns δ, M to $\mathcal{A}$.

$\mathcal{VCO}$ oracle: At most q_{VC} times, $\mathcal{A}$ can make a query for the credential $\mathbb{C}_i$ corresponding to the arbitrarily chosen security level A_V. As a response, $\mathcal{VCO}$ replies to $\mathcal{A}$ with corresponding credentials $\mathbb{C}$.

Then, with an adaptive strategy, $\mathcal{A}$ arbitrarily makes queries to the $\mathcal{SSO}$ and $\mathcal{VCO}$ oracles on its choice of a message M. Let $\mathbb{C}$ be the credentials for the entire security level. For instance, if the system has 12 security levels, then $\mathbb{C} = (\mathbb{V}_1, ..., \mathbb{V}_{12}, \mathbb{R}_1, ..., \mathbb{R}_{12})$. After the queries are processed, assume that $\mathcal{A}$ outputs a forged signature δ^* on a new message M^* with respect to the public key pk_S and multi-level security policy $\mathbb{ML}^*$. $\mathcal{A}$ wins the experiment if:

1. $Accept \leftarrow Verify(M^*, \delta^*, pk_S, \mathbb{ML}^*, \mathbb{C})$.
2. On input $M^*, pk_S, \mathbb{ML}^*$, $\mathcal{A}$ never makes a request for a multi-level controlled signature to the $\mathcal{SSO}$ oracle.

We denote $Succ^{CM\text{-}A}_{EUF\text{-}MLCS}(.)$ as the success probability of $\mathcal{A}^{CM\text{-}A}_{EUF\text{-}MLCS}$ winning the above experiment.

Definition 5.7 *A MLCS scheme is ($\mathtt{t}$,q_H,q_{SS},q_{VC},ϵ)-secure existential unforgeable under a chosen message and credentials exposure attack if there is no PPT adversary $\mathcal{A}^{CM\text{-}A}_{EUF\text{-}MLCS}$ such that the success probability $Succ^{CM\text{-}A}_{EUF\text{-}MLCS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM\text{-}A}_{EUF\text{-}MLCS}$ runs in time at most $\mathtt{t}$, and makes at most q_H hash queries, q_{SS} signing queries and q_{VC} verification queries.*

5.8.3 Coalition-resistance

In this section, we will describe the coalition-resistance property of a MLCS scheme. This property aims to prevent an attacker as a group of corrupted credential holders (verifiers) from verifying a multi-level controlled signature δ^* on a message M^* with a multi-level security policy $\mathbb{ML}$, where the attacker does not have the credentials

to satisfy the security level indicated in $\mathbb{ML}$. Note that the unforgeability property implies security against the coalition-resistance's attacker for trying to forge the signature. Since the attacker trying to break the unforgeability property possesses the credentials for the entire security level, the ability of this attacker implies the ability of coalition-resistance's attacker.

Let $CR\text{-}MLCS$ denote the existential coalition-resistance property of a MLCS scheme. Let $\mathcal{A}^{CMP-A}_{CRI-PCS}$ be the adaptively chosen message and chosen multi-level security policy distinguisher and let $\mathcal{F}$ be a simulator. The experiment between the adversary $\mathcal{A}$ and a simulator $\mathcal{F}$ models the security against the existential coalition-resistance property of a MLCS scheme under a chosen message and chosen multi-level security policy attack, which is described below.

First, the oracles are provided in order to model the ability of adversaries to break the unforgeability of a MLCS scheme as shown below.

$\mathcal{SSO}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a signature δ on its choice of a message M. As a response, $\mathcal{SSO}$ runs the $Sign$ algorithm to generate a signature δ on a message M corresponding with pk_{TA}, pk_S and $\mathbb{ML}$. $\mathcal{SSO}$ then returns δ, M to $\mathcal{A}$.

$\mathcal{VCO}$ oracle: At most q_{VC} times, $\mathcal{A}$ can make a query for the credential $\mathbb{C}_i$ corresponding to the arbitrarily chosen security level A_V. As a response, $\mathcal{VCO}$ replies to $\mathcal{A}$ with corresponding credentials $\mathbb{C}$.

$\mathcal{VSO}$ oracle: At most q_{VS} times, $\mathcal{A}$ can make a query for the verification of a signature δ on a message M to $\mathcal{VSO}$ with a signature δ and message M as input. As a response, $\mathcal{VSO}$ returns with a decision d, which is *Accept* or *Reject* corresponding to a validation of signature δ.

Then, we divide the game into two phases and run them as follows.

1. **Phase** 1**:** With any adaptive strategies, $\mathcal{A}$ arbitrarily issues a query request to the $\mathcal{SSO}$, $\mathcal{VCO}$ and $\mathcal{VSO}$ oracles. The oracles respond as per their design.

2. **Challenge:** After the first phase, $\mathcal{A}$ outputs M^* and $\mathbb{ML}^* =$ "$A_V \geq l$" such that:

   a. Given $\mathbb{ML}^*$ and M^* as input, $\mathcal{A}$ never issues a request for a multi-level controlled signature to the $\mathcal{SSO}$ oracle.

b. With $\mathbb{ML}^* =$ "$A_V \geq l$", $\mathcal{A}$ can issue a request for credentials to the $\boldsymbol{\mathcal{VCO}}$ oracle for a security level $A_V < l$.

If the above condition is held, $\mathcal{F}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$ then, given a multi-level security policy $\mathbb{ML}^*$ and a message M^* as input, $\mathcal{F}$ issues a request for a multi-level controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle. Then $\mathcal{F}$ responds to $\mathcal{A}$ with δ^* as an output from the $\boldsymbol{\mathcal{SSO}}$ oracle. Otherwise, given a multi-level security policy $\mathbb{ML}^*$, a message M^*, a valid multi-level controlled signature δ on message M^* with $\mathbb{ML}^*$ and a credentials $\mathbb{C}$ as input, $\mathcal{F}$ computes a (simulated) invalid multi-level controlled signature δ^*. Then $\mathcal{F}$ responds to $\mathcal{A}$ with δ^*.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can arbitrarily return to *Phase* 1 or *Challenge*. On one condition, that at least one set of challenges $M^*, \mathbb{ML}^*, \delta^*$ must be valid and satisfy the condition in the challenge phase, and that it must not be submitted to $\boldsymbol{\mathcal{VSO}}$ for verification.

4. **Guessing:** $\mathcal{A}$ finally outputs a guess b' based on a challenge $M^*, \mathbb{ML}^*, \delta^*$. The distinguisher wins the game if $b = b'$.

We denote by $Succ^{CMP\text{-}A}_{CR\text{-}MLCS}(.)$ the success probability of $\mathcal{A}^{CMP\text{-}A}_{CR\text{-}MLCS}$ winning the above experiment.

Definition 5.8 *A MLCS scheme is (*$\mathtt{t}$*,*q_H*,*q_{SS}*,*q_{VC}*,*ϵ*)-secure existential coalition-resistant under a chosen message and chosen multi-level security policy attack if there is no PPT distinguisher* $\mathcal{A}^{CMP\text{-}A}_{CR\text{-}MLCS}$ *such that the success probability* $Succ^{CMP\text{-}A}_{CR\text{-}MLCS}(\ell) = |\mathbf{Pr}[b = b'] - \mathbf{Pr}[b \neq b']| = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}^{CMP\text{-}A}_{CR\text{-}MLCS}$ *runs in time at most* $\mathtt{t}$*, and makes at most* q_H *hash queries,* q_{SS} *signing queries and* q_{VC} *verification queries.*

5.9 The First Proposed MLCS Scheme

In this section, we present our first concrete construction of MLCS schemes. Let $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ be a collision-resistant hash function. Let $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ denote two groups of prime order p. Let $\hat{e}$ be the bilinear mapping function, which maps $\mathbb{G}_1$ to $\mathbb{G}_T$. The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The scheme is described as follows.

- *Setup*: Given a security parameter ℓ as input, a trusted third party randomly chooses a prime $p = poly(1^{\ell})$. Choose a random generator $g \in \mathbb{G}_1$ and a bilinear mapping function $\hat{e}$. Select two hash functions $H(.)$ and $h(.)$ Let us denote by $\mathsf{param} = (p, \hat{e}, g, H, h)$ the system parameter. Then, *Setup* returns param.

- $TKeyGen$: Let n be a number of security levels. Given a system parameter param as input, a trusted authority TA randomly generates a private key sk_{TA} and a public key pk_{TA} for each security level as follows: select random integers $\mu_0, ..., \mu_n, \gamma_0, ..., \gamma_n, a, b, c_1, ..., c_n \in \mathbb{Z}_p$. Let $pk_{TA} = (U_0 = g^{\mu_0}, ..., U_n = g^{\mu_n}, W_0 = g^{\gamma_0}, ..., W_n = g^{\gamma_n}, \mathcal{A} = g^a, \mathcal{B} = g^b)$ denote a public key. Then, $TKeyGen$ returns $sk_{TA} = (\mu_0, ..., \mu_n, \gamma_0, ..., \gamma_n, a, b, c_1, ..., c_n)$ as a private key of the trusted authority and $pk_{TA} = (U_0, ..., U_n, W_0, ..., W_n, \mathcal{A}, \mathcal{B})$ as a public key of the trusted authority.

- $SKeyGen$: On input a system parameter param, a signer S randomly generates a private key sk_S and a public key pk_S as follows: first, choose a random integer $x \in \mathbb{Z}_p$. Let $\mathbb{X} = g^x; \mathbb{W} = \mathcal{A}^x; \mathbb{U} = \mathcal{B}^x$ denote a public key. Then, $SKeyGen$ sets $sk_S = x$ as a private key of the signer and $pk_S = (\mathbb{X}, \mathbb{W}, \mathbb{U})$ as a public key of the signer. Finally, $SKeyGen$ returns sk_S, pk_S.

- $CreGen$: Let A_V indicate a security level of a verifier, for example, $A_V =$"D". On input a system parameter param, the trusted authority's public key pk_{TA}, the trusted authority's private key sk_{TA} and a security level of a verifier $A_V = l$ that the verifier is allowed to obtain, a trusted authority TA randomly generates a set of credential strings $\mathbb{C} = (\mathbb{V}_1, ..., \mathbb{V}_l, \mathbb{R}_1, ..., \mathbb{R}_l)$, where i is an index of security level, as follows. TA randomly selects $\nu_1, ..., \nu_l \in \mathbb{Z}_p^*$ and computes each credential $\mathbb{V}_i = g^{c_i \cdot \nu_i}$; $\mathbb{R}_i = g^{(\mu_i \cdot \gamma_i - \mu_{i-1} \cdot \gamma_{i-1} - a \cdot c_i \cdot \nu_i)/b}$, and then returns $\mathbb{C} = (\mathbb{V}_1, ..., \mathbb{V}_l, \mathbb{R}_1, ..., \mathbb{R}_l)$ to the verifier as a credential for a security level assertion $A_V = l$. The verifier checks the validity of both $\mathbb{V}_i$ and $\mathbb{R}_i$ as follows: $\hat{e}(U_i, W_i) \stackrel{?}{=} \hat{e}(\mathcal{A}, \mathbb{V}_i)\hat{e}(\mathcal{B}, \mathbb{R}_i)\hat{e}(U_{i-1}, W_{i-1})$.

- *Sign*: Given param, pk_{TA}, sk_S, pk_S, $\mathbb{ML} =$ "$A_V \geq l$" and a message M, S

computes a multi-level controlled signature δ on a message M as follows.

$$\begin{aligned} r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = \mathbb{X}^r,\ \boldsymbol{\delta_3} = \mathbb{W}^r,\ \boldsymbol{\delta_4} = \mathbb{U}^r, \\ \Gamma &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML},\ \boldsymbol{\delta_5} = g^k, \\ \boldsymbol{\delta_6} &= H(\Gamma)^x, \\ \boldsymbol{\delta_7} &= h(\hat{e}(U_l, W_l)^{x \cdot r}) + h(M||\Gamma||\boldsymbol{\delta_5}), \\ \boldsymbol{\delta_8} &= k + \boldsymbol{\delta_7} \cdot x. \end{aligned}$$

The multi-level controlled signature on a message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. S publishes $M, \delta, \mathbb{ML}$.

- $Verify$: Let $\mathbb{C} = \mathbb{V}_1, ..., \mathbb{V}_l, \mathbb{R}_1, ..., \mathbb{R}_l$, be a set of credentials that a verifier possessed. Parse $\Gamma = \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}$. Given pk_S, pk_{TA}, pk_V, $\mathbb{C}$, $\mathbb{ML} =$ "$A_V \geq l$", δ and a message M, a verifier V checks whether

$$\begin{aligned} \hat{e}(\boldsymbol{\delta_1}, \mathbb{X}) &\stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, g),\ \hat{e}(\boldsymbol{\delta_3}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, \mathcal{A}), \\ \hat{e}(\boldsymbol{\delta_4}, g) &\stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, \mathcal{B}),\ \hat{e}(\boldsymbol{\delta_6}, g) \stackrel{?}{=} \hat{e}(H(\Gamma), \mathbb{X}), \\ \boldsymbol{\delta_7} &\stackrel{?}{=} h(M||\Gamma||\boldsymbol{\delta_5}) + h(\hat{e}(\boldsymbol{\delta_3}, \prod_{i=1}^{l} \mathbb{V}_i)\hat{e}(\boldsymbol{\delta_4}, \prod_{i=1}^{l} \mathbb{R}_i)), \\ g^{\boldsymbol{\delta_8}} &\stackrel{?}{=} \boldsymbol{\delta_5} \cdot \mathbb{X}^{\boldsymbol{\delta_7}} \end{aligned}$$

hold. If it does not hold, then V outputs $Reject$. Otherwise, it outputs $Accept$.

5.10 Security Analysis of the First MLCS scheme

5.10.1 Unforgeability

Theorem 5.7 *The above multi-level controlled signature scheme is existential unforgeable under an adaptive chosen message and credentials exposure attack if the CDH assumption holds in the random oracle model.*

Proof: Assume that there exists a forger $\mathcal{A}$ running the existential unforgeability game defined in Section 5.8.2. Then we will show that, by using $\mathcal{A}$, an adversary $\mathcal{F}$ can solve the CDH problem. We now begin with the construction of oracles. To begin with, $\mathcal{F}$ runs $Setup$ and $TKeyGen$ to obtain a system parameter param, a private key sk_{TA} and a public key of TA. Next, given g, g^x and g^y as an instance of the CDH problem, $\mathcal{F}$ sets $\mathbb{X} = g^x; \mathbb{W} = \mathbb{X}^a; \mathbb{U} = \mathbb{X}^b$ as the signer's public key pk_S.

$\mathcal{F}$ sets g^y as one of the answers for the hash query to the random oracle. Then, $\mathcal{F}$ constructs oracles as follows.

- $\boldsymbol{\mathcal{HO}}$ oracle: On input a string Γ, if it is a request for a hash value of $H(\Gamma)$, the $\boldsymbol{\mathcal{HO}}$ oracle randomly choose $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d}=1$ is $\frac{1}{q_H}$. If $\mathsf{d}=1$, set $H(\Gamma)=g^y$ and return $H(\Gamma)$. Otherwise, $l \xleftarrow{\$} \mathbb{Z}_p$; $H(\Gamma)=g^l$ and return $H(\Gamma)$. In the case of $h(\Gamma)$, $\boldsymbol{\mathcal{HO}}$ chooses $\iota \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(\Gamma)=\iota$. Then $\boldsymbol{\mathcal{HO}}$ keeps l and ι in the list and this list can be accessed only by $\mathcal{F}$.

- $\boldsymbol{\mathcal{VCO}}$ oracle: Given a private key sk_{TA} as input, $\boldsymbol{\mathcal{VCO}}$ runs $CreGen$ to generate the credential $\mathbb{C}$ for the security level assertion $A_V = l$ and then returns $\mathbb{C}$.

- $\boldsymbol{\mathcal{SSO}}$ oracle: Given $\mathbb{ML}=$ "$A_V \geq l$" and a message M as input, $\boldsymbol{\mathcal{SSO}}$ computes a multi-level controlled signature as follows.

$$\begin{aligned} r,k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1}=g^r,\ \boldsymbol{\delta_2}=\mathbb{X}^r,\ \boldsymbol{\delta_3}=\mathbb{W}^r, \\ \boldsymbol{\delta_4} &= \mathbb{U}^r,\ \Gamma=\boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}. \end{aligned}$$

  Before processing the next step, on accessing to the lists of l and ι, $\mathcal{F}$ checks whether $H(\Gamma) \stackrel{?}{=} g^y$. If it holds, it outputs $\perp$. Otherwise, $\mathcal{F}$ gives l to $\boldsymbol{\mathcal{SSO}}$. Next, $\mathcal{F}$ randomly selects $\iota' \xleftarrow{\$} \mathbb{Z}_p$; $K \xleftarrow{\$} \mathbb{G}_1$ and $\mathcal{F}$ adds $\iota', h(M||\Gamma||K)$ to the list. Then, $\mathcal{F}$ returns K to $\boldsymbol{\mathcal{SSO}}$. As a result, $\boldsymbol{\mathcal{SSO}}$ computes the rest of the signature as follows.

$$\begin{aligned} z &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_8}=z,\ \boldsymbol{\delta_6}=\mathbb{X}^l,\ \boldsymbol{\delta_5}=g^{\boldsymbol{\delta_8}}\mathbb{X}^{-\boldsymbol{\delta_7}}, \\ \boldsymbol{\delta_7} &= h(\hat{e}(\mathbb{X},W_l)^{\mu_l \cdot r})+h(M||\Gamma||K). \end{aligned}$$

  At the end of the process, on input of $\boldsymbol{\delta_5}$ from $\boldsymbol{\mathcal{SSO}}$, $\mathcal{F}$ updates $\iota', h(M||\Gamma||\boldsymbol{\delta_5})$ to the list. Hence, a multi-level controlled signature on message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. $\boldsymbol{\mathcal{SSO}}$ then responds with $M, \delta, \mathbb{ML}$.

Now, we begin the game by giving access to the above oracles to $\mathcal{A}$. Assume that $\mathcal{A}$ always makes a query for a string or a message to the $\boldsymbol{\mathcal{HO}}$ oracle before it outputs a potential forgery, denoted by $M^*, \delta^*, \mathbb{ML}^*$. After executing an adaptive strategy with the above oracles, $\mathcal{A}$ outputs a forgery δ^* on a message M^* with respect to $\mathbb{ML}^*$. $\mathcal{A}$ wins the game if a multi-level controlled signature δ^* on message M^* with respect to $\mathbb{ML}^*$ is valid and is not an output from the $\boldsymbol{\mathcal{SSO}}$ oracle.

We denote by ϵ the success probability $Succ^{CM\text{-}A}_{EUF\text{-}MLCS}(.)$ of $\mathcal{A}$ winning the game. Let e be the base of the natural logarithm. As we mentioned earlier, a query for a hash of a string or message to $\boldsymbol{\mathcal{HO}}$ is always issued before $\mathcal{A}$ issues a query for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle; hence, $q_H \geq q_S$. Now, we can analyse the success probability where $\mathcal{A}$ outputs a signature δ^* on message M^* with respect to $\mathbb{ML}^*$, where $\boldsymbol{\delta^*_6} = H(\Gamma)^x = (g^y)^x$, and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{SSO}}$ oracle.* The probability of this event $\Pr[E_1]$ is $(1 - \frac{1}{q_H})^{q_S}$. This is because $\mathcal{A}$ needs to have at least one query for $H(\Gamma)$ to output $\boldsymbol{\delta^*_6}$, which is part of a forgery. Since $q_H \geq q_S$, the upper bound for the $\boldsymbol{\mathcal{SSO}}$ oracle is then $q_H - 1$ and $\Pr[E_1] \geq (1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

- E_2: *$\mathcal{F}$ does not abort after $\mathcal{A}$ outputs δ^*.* $\mathcal{F}$ aborts the experiment after $\mathcal{A}$ outputs δ^* when only $H(\Gamma) \neq g^y$. Therefore, the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

To summarise the probability of success, $\mathcal{A}$ wins the above game and outputs a signature δ^* on a message M^*, where $H(\Gamma) = g^y$ and $\boldsymbol{\delta^*_6} = H(\Gamma)^x$, with a probability equal to $\Pr[Succ^{CM\text{-}A}_{EUF\text{-}MLCS}] \cdot \Pr[Succ^{CM\text{-}A}_{EUF\text{-}MLCS} | E_1 | E_2] \geq \epsilon(\frac{q_H}{e \cdot (q_H - 1)})^2$. From the above results, $\mathcal{F}$ outputs $\boldsymbol{\delta^*_6} = H(\Gamma)^x = g^{xy}$ as an answer to the CDH problem and the above probability shows that our multi-level controlled signature scheme is secure against existential unforgeability under an adaptive chosen message and credentials exposure attack if the success probability of solving the CDH problem is negligible.

5.10.2 Coalition-resistance

Theorem 5.8 *The above multi-level controlled signature scheme is existential coalition-resistant against an adaptively chosen message and chosen multi-level security policy distinguisher $\mathcal{A}^{CMP\text{-}A}_{CR\text{-}MLCS}$ if the DBDH assumption holds in the random oracle model.*

Proof: Assume that an adversary $\mathcal{A}$ runs the existential coalition-resistance game defined in Section 5.8.3 and successfully outputs a correct guess. We will then show that an adversary $\mathcal{F}$ can solve the DBDH problem by using $\mathcal{A}$ as a tool.

First, $\mathcal{F}$ constructs the oracles as follows: let n be the number of a security level. Given g, g^x, g^y, g^z and Z as an instance of the DBDH problem, $\mathcal{F}$ randomly selects

$\mu_1, ..., \mu_n, \gamma_1, ..., \gamma_n,\ a, b, c_1, ..., c_n \in \mathbb{Z}_p$, sets $pk_{TA} = (U_1 = g^{\mu_1}, ..., U_n = g^{\mu_n}, W_1 = g^{\gamma_1}, ..., W_n = g^{\gamma_n}, \mathcal{A} = g^a, \mathcal{B} = g^b)$ and sets $pk_S = (\mathbb{X} = g^x, \mathbb{W} = \mathbb{X}^a, \mathbb{U} = \mathbb{X}^b)$. $\mathcal{F}$ randomly selects $j \in \{1, ..., n\}$ and sets $U_j = g^y, W_j = g^z$. Assume that there exists an algorithm managing the list of each query and then such an algorithm will be omitted. Now, $\mathcal{F}$ constructs the oracles as follows.

- $\boldsymbol{\mathcal{HO}}$ oracle: In the case of $H(\Gamma)$, given a string Γ as input, $\boldsymbol{\mathcal{HO}}$ selects $l \xleftarrow{\$} \mathbb{Z}_p$ and sets $H(\Gamma) = g^l$. $\boldsymbol{\mathcal{HO}}$ returns $H(\Gamma)$. In the case of $h(\Gamma)$, $\boldsymbol{\mathcal{HO}}$ chooses $\iota \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(\Gamma) = \iota$. Then $\boldsymbol{\mathcal{HO}}$ keeps l and ι in the list and this list can be accessed only by $\mathcal{F}$.

- $\boldsymbol{\mathcal{VCO}}$ oracle: Given $A_V = l$ as input, if $l \geq j$ it then outputs $\perp$. Otherwise, $\boldsymbol{\mathcal{VCO}}$ runs $CreGen$ to generate $\mathbb{C} = (\mathbb{V}_1, ..., \mathbb{V}_l, \mathbb{R}_1, ..., \mathbb{R}_l)$. $\boldsymbol{\mathcal{VCO}}$ then returns $\mathbb{C}$.

- $\boldsymbol{\mathcal{SSO}}$ oracle: On input $\mathbb{ML}$ = "$A_V \geq l$" and a message M, $\boldsymbol{\mathcal{SSO}}$ computes a multi-level controlled signature as follows.

$$\begin{aligned} r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = \mathbb{X}^r,\ \boldsymbol{\delta_3} = \mathbb{W}^r, \\ \boldsymbol{\delta_4} &= \mathbb{U}^r,\ \Gamma = \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}. \end{aligned}$$

  Before processing the next steps, if $l = j$ then, with an access to the lists of l and ι, $\mathcal{F}$ randomly selects $\iota^* \xleftarrow{\$} \mathbb{Z}_p$ and sets $\iota^* = h(\hat{e}(\mathbb{X}, W_l)^{\mu_l \cdot r})$. $\mathcal{F}$ updates the list in $\boldsymbol{\mathcal{HO}}$ and returns l for $H(\Gamma)$ and ι^* for $h(\hat{e}(\mathbb{X}, W_l)^{\mu_l \cdot r})$ to $\boldsymbol{\mathcal{SSO}}$. Otherwise, $\mathcal{F}$ returns only l and, for $h(\hat{e}(\mathbb{X}, W_l)^{\mu_l \cdot r})$, $\boldsymbol{\mathcal{SSO}}$ can make a request for the hash value directly to $\boldsymbol{\mathcal{HO}}$. Next, $\mathcal{F}$ randomly selects $\iota' \xleftarrow{\$} \mathbb{Z}_p$; $K \xleftarrow{\$} \mathbb{G}_1$ and $\mathcal{F}$ adds $\iota', h(M||\Gamma||K)$ to the list. Then, $\mathcal{F}$ returns K to $\boldsymbol{\mathcal{SSO}}$. From the above outputs, $\boldsymbol{\mathcal{SSO}}$ computes the rest of the signature as follows.

$$\begin{aligned} \lambda &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_8} = \lambda,\ \boldsymbol{\delta_6} = \mathbb{X}^l,\ \boldsymbol{\delta_5} = g^{\boldsymbol{\delta_8}} \mathbb{X}^{-\boldsymbol{\delta_7}}, \\ \boldsymbol{\delta_7} &= h(\hat{e}(\mathbb{X}, W_l)^{\mu_l \cdot r}) + h(M||\Gamma||K). \end{aligned}$$

  At the end of the process, on input of $\boldsymbol{\delta_5}$ from $\boldsymbol{\mathcal{SSO}}$, $\mathcal{F}$ updates $\iota', h(M||\Gamma||\boldsymbol{\delta_5})$ to the list. Note that if $l = j$ then $\boldsymbol{\delta_7} = \iota^* + h(M||\Gamma||\boldsymbol{\delta_5})$. Hence, a multi-level controlled signature on message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. $\boldsymbol{\mathcal{SSO}}$ then responds with $M, \delta, \mathbb{ML}$.

To begin the experiment, $\mathcal{A}$ is given an access to the above oracles. Next, we run an experiment between $\mathcal{A}$ and $\mathcal{F}$ as modelled in Section 5.8.3 as follows.

1. **Phase** 1**:** With any adaptive strategy, $\mathcal{A}$ arbitrarily makes queries to the $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{VCO}}$ oracles. The oracles respond as outlined.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{ML}$. $\mathcal{F}$ aborts the game if

   1. Given $\mathbb{ML}^*$ and M^* as input, $\mathcal{A}$ issued a request for a multi-level controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

   2. $\mathcal{A}$ has a credential that is equal to or higher than the security level assigned in the multi-level security policy $\mathbb{ML}$.

   Otherwise, $\mathcal{F}$ computes a response as follows.

$$\begin{aligned} r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta^*_1} = g^r,\ \boldsymbol{\delta^*_2} = \mathbb{X}^r,\ \boldsymbol{\delta^*_3} = \mathbb{W}^r, \\ \boldsymbol{\delta^*_4} &= \mathbb{U}^r,\ \Gamma = \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}. \end{aligned}$$

   Before processing the next steps, if $l = j$ then, with access to the lists of l and ι, $\mathcal{F}$ randomly selects $\iota^* \xleftarrow{\$} \mathbb{Z}_p$ and sets $\iota^* = h(Z^r)$. After that, $\mathcal{F}$ updates the list in the $\boldsymbol{\mathcal{HO}}$ oracle. Next, $\mathcal{F}$ randomly selects $\iota' \xleftarrow{\$} \mathbb{Z}_p$; $K \xleftarrow{\$} \mathbb{G}_1$ and $\mathcal{F}$ adds a pair $\iota', h(M||\Gamma||K)$ to the list. $\mathcal{F}$ computes the rest of the signature as follows.

$$\begin{aligned} \lambda &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta^*_8} = \lambda,\ \boldsymbol{\delta^*_6} = \mathbb{X}^l, \\ \boldsymbol{\delta^*_7} &= h(Z^r) + h(M||\Gamma||K),\ \boldsymbol{\delta^*_5} = g^{\boldsymbol{\delta_8}}\mathbb{X}^{-\boldsymbol{\delta_7}}. \end{aligned}$$

   At the end of the process, on input of $\boldsymbol{\delta^*_5}$ from $\boldsymbol{\mathcal{SSO}}$, $\mathcal{F}$ updates $\iota', h(M||\Gamma||\boldsymbol{\delta^*_5})$ to the list. Note that if $l \neq j$ then $\boldsymbol{\delta^*_7} = h(\hat{e}(\mathbb{X}, W_l)^{\mu_l \cdot r}) + h(M||\Gamma||\boldsymbol{\delta^*_5})$. Hence, a multi-level controlled signature on message M is $\delta^* = (\boldsymbol{\delta^*_1}, \boldsymbol{\delta^*_2}, \boldsymbol{\delta^*_3}, \boldsymbol{\delta^*_4}, \boldsymbol{\delta^*_5}, \boldsymbol{\delta^*_6}, \boldsymbol{\delta^*_7}, \boldsymbol{\delta^*_8})$. $\mathcal{F}$ then responds with $M^*, \delta^*, \mathbb{ML}^*$ to $\mathcal{A}$.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can go back to *Phase* 1 or *Challenge* as many times as it requests. However, $\mathcal{F}$ will abort the game if

   1. Given $\mathbb{ML}^*$ and M^* as input, $\mathcal{A}$ issues a request for a multi-level controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

2. $\mathcal{A}$ has a credential that is equal to or higher than the security level assigned in the multi-level security policy $\mathbb{ML}^*$.

4. **Guessing:** On the valid challenge $M^*, \mathbb{ML}^*, \delta^*$, $\mathcal{A}$ finally outputs a guess b'.

Let $Succ^{CM\text{-}A}_{CR\text{-}MLCS} = \epsilon$ be the probability of $\mathcal{A}$ winning the above game. Let q be a polynomial upper bound of queries that $\mathcal{A}$ issues to the $\boldsymbol{\mathcal{HO}}$ oracle. Note that $q \geq q_H$ and $q << p^2$. Since only $\mathcal{F}$ and $\boldsymbol{\mathcal{SSO}}$ access $\boldsymbol{\mathcal{HO}}$ before it outputs a response, we can thus conclude that $q_H \geq q_S$. Therefore, we can analyse the probability that $\mathcal{A}$'s guess is correct and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{VCO}}$ oracle.* Let q_{VC} be the highest security level that $\mathcal{A}$ issues to the $\boldsymbol{\mathcal{VCO}}$ oracle. q_{VC} is not a number of queries that $\mathcal{A}$ makes a query request to the $\boldsymbol{\mathcal{VCO}}$ oracle. Since $\mathcal{A}$ can make just one query for the security level $A_v = n - 1$, $\mathcal{A}$ can use this set of credentials to verify signatures with the entire security level except the security level n. Note that j is a random integer chosen at the beginning of the game and n is the upper bound of the security level. The fact is that $\mathcal{A}$ can make a request for credentials up to the security level $q_{VC} = n - 1$ to the $\boldsymbol{\mathcal{VCO}}$ oracle and the value of j is in range of $\{1, ..., n\}$. However, if $q_{VC} \geq j$, then the $\boldsymbol{\mathcal{VCO}}$ will always terminate the experiment. Otherwise, $q_{VC} < j$, then the $\boldsymbol{\mathcal{VCO}}$ will not terminated the experiment. To pick up q_{VC} and j randomly, the probability that $\mathcal{A}$ chooses q_{VC} is $\frac{1}{n}$ and the probability that $\mathcal{F}$ choose j is $\frac{1}{n}$ Therefore, the probability of this event is $\frac{1}{n^2}$.

- E_2: *$\mathcal{F}$ does not abort after* Phase 1 *and* Phase 2. Since we have assumed that $\mathcal{A}$ follows the experiment and outputs a guess with a valid challenge $(M^*, \mathbb{ML}^*, \delta^*)$, then the probability of this event is 1.

The probability that $\mathcal{A}$ wins the above game and outputs a correct guess $b' = b$ is $\Pr[Succ^{CM\text{-}A}_{CR\text{-}MLCS}] \cdot \Pr[Succ^{CM\text{-}A}_{CR\text{-}MLCS}|E_1|E_2] \geq \epsilon \cdot \frac{1}{n^2}$.

However, there is a situation when $\mathcal{A}$'s guess in the game is not the correct guess for the DBDH problem, and this is where $\mathbb{ML}^* \neq$ "$A_V \geq j$". The probability of this event is $\frac{1}{n}$. Let ϵ' be an advantage in solving the DBDH problem. To conclude, the probability that $\mathcal{F}$ will output a correct guess for the DBDH problem by using $\mathcal{A}$ is $\epsilon' \geq \epsilon \cdot \frac{1}{n^2} \cdot \frac{1}{n} = \epsilon \cdot \frac{1}{n^3}$. Hence, the probability that $\mathcal{A}$ can break the existential

[2]the notation $q << p$ means that q is significanly smaller than p

coalition-resistance property of a MLCS scheme secure against adaptively chosen message and chosen multi-level security policy attack is $\epsilon \leq n^3\epsilon'$. Since $n << q_H << q$, the analysis of the probability above shows that the success probability of breaking the existential coalition-resistance of our MLCS scheme is non-negligible if the probability of breaking the DBDH problem is non-negligible.

5.11 The Second Proposed MLCS Scheme

In this section, we present our second construction of MLCS schemes. The scheme is described as follows.

- *Setup*: Given a security parameter ℓ as input, a trusted third party randomly selects a prime $p = poly(1^\ell)$. Choose a random generator $g \in \mathbb{G}_1$ and a bilinear mapping function $\hat{e}$. Select two hash functions $H(.)$ and $h(.)$. $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a collision-resistant hash function that takes strings as input and outputs an element in $\mathbb{G}_1$. $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a collision-resistant hash function that takes strings as input and outputs an element in $\mathbb{Z}_p^*$. Let $\mathsf{param} = (p, \hat{e}, g, H, h)$ denote the system parameter. Then, *Setup* returns param.

- $TKeyGen$: Let n be a number of security levels. Given a system parameter param as input, a trusted authority TA randomly generates a private key sk_{TA} and a public key pk_{TA} for each security level as follows: select random integers $\mu, a, b, w_1, ..., w_n \in \mathbb{Z}_p$. Let $pk_{TA} = (U = g^\mu, \mathcal{A} = g^a, \mathcal{B} = g^b, W_1 = g^{w_1}, ..., W_n = g^{w_n}$ denote a public key. Then, $TKeyGen$ returns $sk_{TA} = (\mu, a, b, w_1, ..., w_n)$ as a private key of the trusted authority and $pk_{TA} = (\mathcal{A}, \mathcal{B}, U, W_1, ..., W_n,)$ as a public key of the trusted authority.

- $SKeyGen$: Given a system parameter param, a signer S randomly generates a private key sk_S and a public key pk_S as follows: choose a random integer $x \in \mathbb{Z}_p$. Let us set $pk_S = (\mathbb{X} = g^x, \mathbb{U} = U^x, \mathbb{W}_1 = W_1^x, ..., \mathbb{W}_n = W_n^x)$ to denote a public key. Then, $SKeyGen$ returns $sk_S = x$ as a private key of the signer and $pk_S = (\mathbb{X}, \mathbb{U}, \mathbb{W}_1, ..., \mathbb{W}_n)$ as a public key of the signer.

- $CreGen$: Let A_V indicates a security level of a verifier. Given a system parameter param, the trusted authority's public key pk_{TA}, the trusted authority's private key sk_{TA} and a security level of a verifier $A_V = l$ that the verifier is

allowed to obtain, a trusted authority TA randomly generates a set of credential strings $\mathbb{C} = (\mathbb{V}, \mathbb{R})$ as follows. TA randomly selects $s \in \mathbb{Z}_p^*$ and computes each credential $\mathbb{V} = g^s$; $\mathbb{R} = g^{((a \cdot b - s \cdot \mu)/w_l)}$ and then returns $\mathbb{C} = (\mathbb{V}, \mathbb{R})$ to the verifier as a credential for a security level assertion $A_V = l$. The verifier checks the validity of $\mathbb{V}, \mathbb{R}$ as follows: $\hat{e}(\mathcal{A}, \mathcal{B}) \stackrel{?}{=} \hat{e}(U, \mathbb{V})\hat{e}(W_l, \mathbb{R})$.

- *Sign*: Given param, pk_{TA}, sk_S, pk_S, $\mathbb{ML}$ = "$A_V \geq l$" and a message M, S computes a multi-level controlled signature δ on a message M as follows.

$$\begin{aligned} r, k &\stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = \mathbb{X}^r, \\ \boldsymbol{\delta_3} &= \{\boldsymbol{\delta_{3,l}} = \mathbb{W}_l^r, ..., \boldsymbol{\delta_{3,n}} = \mathbb{W}_n^r\},\ \boldsymbol{\delta_4} = \mathbb{U}^r, \\ \Gamma &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}, \\ \boldsymbol{\delta_5} &= g^k,\ \boldsymbol{\delta_6} = H(\Gamma)^x,\ \boldsymbol{\delta_8} = k + \boldsymbol{\delta_7} \cdot x, \\ \boldsymbol{\delta_7} &= h(\hat{e}(\mathcal{A}, \mathcal{B})^{x \cdot r}) + h(M||\Gamma||\boldsymbol{\delta_5}). \end{aligned}$$

  The multi-level controlled signature on a message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. S publishes $M, \delta, \mathbb{ML}$.

- *Verify*: Let $\mathbb{C} = (\mathbb{V}, \mathbb{R})$ be a credential that a verifier possesses for security level "$A_V = t$", where $l \leq t \leq n$. Let parse $\Gamma = \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||\ pk_S||pk_{TA}||\mathbb{ML}$. Given pk_S, pk_{TA}, pk_V, $\mathbb{C}$, $\mathbb{ML}$ = "$A_V \geq l$", δ and a message M as input, a verifier V checks whether, for $i = l$ to n, $\hat{e}(\boldsymbol{\delta_{3,i}}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, W_i)$, and then checks whether

$$\begin{gathered} \hat{e}(\boldsymbol{\delta_1}, \mathbb{X}) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, g),\ \hat{e}(\boldsymbol{\delta_4}, g) \stackrel{?}{=} \hat{e}(\boldsymbol{\delta_2}, U), \\ \hat{e}(\boldsymbol{\delta_6}, g) \stackrel{?}{=} \hat{e}(H(\Gamma), \mathbb{X}),\ g^{\boldsymbol{\delta_8}} \stackrel{?}{=} \boldsymbol{\delta_5} \cdot \mathbb{X}^{\boldsymbol{\delta_7}}, \\ \boldsymbol{\delta_7} \stackrel{?}{=} h(M||\Gamma||\boldsymbol{\delta_5}) + h(\hat{e}(\boldsymbol{\delta_4}, \mathbb{V})\hat{e}(\boldsymbol{\delta_{3,t}}, \mathbb{R})) \end{gathered}$$

  hold or not. If it does not hold, then V outputs *Reject*. Otherwise, it outputs *Accept*.

5.12 Security Analysis of the Second MLCS Scheme

5.12.1 Unforgeability

Theorem 5.9 *Our second multi-level controlled signature scheme is existential unforgeability under an adaptive chosen message and credentials exposure attack if the CDH assumption holds in the random oracle model.*

Proof: Assume that there exists a forger $\mathcal{A}$ running the existential unforgeability game defined in Section 5.8.2. Then we will show that, by using $\mathcal{A}$, an adversary $\mathcal{F}$ can solve the CDH problem. We now begin with the construction of oracles. To begin with, $\mathcal{F}$ runs *Setup* and *TKeyGen* to obtain a system parameter param, a private key sk_{TA} and a public key of TA. Next, given g, g^x and g^y as an instance of the CDH problem, $\mathcal{F}$ sets $\mathbb{X} = g^x; \mathbb{U} = \mathbb{X}^{\mu}; \mathbb{W}_1 = \mathbb{X}^{w_1}; ...; \mathbb{W}_n = \mathbb{X}^{w_n}$ as the signer's public key pk_S. $\mathcal{F}$ sets g^y as one of the answers for the hash query to the random oracle. Then, $\mathcal{F}$ constructs oracles as follows.

$\mathcal{HO}$ oracle: Given a string Γ as input, if it is a request for a hash value of $H(\Gamma)$, the $\mathcal{HO}$ oracle randomly choose $\mathsf{d} \xleftarrow{\$} \{0,1\}$ such that the probability of $\mathsf{d} = 1$ is $\frac{1}{q_H}$. If $\mathsf{d} = 1$, set $H(\Gamma) = g^y$ and return $H(\Gamma)$. Otherwise, $l \xleftarrow{\$} \mathbb{Z}_p$; $H(\Gamma) = g^l$ and return $H(\Gamma)$. In the case of $h(\Gamma)$, $\mathcal{HO}$ chooses $\iota \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(\Gamma) = \iota$. Then $\mathcal{HO}$ keeps l and ι in the list and this list can be accessed only by $\mathcal{F}$. Note that $\mathcal{HO}$ manages the duplicated hash value of the list by repeating the process such that the output of $\mathcal{HO}$ behaves like a result from a random oracle.

$\mathcal{VCO}$ oracle: On input a private key sk_{TA}, $\mathcal{VCO}$ runs *CreGen* to generate the credential $\mathbb{C}$ for the security level assertion $A_V = l$ and then returns $\mathbb{C}$.

$\mathcal{SSO}$ oracle: On input $\mathbb{ML}$ = "$A_V \geq l$" and a message M, $\mathcal{SSO}$ computes a multi-level controlled signature as follows.

$$\begin{aligned} r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = \mathbb{X}^r,\ \boldsymbol{\delta_3} = (\mathbb{W}_l^r, ..., \mathbb{W}_n^r),\ \boldsymbol{\delta_4} = \mathbb{U}^r, \\ \Gamma &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}. \end{aligned}$$

Before processing the next step, on accessing the lists of l and ι, $\mathcal{F}$ checks whether $H(\Gamma) \stackrel{?}{=} g^y$). If it holds, $\mathcal{F}$ outputs $\perp$. Otherwise, $\mathcal{F}$ gives l to $\mathcal{SSO}$. Next, $\mathcal{F}$ randomly selects $\iota' \xleftarrow{\$} \mathbb{Z}_p$; $K \xleftarrow{\$} \mathbb{G}_1$ and $\mathcal{F}$ adds $\iota', h(M||\Gamma||K)$ to the list. Then, $\mathcal{F}$ returns K to $\mathcal{SSO}$. As a result, $\mathcal{SSO}$ computes the rest of the signature as follows.

$$z \xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_8} = z,\ \boldsymbol{\delta_6} = \mathbb{X}^l,\ \boldsymbol{\delta_7} = h(\hat{e}(\mathbb{X}, \mathcal{A})^{b \cdot r}) + h(M||\Gamma||K),\ \boldsymbol{\delta_5} = g^{\boldsymbol{\delta_8}} \mathbb{X}^{-\boldsymbol{\delta_7}}.$$

At the end of the process, on input of $\boldsymbol{\delta_5}$ from $\mathcal{SSO}$, $\mathcal{F}$ updates $\iota', h(M||\Gamma||\boldsymbol{\delta_5})$ to the list. Hence, a multi-level controlled signature on message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. $\mathcal{SSO}$ then responds with $M, \delta, \mathbb{ML}$.

Now, we begin the game by giving access to the above oracles to $\mathcal{A}$. Assume that $\mathcal{A}$ always makes a query for a string or a message to the $\boldsymbol{\mathcal{HO}}$ oracle before it outputs a potential forgery, denoted by $M^*, \delta^*, \mathbb{ML}^*$. After executing an adaptive strategy with the above oracles, $\mathcal{A}$ outputs a forgery δ^* on a message M^* with respect to $\mathbb{ML}^*$. $\mathcal{A}$ wins the game if a multi-level controlled signature δ^* on message M^* with respect to $\mathbb{ML}^*$ is valid and is not an output from the $\boldsymbol{\mathcal{SSO}}$ oracle.

We denote by ϵ the probability of success $Succ^{CM\text{-}A}_{EUF\text{-}MLCS}(.)$ of $\mathcal{A}$ winning the game. Let e be the base of the natural logarithm. As we mentioned earlier, a query for a hash of a string or message to $\boldsymbol{\mathcal{HO}}$ is always issued before $\mathcal{A}$ issues a query for a signature to the $\boldsymbol{\mathcal{SSO}}$ oracle; hence, $q_H \geq q_S$. Now, we can analyse the probability of success where $\mathcal{A}$ outputs a signature δ^* on message M^* with respect to $\mathbb{ML}^*$, where $\boldsymbol{\delta^*_6} = H(\Gamma)^x = (g^y)^x$, and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{SSO}}$ oracle.* The probability of this event $\Pr[E_1]$ is $(1 - \frac{1}{q_H})^{q_S}$. This is because $\mathcal{A}$ needs to have at least one query for $H(\Gamma)$ to output $\boldsymbol{\delta^*_6}$, which is part of a forgery. Since $q_H \geq q_S$, the upper bound for the $\boldsymbol{\mathcal{SSO}}$ oracle is then $q_H - 1$ and $\Pr[E_1] \geq (1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

- E_2: *$\mathcal{F}$ does not abort after $\mathcal{A}$ output δ^*.* $\mathcal{F}$ aborts the experiment after $\mathcal{A}$ output δ^* when only $H(\Gamma) \neq g^y$. Therefore, the probability of this event is greater than $(1 - \frac{1}{q_H})^{q_H - 1} \approx \frac{q_H}{e \cdot (q_H - 1)}$.

To summarise the probability, $\mathcal{A}$ wins the above game and outputs a signature δ^* on a message M^*, where $H(\Gamma) = g^y$ and $\boldsymbol{\delta^*_6} = H(\Gamma)^x$, with a probability equal to $\Pr[Succ^{CM\text{-}A}_{EUF\text{-}MLCS}] \cdot \Pr[Succ^{CM\text{-}A}_{EUF\text{-}MLCS} | E_1 | E_2] \geq \epsilon(\frac{q_H}{e \cdot (q_H - 1)})^2$. From these results, $\mathcal{F}$ outputs $\boldsymbol{\delta^*_6} = H(\Gamma)^x = g^{xy}$ as an answer to the CDH problem and the above probability shows that our multi-level controlled signature scheme is secure against existential unforgeability under an adaptive chosen message and credentials exposure attack if the success probability of solving the CDH problem is negligible.

5.12.2 Coalition-resistance

Theorem 5.10 *Our second multi-level controlled signature scheme is existential coalition-resistant against an adaptively chosen message and chosen multi-level security policy distinguisher $\mathcal{A}^{CMP\text{-}A}_{CR\text{-}MLCS}$ if the DBDH assumption holds in the random oracle model.*

Proof: Assume that an adversary $\mathcal{A}$ runs the existential coalition-resistance game defined in Section 5.8.3 and successfully outputs a correct guess. We will then show that an adversary $\mathcal{F}$ can solve the DBDH problem by using $\mathcal{A}$ as a tool.

First, $\mathcal{F}$ constructs the oracles as follows: let n be the number of a security level. Given g, g^a, g^b, g^c and Z as an instance of the DBDH problem, $\mathcal{F}$ randomly selects $j \in \{1, ..., n\}$ and $w_1, ..., w_n, x, \mu, k_1, k_2 \in \mathbb{Z}_p$. Next, $\mathcal{F}$ sets $pk_{TA} = (U = g^{\mu}, W_1 = g^{a \cdot w_1}, ..., W_{j-1} = g^{a \cdot w_{j-1}}, W_j = g^{w_j}, ..., W_n = g^{w_n}, \mathcal{A} = g^a, \mathcal{B} = g^b)$ and sets $pk_S = (\mathbb{X} = g^x, \mathbb{U} = U^x, \mathbb{W}_1 = W_1^a, ..., \mathbb{W}_n = W_n^x,)$.

Assume that there exists an algorithm managing the list of each query and then such an algorithm will be omitted. Now, $\mathcal{F}$ constructs the oracles as follows.

$\mathcal{HO}$ oracle: In the case of $H(\Gamma)$, given a string Γ as input, $\mathcal{HO}$ selects $l \xleftarrow{\$} \mathbb{Z}_p$ and sets $H(\Gamma) = g^l$. $\mathcal{HO}$ returns $H(\Gamma)$. In the case of $h(\Gamma)$, $\mathcal{HO}$ chooses $\iota \xleftarrow{\$} \mathbb{Z}_p$ and then returns $h(\Gamma) = \iota$. Then $\mathcal{HO}$ keeps l and ι in the list and this list can be accessed only by $\mathcal{F}$.

$\mathcal{VCO}$ oracle: Given $A_V = l$ as input, if $l \geq j$ it then output $\perp$. Otherwise, $\mathcal{VCO}$ first selects random integer s. $\mathcal{VCO}$ then computes $\mathbb{C} = (\mathbb{V} = g^{a \cdot s}, \mathbb{R} = g^{(b - s \cdot \mu)/w_l})$. $\mathcal{VCO}$ then returns $\mathbb{C}$.

$\mathcal{SSO}$ oracle: Given $\mathbb{ML}$ = "$A_V \geq l$" and a message M as input, $\mathcal{SSO}$ computes a multi-level controlled signature as follows.

$$
\begin{aligned}
r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta_1} = g^r,\ \boldsymbol{\delta_2} = \mathbb{X}^r,\ \boldsymbol{\delta_3} = (\mathbb{W}_l^r, ..., \mathbb{W}_n^r),\ \boldsymbol{\delta_4} = \mathbb{U}^r, \\
\Gamma &= \boldsymbol{\delta_1} || \boldsymbol{\delta_2} || \boldsymbol{\delta_3} || \boldsymbol{\delta_4} || pk_S || pk_{TA} || \mathbb{ML}, \\
\boldsymbol{\delta_5} &= g^k,\ \boldsymbol{\delta_6} = H(\Gamma)^x,\ \boldsymbol{\delta_7} = h(\hat{e}(\mathcal{A}, \mathcal{B})^{x \cdot r}) + h(M || \Gamma || \boldsymbol{\delta_5}),\ \boldsymbol{\delta_8} = k + \boldsymbol{\delta_7} \cdot x.
\end{aligned}
$$

Hence, a multi-level controlled signature on message M is $\delta = (\boldsymbol{\delta_1}, \boldsymbol{\delta_2}, \boldsymbol{\delta_3}, \boldsymbol{\delta_4}, \boldsymbol{\delta_5}, \boldsymbol{\delta_6}, \boldsymbol{\delta_7}, \boldsymbol{\delta_8})$. $\mathcal{SSO}$ then responds with $M, \delta, \mathbb{ML}$.

To begin the experiment, $\mathcal{A}$ is given an access to the above oracles. Next, we run an experiment between $\mathcal{A}$ and $\mathcal{F}$ as modelled in Section 5.8.3 as follows.

1. **Phase 1:** With any adaptive strategy, $\mathcal{A}$ arbitrarily makes queries to the $\mathcal{SSO}$ and $\mathcal{VCO}$ oracles. The oracles respond as outlined the above.

2. **Challenge:** At the end of the first phase, $\mathcal{A}$ decides to challenge and then outputs M^* and $\mathbb{ML}$. $\mathcal{F}$ aborts the game if

1. Given $\mathbb{ML}^*$ and M^* as input, $\mathcal{A}$ issues a request for a multi-level controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

2. $\mathcal{A}$ has a credential that is equal to or higher than the security level assigned in the multi-level security policy $\mathbb{ML}^*$.

Otherwise, $\mathcal{F}$ computes a response as follows: if A_V in the multi-level security policy $\mathbb{ML}^*$ is more than or equal to j, then $\mathcal{F}$ computes

$$\begin{aligned} k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta^*_1} = g^c,\ \boldsymbol{\delta^*_2} = g^{c \cdot x},\ \boldsymbol{\delta^*_3} = g^{c \cdot x \cdot w_l}, ..., g^{c \cdot x \cdot w_n},\ \boldsymbol{\delta^*_4} = g^{c \cdot x \cdot \mu}, \\ \Gamma &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}, \\ \boldsymbol{\delta^*_5} &= g^k,\ \boldsymbol{\delta^*_6} = H(\Gamma)^x,\ \boldsymbol{\delta^*_7} = h(Z^x) + h(M^*||\Gamma||\boldsymbol{\delta_5}),\ \boldsymbol{\delta^*_8} = k + \boldsymbol{\delta_7} \cdot x. \end{aligned}$$

Otherwise, $A_V > j$ then computes

$$\begin{aligned} r, k &\xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{\delta^*_1} = g^r,\ \boldsymbol{\delta^*_2} = \mathbb{X}^r,\ \boldsymbol{\delta^*_3} = (\mathbb{W}^r_l, ..., \mathbb{W}^r_n),\ \boldsymbol{\delta^*_4} = \mathbb{U}^r, \\ \Gamma &= \boldsymbol{\delta_1}||\boldsymbol{\delta_2}||\boldsymbol{\delta_3}||\boldsymbol{\delta_4}||pk_S||pk_{TA}||\mathbb{ML}, \\ \boldsymbol{\delta^*_5} &= g^k,\ \boldsymbol{\delta^*_6} = H(\Gamma)^x,\ \boldsymbol{\delta^*_7} = h(\hat{e}(\mathcal{A}, \mathcal{B})^{x \cdot r}) + h(M^*||\Gamma||\boldsymbol{\delta_5}), \\ \boldsymbol{\delta^*_8} &= k + \boldsymbol{\delta_7} \cdot x. \end{aligned}$$

Hence, a multi-level controlled signature on message M is $\delta^* = (\boldsymbol{\delta^*_1}, \boldsymbol{\delta^*_2}, \boldsymbol{\delta^*_3}, \boldsymbol{\delta^*_4}, \boldsymbol{\delta^*_5}, \boldsymbol{\delta^*_6}, \boldsymbol{\delta^*_7}, \boldsymbol{\delta^*_8})$. $\mathcal{F}$ then responds with $M^*, \delta^*, \mathbb{ML}^*$ to $\mathcal{A}$.

3. **Phase** 2**:** In this phase, $\mathcal{A}$ can go back to *Phase* 1 or *Challenge* as many times as it requests. However, $\mathcal{F}$ will abort the game if

1. Given $\mathbb{ML}^*$ and M^* as input, $\mathcal{A}$ issues a request for a multi-level controlled signature to the $\boldsymbol{\mathcal{SSO}}$ oracle.

2. $\mathcal{A}$ has a credential that is equal to or higher than the security level assigned in the multi-level security policy $\mathbb{ML}^*$.

4. **Guessing:** On the valid challenge $M^*, \mathbb{ML}^*, \delta^*$, $\mathcal{A}$ finally outputs a guess b'.

Let $Succ^{CM\text{-}A}_{CR\text{-}MLCS} = \epsilon$ be the probability of $\mathcal{A}$ winning the above game. Let q be a polynomial upper bound of queries that $\mathcal{A}$ issues to the $\boldsymbol{\mathcal{HO}}$ oracle. Note that $q \geq q_H$ and $q << p$. Since only $\mathcal{F}$ and $\boldsymbol{\mathcal{SSO}}$ access $\boldsymbol{\mathcal{HO}}$ before it outputs a response, we can thus conclude that $q_H \geq q_S$. Therefore, we can analyse the probability that $\mathcal{A}$'s guess is correct and wins the above game as follows.

- E_1: *$\mathcal{F}$ does not abort during the issuing of queries to the $\boldsymbol{\mathcal{VCO}}$ oracle.*

  To simplify, we denote q_{VC} as the security level that $\mathcal{A}$ issues to the $\boldsymbol{\mathcal{VCO}}$ oracle. Note that q_{VC} is not a number of queries that make to the $\boldsymbol{\mathcal{VCO}}$ oracle. This is because $\mathcal{A}$ can issue one request for security level $A_v = n - 1$ and then use it to verify signatures with the entire security level except the security level n. Let q_{vc} be the actual number of queries that $\mathcal{A}$ issued to the $\boldsymbol{\mathcal{VCO}}$ oracle and the upper bound of queries is $q_{VC} < n$. There are two cases for this event:

  1. In the case where $q_{VC} \geq j$, the probability of this event is 0, since $\mathcal{F}$ will always abort the simulation.
  2. In the case where $q_{VC} < j$, the probability of this event is 1.

  The fact is that $\mathcal{A}$ can make a request for credentials up to security level $q_{VC} = n - 1$ to the $\boldsymbol{\mathcal{VCO}}$ oracle and the value of j is randomly chosen from 1 to n. Hence, the probability of this event is $\frac{1}{n^2}$, since it is only concerned about the random integer $j \in \{1, ...n\}$ selected by $\mathcal{F}$ at the beginning of the game and the highest security level $q_{VC} \in \{0, ...n-1\}$ selected by $\mathcal{A}$. Note that the probability for this event is not tied to the number of queries according to the explanation above.

- E_2: *$\mathcal{F}$ does not abort after* Phase 1 *and* Phase 2. Since we have assumed that $\mathcal{A}$ follows the experiment and outputs a guess with a valid challenge $(M^*, \mathbb{ML}^*, \delta^*)$, then the probability of this event is 1.

The probability that $\mathcal{A}$ wins the above game and outputs a correct guess $b' = b$ is $\Pr[Succ^{CM\text{-}A}_{CR\text{-}MLCS}] \cdot \Pr[Succ^{CM\text{-}A}_{CR\text{-}MLCS}|E_1|E_2] \geq \epsilon\frac{1}{n^2}$.

Let ϵ' be an advantage to solve the DBDH problem. From the above game, $\mathcal{F}$ outputs a guess for the DBDH problem with $\mathcal{A}$'s guess. Note that $\mathcal{A}$ can choose a challenge multi-level security policy $\mathbb{ML}^*$, where $\mathcal{A}$ does not have the credentials for that security level or above.

Hence, there is an event when $\mathcal{A}$'s guess in the game is not the correct guess for the DBDH problem, and this is where A^*_V is lower than j. The probability of this event does not happening is $\frac{j}{n}$. To conclude, the probability that $\mathcal{F}$ will output a correct guess for the DBDH problem by using $\mathcal{A}$ is $\epsilon' \geq \epsilon \cdot \frac{1}{n^2} \cdot \frac{j}{n}$. Hence, the probability that $\mathcal{A}$ can break the existential coalition-resistance probability of our MLCS scheme

secure against an adaptively chosen message and chosen multi-level security policy attack is $\epsilon \leq \frac{n^3}{j}\epsilon'$. Since $n << q_H << q$, the analysis of the probabilities above shows that probability of breaking the existential coalition-resistance property of our MLCS scheme is non-negligible if the probability of breaking the DBDH problem is non-negligible.

5.13 Conclusion

In this chapter, we have introduced the notion of policy-controlled signatures and their applications, which are universal policy-controlled signatures and multi-level controlled signatures. The notion of policy-controlled signatures allows a signer to control his signature's verification by an assigned policy. Only a verifier that satisfies the assigned policy can verify the authenticity of the message. Hence, in order to verify the signature on the message, the verifier has to have valid credentials satisfying the policy designed by the signer. We have presented a definition of PCS schemes and its security model. We have also provided a concrete construction that is secure in our model.

The notion of universal policy-controlled signatures allows a policy signer to enclose an assigned policy on a signed message. Hence, a policy signer can claim possession of a signer's signature on a message to verifiers that satisfy the policy appointed by the policy signer. We have presented a definition of UPCS schemes and their security model. A concrete construction that is secure in our model has also presented.

The notion of multi-level controlled signatures allows a signer to enclose a security level on the signature. Only verifiers that possess a credential for the level of security above the level assigned by the signer are able to test the authenticity of the message. We have presented a definition of MLCS schemes and their security model. Two concrete schemes and their proof of security have been presented. To conclude, we are the first researchers to propose these primitive schemes and we provided the concrete schemes together with their security analysis. These cryptographic primitives are difficult to construct. In particular, they have to ensure that any coalition of unauthorised verifiers is not able to verify the authenticity of any signature.

Chapter 6

Fair Multi-Signature Scheme

A new primitive algorithm called a "fair multi-signature" is described in this chapter. This notion is an extension of a multi-signature scheme where a group of signers together fairly generate a signature. In other words, every signer should be able to output a multi-signature if the protocol is complete. Otherwise, none of the signers can output a multi-signature. Moreover, we only require that the third party is semi-trusted, since even if malicious, he/she cannot output a multi-signature. Hence, this notion further strengthens the notion of multi-signature schemes.

6.1 Introduction

Many businesses are unable to hold a meeting with their customers in person due to their different locations and travel cost. Thanks to the virtual conference technology, there is an excellent platform for entrepreneurs to manage their business with distant clients in a way that is fast, easy and cost effective. Apart from virtual conferences, digital signatures play an important role in virtual meetings. A digital document with its digital signature that can be delivered through the Internet network is replacing paperwork of the traditional meeting.

A multi-signature (MS) scheme is needed in the above scenario, when a group of parties who engage in an interactive protocol want to generate a joint signature on an agreement (a message) m. There are several advantages in having a multi-signature rather than a standard digital signature, such as the size of a multi-signature being constant and short, and the fact that the verification algorithm is as efficient as a standard signature. Therefore, for a practical application such as signing an online contract, a multi-signature scheme allows a group of parties to co-sign an agreed document.

Consider the following scenario. Alice, Bob and Charlie agree to sign some agreements jointly. Let us assume that Charlie and Bob comply correctly and completely according to the signing protocol of the multi-signature scheme, so that their partial signatures are generated and distributed to every signer. Now, after Alice obtains the partial signatures from Charlie and Bob, if she does not complete the protocol or, in other words, she does not send her partial signature to Charlie and Bob, then it becomes unfair to Charlie and Bob who have engaged in the protocol honestly (because Alice has obtained the full multi-signature already).

Generally, for multi-signature schemes, if all parties follow the protocol correctly, then a multi-signature is securely generated. However, the security model of multi-signature schemes does not provide security against an attacker who is in the group of signers and who eventually refuses to complete his part in the signing protocol whilst the others have completed their parts. As we have illustrated in the above scenario, multi-signature schemes are not fair to honest signers since only the malicious signer can output a multi-signature whilst the others cannot. Therefore, a fair multi-signature scheme is needed in order to provide fairness for honest signers.

6.1.1 Related Work

Invented by Itakura and Nakamura in 1983 [IN83], a multi-signature scheme allows a group of n signers together to generate a signature with a constant size on the same message. After its invention, many variants of the multi-signature scheme have been proposed, including [IN83, BN06, Oka88, MOR01, Bol03, LOS$^+$06, BJ08, BCJ08, HRL09, RY07].

In 1988, a multi-signature scheme using bijective public-key cryptosystems was proposed by Okamoto in [Oka88]. Later, Micali, Otha and Reyzin [MOR01] proposed a formal security model for multi-signatures. The rouge key attack is an attack model where a malicious can arbitrarily register a public key without a knowledge of the private key associated with the registered public key. To avoid rouge key attacks, the "Knowledge of Secret Key" assumption was first introduced in their paper. To achieve the Knowledge of Secret Key assumption, all of the signers must be involved in the interactive pre-protocol. There is another way to achieve the Knowledge of Secret Key assumption, which is by employing the Key Registration Model for Public Key Infrastructure. This is introduced by Ristenpart and Yilek in [RY07].

In 2003, a multi-signature generation that does not require the signers to interact was proposed by Boldyreva [Bol03]. This is a multi-signature scheme based on a short signature scheme proposed by Boneh, Lynn and Shacham in [BLS01].

The plain public-key model is a model where all parties are required to register their public key to a Certification Authority before the multi-signature generation begins. In CCS'06, Bellare and Neven [BN06] were the first to propose a multi-signature scheme that is secure against rouge key attacks in the plain public-key model. In EUROCRYPT'07, Ristenpart and Yilek [RY07] proposed a solution for the security against rouge key attacks in the registered key model. In this model, all parties are required to provide a proof of possession of their private key in order to register their public key to a Certification Authority before the multi-signature generation begins. Hence, rouge key attacks are contained in the registered key model.

The closest primitive to a multi-signature is an aggregate signature. This was first introduced by Boneh, Gentry, Lynn, and Shacham (BGLS in short) [BGLS03]. This notion allows each signer to compute his/her signature on the same message and then aggregates those signatures to obtain a multi-signature. Moreover, the verifiable encrypted signature based on the aggregate signature was also introduced in [BGLS03]. The BGLS's verifiable encrypted signature are similar to an aggregate signature by appending a simulated signature into an original signature. The simulated signature is in fact the variant of ElGamal encryption algorithm.

The other variant is a sequential aggregate signature scheme based on The RSA cryptosystem proposed by Lysyanskaya, Micali, Reyzin and Shacham in [LMRS04]. This notion is similar to an aggregate signature and allows a group of signers to construct a multi-signature sequentially. Hence, if the last signer in the sequence refuses to output a multi-signature, then none of the rest can output a multi-signature. Later, a new sequential aggregate signature scheme and a new verifiable encrypted signature that are efficient and provable as secure in the standard model were proposed by Lu, Ostrovsky, Sahai, Shacham and Waters (LOSSW in short) in [LOS+06].

There are verifiable encrypted signature schemes that are not designed for a signature to be aggregated after decrypting the verifiable encrypted signature, such as [Ate04, CD00]. These schemes are the traditional verifiable encrypted signature schemes, which are constructed from a signature that is non-aggregatable, and hence they do not fit in with our generic construction of fair multi-signature schemes.

6.1.2 Our Contributions

In this chapter, we introduce the notion of fair multi-signature (FMS) schemes to solve the outlined problem above. In our notion, nobody will be able to produce a valid signature until all signers produce their partial signatures correctly. We describe the model of the FMS scheme and its security notions to capture the integrity of a message, and the non-repudiation of the signers. We also present a generic construction of the FMS scheme that is proven to be secure in our security model.

Chapter Organisation

The rest of the chapter is organised as follows: in the next section, the definition of FMS and its security notations will be described. We will give a generic construction of the FMS scheme from a verifiable encrypted signature scheme based on an aggregate signature in Section 6.3. In Section 6.4, we will provide a proof of the security of this generic construction. Next, we will give two instantiations of this generic construction in Section 6.5 and Section 6.6. Finally, we will conclude the chapter.

6.2 Definition of Fair Multi-Signature Schemes

In this section, we give a definition of fair multi-signature (FMS) schemes that allow a group of signers to cooperate fairly and sign the same message. In fair multi-signature schemes, if the protocol is complete, then every signer can output a multi-signature. However, if a signer cannot output a multi-signature after completing the interaction then the other signers cannot output a multi-signature either. We describe below the notion of a fair multi-signature scheme and its security model.

6.2.1 Outline of FMS

It is assumed that all parties who need to use their public-private key pair in this scheme comply with a registration protocol with a certificate of authority to obtain certificates on their public key prior to communication with others. Let $\mathfrak{LS}$ be a list of all of the signers, such that $\mathfrak{LS} = \{pk_{S_i}\}$ where i is an index of the signer and pk_i is a public key of the i-signer. Let n be the total number of signers involved in the signature. Let TP denote a semi-trusted third party who is not involved in

the list of signers $\mathfrak{LS}$. TP is assumed to be trusted to handle the partial signature computation and TP is assumed that it does not collude with the malicious signers. A fair multi-signature scheme Σ is a 5-tuple ($Setup$, $TKeyGen$, $SKeyGen$, $Sign$, $Verify$), which is described as follows.

System Parameters Generation ($Setup$):

This is a probabilistic algorithm that, given a security parameter ℓ as input, outputs the system parameter param. That is,

$$Setup(1^{\ell}) \rightarrow \mathsf{param}.$$

TP Key Generator ($TKeyGen$):

This is a probabilistic algorithm that, given the system parameter param as input, outputs strings (sk_{TP}, pk_{TP})which denote a private key and a public key of a semi-trusted third party, respectively. That is,

$$TKeyGen(\mathsf{param}) \rightarrow (pk_{TP}, sk_{TP}).$$

Signer Key Generator ($SKeyGen$):

This is a probabilistic algorithm that, given the system parameter param as input, outputs strings (sk_S, pk_S), which denote a private key and a public key of a signer, respectively. That is,

$$SKeyGen(\mathsf{param}) \rightarrow (pk_S, sk_S).$$

Signature Signing ($Sign$):

$Sign$ is an interactive protocol involving a group of signers and a semi-trusted third party. Let us denote by

$$Sign.\langle S_1(sk_{S_1}), ..., S_n(sk_{S_n}), TP(sk_{TP})\rangle(\mathfrak{LS}, pk_{TP}, M) \rightarrow \sigma$$

a signing protocol $Sign$ that involves a group of signers and a semi-trusted third party and outputs a signature σ, where M is an input message and $\mathfrak{LS}$ is a list of the signers' public key involved in the signing process.

Signature Verification ($Verify$):

This is a deterministic algorithm that, given the list of the signers' public key $\mathfrak{LS}$, a message M and a signature σ as input, outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(M, \sigma, \mathfrak{LS}) \rightarrow \mathsf{d}.$$

6.2.2 Unforgeability

In this section, when we discuss the unforgeability property, it means security against existential unforgeability under an adaptive chosen message and chosen public key attack. Intuitively, the unforgeability property of FMS schemes is provided that, with the cooperation of $\mathsf{n}-1$ corrupted signers and a corrupted TP, an adversary should not be able to forge a multi-signature without interacting with an honest signer, where n is the total number of signers signing a message. Here, our definition of unforgeability is to provide assurance that someone with access to a key generation oracle, a signing oracle and a verification oracle, and with the entire set of the signers' public parameters $pk_{S_1}, ..., pk_{S_\mathsf{n}}$ and the knowledge of $\mathsf{n}-1$-signer secret keys $sk_{S_1}, ..., sk_{S_{\mathsf{n}-1}}$ and a TP private key sk_{TP}, should be unable to produce a multi-signature on a new message, even with the capability of arbitrarily choosing the $\mathsf{n}-1$-signers' secret keys, a TP private key and message M as input.

The following game describes the existential unforgeability of the FMS scheme. Let $CM\text{-}CPK\text{-}A$ be the adaptively chosen message, chosen public key and insider corruption attack and let $EUF\text{-}FMS$ be the existential unforgeability of the FMS scheme. We denote by $\mathcal{A}$ the adaptively chosen message, chosen public key and insider corruption adversary. We also denote by $\mathcal{F}$ the simulator.

First, let $\mathfrak{SL}$ be an algorithm that maintains the list of public-private key pairs and let $\mathfrak{QK}$ be an algorithm that maintains the list of queried private keys. Next, we define the signer's public key generation oracle $\boldsymbol{\mathcal{SPO}}$, the signer's private key generation oracle $\boldsymbol{\mathcal{SKO}}$, the semi-trusted third party's private key generation oracle $\boldsymbol{\mathcal{TKO}}$ and the interactive signing oracle $\boldsymbol{\mathcal{SSO}}$ as follows.

$\boldsymbol{\mathcal{SPO}}$ oracle: At most q_{SP} times, $\mathcal{A}$ can make a query for a new public key of a signer S or semi-trusted third party TP to $\boldsymbol{\mathcal{SPO}}$. Let i represent the signer S or the semi-trusted third party TP. As a response, given the system parameter param, if $i = S$ then $\boldsymbol{\mathcal{SPO}}$ runs the $SKeyGen$ algorithm to generate a public-private key pair of the signer (pk_S, pk_S). Otherwise, $\boldsymbol{\mathcal{SPO}}$ runs the $TKeyGen$ algorithm to generate a public-private key pair of the signer (pk_{TP}, pk_{TP}). Next, $\boldsymbol{\mathcal{SPO}}$ returns pk_i to $\mathcal{A}$. Then $\boldsymbol{\mathcal{SPO}}$ keeps a record in the $\mathfrak{SL}$, which is $\mathfrak{SL} \leftarrow \mathfrak{SL}(pk_i, sk_i)$.

$\boldsymbol{\mathcal{SKO}}$ oracle: At most q_{SK} times, $\mathcal{A}$ can make a query for a signer's private key sk_S with respect to a chosen signer's public key pk_S to $\boldsymbol{\mathcal{SKO}}$. As a response,

$\boldsymbol{\mathcal{SKO}}$ matches the signer's public key pk_S in the list $\mathfrak{SL}$ to obtain the signer's private key pk_S. Then $\boldsymbol{\mathcal{SKO}}$ returns sk_S to $\mathcal{A}$. Next, $\boldsymbol{\mathcal{SKO}}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_S, sk_S)$.

$\boldsymbol{\mathcal{TKO}}$ oracle: At most q_{TK} times, $\mathcal{A}$ can make a query for a semi-trusted third party's private key sk_{TP} with a respect to a chosen semi-trusted third party's public key pk_{TP} to $\boldsymbol{\mathcal{TKO}}$. As a response, $\boldsymbol{\mathcal{TKO}}$ matches the semi-trusted third party's public key pk_{TP} in the list $\mathfrak{SL}$ to obtain the semi-trusted third party's private key pk_{TP} Then $\boldsymbol{\mathcal{TKO}}$ returns sk_TP to $\mathcal{A}$. Next, $\boldsymbol{\mathcal{TKO}}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_{TP}, sk_{TP})$.

$\boldsymbol{\mathcal{SSO}}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a fair multi-signature σ on its choice of a message M. As a response, $\boldsymbol{\mathcal{SSO}}$ acts as the signers $S_1, ..., S_n \in \mathfrak{LS}$ and the semi-trusted third party TP, and then runs the interactive protocol $Sign$ to generate a signature σ on a message M corresponding with $pk_{S_1}, ..., pk_{S_n}$. $\boldsymbol{\mathcal{SSO}}$ then returns σ, M to $\mathcal{A}$.

Next, we begin the experiment $\text{Expt}^{\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}}(\ell)$ as follows: given a choice of messages M and access to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SKO}}$, $\boldsymbol{\mathcal{TKO}}$ and $\boldsymbol{\mathcal{SSO}}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles in an adaptive way. At the end of these queries, we assume that $\mathcal{A}$ outputs a forged multi-signature σ^* on a new message M^* with respect to the list of the signer's public keys $\mathfrak{LS}^*$. We say that $\mathcal{A}$ wins the game if:

1. for at least one $pk_{S_*} \in \mathfrak{LS}^* : sk_{S_*} \notin \mathfrak{QR}$.
2. $sk_{TP_*} \notin \mathfrak{QR}$.
3. $Accept \leftarrow Verify(M^*, \sigma^*, \mathfrak{LS}^*)$.

The probability of success of $\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}$ winning the above game is defined as $Succ^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}(.)$.

Definition 6.1 *The FMS scheme is said to be ($\mathtt{t}$,q_H,q_{SP},q_{SS},q_{SK},ϵ)-secure existential unforgeable under an adaptive chosen message, chosen public key and insider corruption attack if there is no PPT adversary $\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}$ such that the success probability $Succ^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}(\ell) = \epsilon$ is non-negligible in ℓ, where $\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EUF\text{-}FMS}$ runs in time at most $\mathtt{t}$, and makes at most q_H, q_{SP}, q_{SS}, q_{TK} and q_{SK} queries to the random oracles, $\boldsymbol{\mathcal{SPO}}$ oracle, $\boldsymbol{\mathcal{SSO}}$ oracle, $\boldsymbol{\mathcal{TKO}}$ oracle and $\boldsymbol{\mathcal{SKO}}$ oracle, respectively.*

6.2.3 Fairness

Intuitively, the definition of fairness has two sub-properties. First, completeness if the signing protocol is completed and the semi-trusted third party is not compromised, then every signer in $\mathfrak{LS}$ should output the same multi-signature. Second, soundness if the signing protocol is uncompleted or interrupted and the semi-trusted third party is not compromised, then no one should be able to output a multi-signature corresponding to the list $\mathfrak{LS}$ and a message M. However, the completeness of fairness property is straightforward Hence, we will only consider for the soundness of fairness. The following game describes the existential fairness with soundness of the FMS scheme. We denote by $EFS\text{-}FMS$ the existential fairness with soundness of the FMS scheme. Let $\mathcal{A}_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ be the adaptively chosen message, chosen public key and insider corruption adversary. Let $\mathcal{F}$ be a simulator of the existential fairness with soundness game. First, the interactive signing oracle $\boldsymbol{\mathcal{SSO}}$, the signer's public key generation oracle $\boldsymbol{\mathcal{SPO}}$ and the signer's private key generation oracle $\boldsymbol{\mathcal{SKO}}$ are defined below. Let $\mathfrak{SL}$ be an algorithm that maintains the list of public-private key pairs and let $\mathfrak{QR}$ be an algorithm that maintains the list of queried private keys.

$\boldsymbol{\mathcal{SPO}}$ oracle: At most q_{SP} times, $\mathcal{A}$ can make a query for a new public key of a signer S or semi-trusted third party TP to $\boldsymbol{\mathcal{SPO}}$. Let i represent the signer S or the semi-trusted third party TP. As a response, given the system parameter param, if $i = S$, then $\boldsymbol{\mathcal{SPO}}$ runs the $SKeyGen$ algorithm to generate a public-private key pair of the signer (pk_S, pk_S). Otherwise, $\boldsymbol{\mathcal{SPO}}$ runs the $TKeyGen$ algorithm to generate a public-private key pair of the signer (pk_{TP}, pk_{TP}). Next, $\boldsymbol{\mathcal{SPO}}$ returns pk_i to $\mathcal{A}$. Then $\boldsymbol{\mathcal{SPO}}$ keeps a record in the $\mathfrak{SL}$, which is $\mathfrak{SL} \leftarrow \mathfrak{SL}(pk_i, sk_i)$.

$\boldsymbol{\mathcal{SKO}}$ oracle: At most q_{SK} times, $\mathcal{A}$ can make a query for a signer's private key sk_S with a respect to a chosen signer's public key pk_S to $\boldsymbol{\mathcal{SKO}}$. As a response, $\boldsymbol{\mathcal{SKO}}$ matches the signer's public key pk_S in the list $\mathfrak{SL}$ to obtain the signer's private key pk_S. Then $\boldsymbol{\mathcal{SKO}}$ returns sk_S to $\mathcal{A}$. Next, $\boldsymbol{\mathcal{SKO}}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_S, sk_S)$.

$\boldsymbol{\mathcal{SSO}}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a fair multi-signature σ on its choice of a message M. As a response, $\boldsymbol{\mathcal{SSO}}$ acts as the signers $S_1, ..., S_n \in \mathfrak{LS}$ and the semi-trusted third party TP, and then runs the

interactive protocol *Sign* to generate a multi-signature σ on a message M corresponding with $pk_{S_1}, ..., pk_{S_n}$. $\boldsymbol{\mathcal{SSO}}$ then returns σ, M to $\mathcal{A}$.

We then begin the experiment as follows: given a choice of messages M and access to the above oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles. At the end of these queries, $\mathcal{F}$, who plays the role of an honest signer pk_S^*, interacts with $\mathcal{A}$, who plays the role of a corrupted signers. We assume that $\mathcal{A}$ outputs a forged multi-signature σ^* on a message M^* with respect to $\mathfrak{LG}^*, pk_{TA}$. $\mathcal{A}$ wins the above game if the private key of the honest signer pk_S^* in $\mathfrak{LG}^*$ is not known to $\mathcal{A}$, and $\mathcal{F}$ (as a signer S^*) did not fully complete the signing protocol with $\mathcal{A}$ and TP. Note that "not fully completed the interaction" means that $\mathcal{F}$ communicated only with the signers in the $\mathfrak{LG}^*$ but not with TP; hence, neither $\mathcal{F}$ nor the signers in the $\mathfrak{LG}^*$ should be able to output a multi-signature σ^*. The success probability that $\mathcal{A}_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ wins the above game is defined as $Succ_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}(.)$.

Definition 6.2 *The FMS scheme is said to be* ($\mathtt{t}$,q_H,q_{SP},q_{SS},q_{SK},ϵ)*-secure existential fair with soundness under an adaptive chosen message, chosen public key and insider corruption attack if there is no PPT adversary* $\mathcal{A}_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ *such that the success probability* $Succ_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}(\ell) = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}_{EFS\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ *runs in time at most* $\mathtt{t}$*, makes at most* q_H*,* q_{SP}*,* q_{SS}*, and* q_{SK} *queries to the random oracles,* $\boldsymbol{\mathcal{SPO}}$ *oracle,* $\boldsymbol{\mathcal{SSO}}$ *oracle and* $\boldsymbol{\mathcal{SKO}}$ *oracle, respectively.*

6.2.4 Semi-trust

Intuitively, the definition of semi-trust is to prevent an adversary that acts as a semi-trusted third party from outputting a multi-signature after interacting with a signing protocol that involves honest signers. To simplify this, we say that an adversary, corrupted with a semi-trusted third party, arbitrarily interacts with honest signers and breaks the semi-trust of the FMS scheme if the adversary outputs a multi-signature σ^* on a new message M^* after completing an interaction with the arbitrarily chosen honest signers. Let us denote by $CM\text{-}CPK\text{-}A$ the adaptively chosen message and chosen public key attack, and let us denote by $EST\text{-}FMS$ the existential semi-trust of the FMS scheme. Let $\mathcal{A}_{EST\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ be the adaptively chosen message, chosen public key and insider corruption adversary. Let $\mathcal{F}$ be a simulator of the existential semi-trust game.

First, let $\mathfrak{SL}$ be an algorithm that maintains the list of public-private key pairs and let $\mathfrak{QR}$ be an algorithm that maintains the list of queried private keys. Next,

we define the signer's public key generation oracle $\boldsymbol{\mathcal{SPO}}$, the signer's private key generation oracle $\boldsymbol{\mathcal{SKO}}$ and the interactive signing oracle $\boldsymbol{\mathcal{SSO}}$ as follows.

$\boldsymbol{\mathcal{SPO}}$ oracle: At most q_{SP} times, $\mathcal{A}$ can make a query for a new public key of a signer S or semi-trusted third party TP to $\boldsymbol{\mathcal{SPO}}$. Let i represent the signer S or the semi-trusted third party TP. As a response, given the system parameter param, if $i = S$, then $\boldsymbol{\mathcal{SPO}}$ runs the $SKeyGen$ algorithm to generate a public-private key pair of the signer (pk_S, pk_S). Otherwise, $\boldsymbol{\mathcal{SPO}}$ runs the $TKeyGen$ algorithm to generate a public-private key pair of the signer (pk_{TP}, pk_{TP}). Next, $\boldsymbol{\mathcal{SPO}}$ returns pk_i to $\mathcal{A}$. Then $\boldsymbol{\mathcal{SPO}}$ keeps a record in the $\mathfrak{SL}$, which is $\mathfrak{SL} \leftarrow \mathfrak{SL}(pk_i, sk_i)$.

$\boldsymbol{\mathcal{SKO}}$ oracle: At most q_{SK} times, $\mathcal{A}$ can make a query for a signer's private key sk_S with a respect to a chosen signer's public key pk_S to $\boldsymbol{\mathcal{SKO}}$. As a response, $\boldsymbol{\mathcal{SKO}}$ matches the signer's public key pk_S in the list $\mathfrak{SL}$ to obtain the signer's private key pk_S. Then $\boldsymbol{\mathcal{SKO}}$ returns sk_S to $\mathcal{A}$. Next, $\boldsymbol{\mathcal{SKO}}$ keeps a record of this query in the $\mathfrak{QR}$, which is $\mathfrak{QR} \leftarrow \mathfrak{QR}(pk_S, sk_S)$.

$\boldsymbol{\mathcal{SSO}}$ oracle: At most q_{SS} times, $\mathcal{A}$ can make a query for a fair multi-signature σ on its choice of a message M. As a response, $\boldsymbol{\mathcal{SSO}}$ acts as the signers $S_1, ..., S_n \in \mathfrak{LS}$ and the semi-trusted third party TP, and then runs the interactive protocol $Sign$ to generate a signature σ on a message M corresponding with $pk_{S_1}, ..., pk_{S_n}$. $\boldsymbol{\mathcal{SSO}}$ then returns σ, M to $\mathcal{A}$.

Next, we begin the experiment $\text{Expt}^{\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EST\text{-}FMS}}(\ell)$ as follows: given a choice of messages M and access to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SKO}}$, $\boldsymbol{\mathcal{TKO}}$ and $\boldsymbol{\mathcal{SSO}}$ oracles, $\mathcal{A}$ arbitrarily makes queries to the oracles in an adaptive way. At the end of these queries, $\mathcal{A}$ begins the challenge by running an interactive protocol $Sign$ with $\mathcal{F}$. $\mathcal{F}$ acts as $S_1, ...S_n \in \mathfrak{LS}$ and $\mathcal{A}$ acts as TP. We assume that $\mathcal{A}$ outputs a multi-signature σ^* on a new message M^* with respect to the list of the signer's public keys $\mathfrak{LS}^*$. We say that $\mathcal{A}$ wins the game if:

1. $sk_{S_1}, ..., sk_{S_n} \notin \mathfrak{QR}$.
2. $Accept \leftarrow Verify(M, \sigma, \mathfrak{LS})$.

The success probability that $\mathcal{A}^{CM\text{-}CPK\text{-}A}_{EST\text{-}FMS}$ wins the above game is defined as $Succ^{CM\text{-}CPK\text{-}A}_{EST\text{-}FMS}(.)$.

Definition 6.3 *The FMS scheme is said to be* ($\mathtt{t}$,q_H,q_{SP},q_{SS},q_{SK},ϵ)*-secure existential semi-trust under an adaptive chosen message and chosen public key attack if there is no PPT adversary* $\mathcal{A}_{EST\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ *such that the success probability* $Succ_{EST\text{-}FMS}^{CM\text{-}CPK\text{-}A}(\ell) = \epsilon$ *is non-negligible in* ℓ*, where* $\mathcal{A}_{EST\text{-}FMS}^{CM\text{-}CPK\text{-}A}$ *runs in time at most* $\mathtt{t}$*, makes at most* q_H*,* q_{SP}*,* q_{SS}*, and* q_{SK} *queries to the random oracles,* $\mathcal{SPO}$ *oracle,* $\mathcal{SSO}$ *oracle and* $\mathcal{SKO}$ *oracle, respectively.*

6.3 Generic Construction of FMS scheme

In this section, we present a generic construction of the FMS scheme. Before describing our generic construction, in the following subsection we will discuss about the definition of a verifiable encrypted signature scheme that constructed from an aggregate signature scheme. Next, we provide the definition of an aggregate signature scheme. Then we will proceed with the generic construction of the FMS scheme.

6.3.1 Verifiable Encrypted Signature Scheme from Aggregate Signature

There are two well-known verifiable encrypted signatures (VES) that are constructed from aggregate signatures. The first scheme [BGLS03] is constructed from the BLS signature and the second scheme [LOS+06] is constructed from Waters signature scheme [Wat05]. From these two schemes, we will adopt the VES model that they propose and describe it as follows.

System Parameter Generation ($VES.Setup$):

Setup is a PPT algorithm that, on input a security parameter ℓ, outputs the system parameter param.

Key Generator ($VES.KeyGen$): *KeyGen* is a PPT algorithm that, on input the system parameter param, outputs strings (sk, pk), which denote a private key and a public key, respectively. That is,

$$KeyGen(\mathsf{param}) \rightarrow (pk, sk).$$

Note that we assume that the key generator algorithm for a signer is the same as the key generator algorithm for an adjudicator. Even through some

VES schemes require these algorithms to be different, there is only a trivial adjustment.

Signature Signing ($VES.Sign$):

On input the system parameter param, the signer's private key sk_S, the signer's public key pk_S and a message M, $Sign$ outputs the signer's signature σ. That is,

$$Sign(\mathsf{param}, M, sk_S, pk_S) \rightarrow \sigma.$$

Signature Verification ($VES.Verify$):

On input the system parameter param, the signer's public key pk_S, a message M and a signature σ, $Verify$ outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$Verify(\mathsf{param}, M, \sigma, pk_S) \rightarrow \mathsf{d}.$$

Verifiable Encryption ($VES.Enc$):

On input the system parameter param, the adjudicator's public key pk_{AD} and a signature σ, Enc outputs a verifiable encrypted signature $\mathfrak{o}$. That is,

$$Enc(\mathsf{param}, \sigma, pk_{AD}) \rightarrow \mathfrak{o}.$$

Verifiable Encrypted Signature's Verification ($VES.EVF$):

On input the system parameter param, the adjudicator's public key pk_{AD}, the signer's public key pk_S, a message M and a verifiable encrypted signature $\mathfrak{o}$, EVF outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$EVF(\mathsf{param}, M, \mathfrak{o}, pk_{AD}, pk_S) \rightarrow \mathsf{d}.$$

Verifiable Encrypted Signature's Adjudication ($VES.ADJ$):

On input the system parameter param, the adjudicator's public key pk_{AD}, the adjudicator's private key sk_{AD}, the signer's public key pk_S, a verifiable encrypted signature $\mathfrak{o}$ and a message M, ADJ outputs a signature σ. That is,

$$ADJ(\mathsf{param}, M, \mathfrak{o}, sk_{AD}, pk_{AD}, pk_S) \rightarrow \sigma.$$

Note that signatures from $VES.Sign$ can be aggregated since the verifiable encryption signature scheme is based on the aggregate signature scheme.

6.3.2 Aggregate Signature Scheme

Let AS denote a aggregate signature scheme. We describe an aggregate signature scheme as follows.

Setup ($AS.Setup$) and Key Generator ($AS.KeyGen$):

$AS.Setup$ and $AS.KeyGen$ are the same as $VES.Setup$ and $VES.KeyGen$ in the VES scheme, respectively.

Sign ($AS.Sign$) and Verify ($AS.Verify$):

$AS.Sign$ and $AS.Verify$ are the same as $VES.Sign$ and $VES.Verify$ in the VES scheme, respectively.

Aggregation ($AS.Aggregate$):

On input the system parameter param, signatures $\sigma_{S_1}, ..., \sigma_{S_n}$ and a message M, $AVerify$ outputs an aggregate signature $\mathfrak{q}$. That is,

$$Aggregate(\mathsf{param}, M, \sigma_{S_1}, ..., \sigma_{S_n}) \rightarrow \mathfrak{q}.$$

Aggregate Signature Verification ($AS.AVerify$):

On input the system parameter param, the signer's public keys $pk_{S_1}, ..., pk_{S_n}$, a message M and an aggregate signature $\mathfrak{q}$, $AVerify$ outputs a verification decision $\mathsf{d} \in \{Accept, Reject\}$. That is,

$$AVerify(\mathsf{param}, M, \mathfrak{q}, pk_{S_1}, ..., pk_{S_n}) \rightarrow \mathsf{d}.$$

6.3.3 Generic Construction Scheme

In this section, we present our generic construction scheme. The scheme works as follows.

Setup: On input a security parameter ℓ, *Setup* runs $VES.Setup$ and returns param.

TKeyGen: On input a system parameter param, a semi-trusted third party TP randomly generates a private key sk_{TP} and a public key pk_{TP} as follows: run $VES.KeyGen$ and output (sk_{TP}, pk_{TP}) as a private key and public key of the semi-trusted third party, respectively.

SKeyGen: Similar to $TKeyGen$, $SKeyGen$ runs $VES.KeyGen$ to obtain (sk_S, pk_S) as a private key and public key of the signer, respectively.

Sign: Assume that communication between the parties is secure. Given a message M, a list of signers $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$ and a private key sk_{S_i}, a signer S_i, where $i \in \{1, ..., n\}$, processes the $Sign$ protocol as follows.

- Round 1: All the signers work together and run $VES.KeyGen$ to generate (sk_R, pk_R) as a shared random private key and a shared random public key, respectively.
- Round 2: On input $sk_R, pk_R, sk_{S_i}, pk_{S_i}, pk_{TP}$, a signer S_i computes as follows.

$$\begin{aligned} \varpi_i &= VES.Sign(\mathsf{param}, M, sk_{S_i}, pk_{S_i}), \\ \vartheta_i &= VES.Esign(\mathsf{param}, \varpi_i, pk_R), \\ \mathtt{d}_i &= VES.Esign(\mathsf{param}, \vartheta_i, pk_{TP}). \end{aligned}$$

  S_i then sends $\Upsilon_i = (\mathtt{d}_i, pk_R)$ to TP.
- Round 3: Upon receiving $\Upsilon_1, ..., \Upsilon_n$, TP first checks whether pk_R in each $\Upsilon_1, ..., \Upsilon_n$ are the same. Next, check whether $\forall i \in \{1, ..., n\}$:

$$VES.EVF(\mathsf{param}, M, \Upsilon_i, pk_{TP}, pk_S) \stackrel{?}{=} Accept$$

  and let $\lambda_i = VES.ADJ(\mathsf{param}, \mathtt{d}_i, sk_{TP})$. Then check whether

$$VES.EVF(\mathsf{param}, M, \lambda_i, pk_R, pk_S) \stackrel{?}{=} Accept$$

  Finally, if the above holds then TP outputs a vector $\sigma_1 = \{\lambda_1, ..., \lambda_n\}$ and sends it to all the signers.
- Extract the multi-signature: each signer computes $\forall i \in \{1, ..., n\} : \sigma_i = VES.ADJ(\mathsf{param}, \lambda_i, sk_R)$. The multi-signature on message M is

$$\Theta := AS.Aggregate(\sigma_1, ..., \sigma_n).$$

Verify: Given $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$, Θ and a message M, a verifier V runs $AS.AVerify(\ \mathsf{param}, M, \Theta, pk_{S_1}, ..., pk_{S_n})$. If $AS.AVerify$ outputs *Accept* then it accepts the signature. Otherwise, it outputs reject.

6.4 Security Analysis for The Generic Construction Scheme

6.4.1 Unforgeability

Theorem 6.1 *Our fair multi-signature scheme is existential unforgeable under an adaptive chosen message, chosen public key attack and insider corruption if the verifiable encrypted signature scheme is secure against existential forgery.*

Proof: The following experiment between an unforgeability adversary $\mathcal{A}$ and a VES forgery $\mathcal{F}$ shows that $\mathcal{F}$ can use $\mathcal{A}$ to forge a VES signature if $\mathcal{A}$ breaks the unforgeability under an adaptive chosen message and chosen public key attack, and insider corruption exists. Let $\mathcal{S}$ be the VES simulator. Since $\mathcal{F}$ can forward all queries to $\mathcal{S}$, $\mathcal{F}$ can construct the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{TKO}$, $\mathcal{SKO}$ and $\mathcal{HO}$ oracles straightforwardly and leave only one signer S^*, which will forward the signature queries to $\mathcal{S}$. Note that if $\mathcal{A}$ makes a query for private key of the signer S^* then $\mathcal{F}$ aborts the experiment. $\mathcal{F}$ gives access to the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{TKO}$, $\mathcal{SKO}$ and $\mathcal{HO}$ oracles to $\mathcal{A}$. After $\mathcal{A}$ finishes making arbitrary queries to these oracles, it chooses a set of public keys and outputs a multi-signature Θ^*. The probability that $\mathcal{F}$ will not abort is $(1 - \frac{1}{q_{SK}+n})^{q_{SK}}$ where q_{SK} is an upper bound of the queries that are made to the $\mathcal{SKO}$ oracle. This probability is negligible; hence, $\mathcal{F}$ outputs a forgery $\mathfrak{o}$ by extracting from Θ^*, since Θ^* is an aggregate signature and $\mathcal{F}$ knows all the secret keys of the signers except for the signer S^*. □

6.4.2 Fairness

Theorem 6.2 *Our fair multi-signature scheme is existential fair with soundness secure under an adaptive chosen message, chosen public key attack and insider corruption if the verifiable encrypted signature scheme is secure against existential forgery.*

Proof: The following experiment between an unforgeability adversary $\mathcal{A}$ and a VES forgery $\mathcal{F}$ shows that $\mathcal{F}$ can use $\mathcal{A}$ to forge a VES signature if $\mathcal{A}$ that breaks the fairness with soundness under an adaptive chosen message, chosen public key attack and insider corruption exists. Let $\mathcal{S}$ be the VES simulator. The proof here is similar to the proof in Theorem 6.1. Since $\mathcal{F}$ can forward all queries to $\mathcal{S}$, $\mathcal{F}$ can construct the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{TKO}$, $\mathcal{SKO}$ and $\mathcal{HO}$ oracles straightforwardly and leave only

one signer S^*, which will forward the signature queries to $\mathcal{S}$. If $\mathcal{A}$ makes a query for the private key of the signer S^* to the $\boldsymbol{\mathcal{SKO}}$ oracle, then $\mathcal{F}$ aborts the experiment. $\mathcal{F}$ gives access to the $\boldsymbol{\mathcal{SPO}}$, $\boldsymbol{\mathcal{SSO}}$, $\boldsymbol{\mathcal{TKO}}$, $\boldsymbol{\mathcal{SKO}}$ and $\boldsymbol{\mathcal{HO}}$ oracles to $\mathcal{A}$. After $\mathcal{A}$ finishes making arbitrary queries to these oracles, it chooses a set of public keys and interacts with the TP^* played by $\mathcal{F}$. $\mathcal{A}$ wins the game if the following conditions hold:

- $\mathcal{A}$ outputs Θ^*.
- $Accept \leftarrow Verify(M^*, \Theta^*, \mathfrak{LS})$.
- The input: M^* and $\mathfrak{LS}^*$ are never submitted to the $\boldsymbol{\mathcal{SSO}}$ oracles.
- At least one honest signer's private key in $\mathfrak{LS}$ have never been queried to the $\boldsymbol{\mathcal{SKO}}$ oracle.
- pk_{TP^*} has never been queried to the $\boldsymbol{\mathcal{TKO}}$ oracle.
- The transaction between $\mathcal{A}$ and the honest signers is uncompleted.

Note that, since the honest signer(s) generates a random public-private key pair in the first round and does not need to participate in the second and third rounds, $\mathcal{F}$ does not need to provide any information except the public key of the honest signer(s) to $\mathcal{A}$. The probability that $\mathcal{F}$ will not abort is $(1 - \frac{1}{q_{SK}+n})^{q_{SK}}$ where q_{SK} is an upper bound of queries that are made to the $\boldsymbol{\mathcal{SKO}}$ oracle. This probability is negligible and hence $\mathcal{F}$ outputs a forgery $\mathfrak{o}$ by extracting from Θ^*, since Θ^* is an aggregate signature and $\mathcal{F}$ knows all the private key of signers except for the signer S^*. □

6.4.3 Semi-trust

Theorem 6.3 *Our fair multi-signature scheme is existential semi-trust secure under an adaptive chosen message and chosen public key attack if the verifiable encrypted signature is secure against the extraction defined in [BGLS03].*

Proof: Assume that an adversary $\mathcal{A}$ runs the existential semi-trust game defined in Section 6.2.4 and successfully outputs a multi-signature. We will then show that an adversary $\mathcal{F}$ can use $\mathcal{A}$ to break the extraction of the verifiable encrypted signature. Let $\mathcal{S}$ be the VES simulator in the extraction game [BGLS03]. The proof of this is

similar to the proof in Theorem 6.1. Since $\mathcal{F}$ can generate the secret keys for all the signers, except for one signer S^* where its signature queries will be forwarded to $\mathcal{S}$, $\mathcal{F}$ can construct the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{TKO}$, $\mathcal{SKO}$ and $\mathcal{HO}$ oracles straightforwardly. $\mathcal{F}$ gives access to the $\mathcal{SPO}$, $\mathcal{SSO}$, $\mathcal{TKO}$, $\mathcal{SKO}$ and $\mathcal{HO}$ oracles to $\mathcal{A}$. After $\mathcal{A}$ finishes making arbitrary queries to these oracles, it chooses a set of public keys. Then $\mathcal{A}$, acting as TP, interacts with the signers acted by $\mathcal{F}$. The signing protocol runs as follows.

Round 1: Let AD be an adjudicator assigned by $\mathcal{S}$. In this case, $\mathcal{F}$ does not have AD's private key. $\mathcal{F}$ sets $pk_R = pk_{AD}$.

Round 2: For the signer S^*, $\mathcal{F}$ makes a request of

$$\theta^* = VES.Esign(\mathsf{param}, M, VES.Sign(\mathsf{param}, M, sk_{S^*}, pk_{S^*}), pk_R)$$

to $\mathcal{S}$. Then it computes $\Phi_{S^*} = VES.Esign(\mathsf{param}, M, \theta^*, pk_{TP})$. and sends Φ_{S^*} to TP. For the rest of the signers, $\mathcal{F}$ computes Φ_i straightforwardly for the signer S_i and sends it to TP.

Output from $\mathcal{A}$: $\mathcal{A}$ wins the game if $\mathcal{A}$ outputs Θ^* and

$$Verify(\mathsf{param}, M, \Theta^*, pk_{S_1}, ..., pk_{S^*}, ., pk_{S_n}) = Accept.$$

$\mathcal{F}$ outputs a signature σ on a message M of the signer S^* to $\mathcal{S}$ by extracting from Θ^*. since Θ^* is an aggregate signature and $\mathcal{F}$ knows all the secret keys of signers except for the signer S^*. □

6.5 An Instantiation

In this section, we present the instantiation of the generic construction from BGLS's verifiably encrypted signatures.

6.5.1 BGLS's Verifiably Encrypted Signatures

Introduced by Boneh, Gentry, Lynn and Shacham [BGLS03], a verifiably encrypted signature (VES) scheme based on an aggregate signature is a 7-triple ($Setup$, $KeyGen$, $Sign$, $Verify$, Enc, EVF, ADJ). We elaborate the verifiably encrypted signature scheme as follows.

Setup: *Setup* sets $\mathsf{param} = (p, \hat{e}, g_1 \in \mathbb{G}_1, \psi : \mathbb{G}_1 \to \mathbb{G}_2, H : \{0,1\}^* \to \mathbb{G}_2, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T)$ to be a system parameter.

KeyGen: On input a system parameter param, *KeyGen* chooses a random private key $x, y \in \mathbb{Z}_p$. Then, for the signer, *KeyGen* returns $pk_S = X$ and $sk_S = x$ as a public key and a private key of the signer, respectively. For the adjudicator, *KeyGen* returns $pk_{AD} = Y = g_1^y$ and $sk_{AD} = y$ as a public key and a private key of the signer, respectively.

Sign: Given a message M, pk_S and sk_S, S computes $\sigma = H(M)^x$ as a signature on message M.

Verify: Given pk_S, σ and a message M, a verifier V checks whether $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H(M), X)$ holds or not. If it does not, then it outputs *Reject*. Otherwise, it outputs *Accept*.

Enc: Given a signature σ on a message M, *Esign* chooses a random integer $r \in \mathbb{Z}_p$ and computes $\mu = \psi(g_1)^r; \hat{\sigma} = \psi(Y)^r$. Then it computes a verifiably encrypted signature $\mathfrak{o} = (\mu, \omega = \hat{\sigma} \cdot \sigma)$. Note that, in the original BGLS scheme, *Sign* is also a part of *Esign*. However, we separate it to make it unique for constructing the fair multi-signature, so that the signer must run *Sign* to obtain a signature σ before processing *Enc*.

EVF: Given pk_S, pk_{AD}, $\mathfrak{o}$ and a message M, a verifier V checks whether $\hat{e}(g_1, \omega) \stackrel{?}{=} \hat{e}(H(M), X) \cdot \hat{e}(\mu, Y)$ holds or not. If it does not, then it outputs *Reject*. Otherwise, it outputs *Accept*.

ADJ: Given a message sk_{AD} and $\mathfrak{o}$, S computes $\sigma = \omega \cdot \mu^{-y}$ as a signature on a message M.

6.5.2 Instantiation from BGLS Scheme

In this section, we present the instantiation of our generic construction scheme. The scheme works as follows.

Setup: *Setup* works in the same way as the BGLS *VES.Setup*.

TKeyGen: *TKeyGen* works in the same way as the BGLS *VES.KeyGen*. Let $sk_{TP} = y, pk_{TP} = Y = g_1^y$ be a private key and a public key of the semi-trusted third party, respectively.

SKeyGen: Run the BGLS $VES.KeyGen$ to obtain $sk_S = x, pk_S = g_1^x$ as a private key and a public key of the signer, respectively.

Sign: Assume that communication between the parties is secure. Given a message M, a list of signers $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$ and a private key sk_{S_i}, a signer S_i, where $i \in \{1, ..., n\}$, processes the $Sign$ protocol as follows.

- Round 1: Each signer randomly selects γ_i and exchanges it with one another. Each signer generates $sk_R = \gamma = \prod_{i=1}^{n} \gamma_i, pk_R = R = g_1^{\gamma}$ as a shared random secret key and a shared random public key, respectively.
- Round 2: On input $sk_R, pk_R, sk_{S_i}, pk_{S_i}, pk_{TP}$, a signer S_i randomly selects r_1, r_2 and computes $\sigma = H(M)^x, \mu_1 = \psi(g_1)^{r_1}, \mu_2 = \psi(g_1)^{r_2}, \hat{\sigma} = \psi(R)^{r_1}, \check{\sigma} = \psi(Y)^{r_2},$. $\Phi_i = (\mu_1, \mu_2, \omega = \sigma \cdot \hat{\sigma} \cdot \check{\sigma})$ S_i then sends $\Upsilon_i = (\Phi_i, pk_R)$ to TP.
- Round 3: Upon receiving $\Upsilon_1, ..., \Upsilon_n$, TP first checks whether pk_R in each $\Upsilon_1, ..., \Upsilon_n$ are the same. Next, check whether $\forall i \in \{1, ..., n\}$:

$$\hat{e}(g_1, \omega_i) \stackrel{?}{=} \hat{e}(H(M), X) \cdot \hat{e}(\mu_1, R) \cdot \hat{e}(\mu_2, Y)$$

  and let $\lambda_i = (\mu_1, \omega_i' = \omega_i \cdot \mu_2^{-y})$. Then check whether

$$\hat{e}(g_1, \omega_i') \stackrel{?}{=} \hat{e}(H(M), X) \cdot \hat{e}(\mu_1, R).$$

  Finally, if the above holds then TP outputs a vector $\bar{\lambda} = \{\lambda_1, ..., \lambda_n\}$ and sends it to all the signers.
- Extract the multi-signature: each signer computes $\forall i \in \{1, ..., n\} : \sigma_i = \omega_i' \cdot \mu_1^{-\gamma}$. Finally, each signer computes a multi-signature on message M, which is $\Theta = \prod_{i=1}^{n} \sigma_i$.

Verify: Given $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$, Θ and a message M, a verifier V checks whether $\hat{e}(\Theta, g) \stackrel{?}{=} \hat{e}(H(M), \prod_{i=1}^{n} X)$ holds or not. If it holds then a verifier accepts the signature. Otherwise, the verifier reject the signature.

6.6 Another Instantiation in the Standard Model

In this section, we present an instantiation of the generic construction from LOSSW's verifiably encrypted signature. Before describing the instantiation, we briefly review

LOSSW's scheme first. The security of the instantiation is provable in the standard model. The difference between the oracle model and the standard model is presented in Section 2.2.7.

6.6.1 LOSSW's Verifiably Encrypted Signatures

Lu, Ostrovsky, Sahai, Shacham and Waters [LOS+06] proposed a verifiably encrypted signature (VES) scheme based on aggregate signatures in the standard model. We present their VES scheme as a 7-triple (*Setup*, $KeyGen$, *Sign*, $Verify$, *Enc*, EVF, ADJ). We elaborate LOSSW's verifiably encrypted signature scheme as follows.

Setup: *Setup* sets $\mathsf{param} = (p, \hat{e}, g \in \mathbb{G}_1, u_0, u_1, ..., u_k, \psi : \mathbb{G}_1 \to \mathbb{G}_2, H : \{0,1\}^* \to \{0,1\}^k, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T)$ to be a system parameter.

$KeyGen$: On input a system parameter param, $KeyGen$ chooses a random private key $x, y \in \mathbb{Z}_p$. Then, for the signer, $KeyGen$ returns $pk_S = \mathbf{X} = \hat{e}(g,g)^x$ and $sk_S = x$ as a public key and a private key of the signer, respectively. For the adjudicator, $KeyGen$ returns $pk_{AD} = Y = g^y$ and $sk_{AD} = y$ as a public key and a private key of the signer, respectively.

Sign: Given a message M as a bit string $(m_1, ..., m_k) \in \{0,1\}^k$, pk_S and sk_S, S randomly chooses $r \in \mathbb{Z}_p$ and computes $\sigma = (\theta_1 = g^x u_0 \prod_{i=1}^{k} u_i^{m_i}, \theta_2 = g^r)$ as a signature on message M.

$Verify$: Given pk_S, σ and a message M, a verifier V checks whether

$$\hat{e}(\theta_1, g)\hat{e}(\theta_2, u_0 \prod_{i=1}^{k} u_i^{m_i})^{-1} \stackrel{?}{=} \mathbf{X}$$

holds or not. If it does not, then a verifier outputs *Reject*. Otherwise, it outputs *Accept*.

Enc: Given a signature σ on a message M, $Esign$ chooses a random integer $r \in \mathbb{Z}_p$ and computes $E_1 = \theta_1 \cdot Y^r \quad ; \quad E_2 = \theta_1 \quad ; \quad E_3 = g^r$. Then it computes a verifiably encrypted signature $\mathfrak{o} = (E_1, E_2, E_3)$.

EVF: Given pk_S, pk_{AD}, $\mathfrak{o}$ and a message M, a verifier V checks whether

$$\hat{e}(E_1, g) \cdot \hat{e}(E_2, u_0 \prod_{i=1}^{k} u_i^{m_i})^{-1} \cdot \hat{e}(E_3, Y)^{-1} \stackrel{?}{=} \mathbf{X}$$

holds or not. If it does not, then a verifier outputs *Reject*. Otherwise, it outputs *Accept*.

ADJ: Given a message sk_{AD} and $\mathfrak{o}$, S computes $\sigma = (\theta_1 = E_1 \cdot {E_3}^{-y},\ \theta_2 = E_2)$ as a signature on a message M.

6.6.2 Instantiation from LOSSW Scheme

In this section, we present the instantiation of our generic construction scheme in the standard model. The scheme works as follows.

Setup: *Setup* works in the same way as the LOSSW $VES.Setup$.

TKeyGen: *TKeyGen* works in the same way as the LOSSW $VES.KeyGen$. Let $sk_{TP} = y, pk_{TP} = Y = g_1^y$ be a private key and a public key of the semi-trusted third party, respectively.

SKeyGen: Run the LOSSW $VES.KeyGen$ to obtain $sk_S = x, pk_S = \mathbf{X} = \hat{e}(g,g)^x$ as a private key and a public key of the signer, respectively.

Sign: Assume that communication between the parties is secure. Given a message M, a list of signers $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$ and a private key sk_{S_i}, a signer S_i, where $i \in \{1, ..., n\}$, processes the *Sign* protocol as follows.

- Round 1: Each signer randomly selects γ_i and exchanges it with one another. Each signer generates $sk_R = \gamma = \prod_{i=1}^{n} \gamma_i, pk_R = R = g_1^{\gamma}$ as a shared random secret key and a shared random public key, respectively.
- Round 2: On input $sk_R, pk_R, sk_{S_i}, pk_{S_i}, pk_{TP}$, a signer S_i randomly selects r, r_1, r_2 and computes $\theta_1 = g^{x_i} u_0 \prod_{i=1}^{k} u_i^{m_i},\ \ \theta_2 = g^r$. Then compute $E_1 = \theta_1 \cdot R^{r_1} Y^{r_2},\ \ E_2 = \theta_1,\ \ E_3 = g^{r_1},\ \ E_4 = g^{r_2}$. Let $\mathfrak{o}_i = (E_{1,\,i},\ E_{2,\,i},\ E_{3,\,i},\ E_{4,\,i})$. S_i then sends $\Upsilon_i = (\mathfrak{o}_i, pk_R)$ to TP.
- Round 3: Upon receiving $\Upsilon_1, ..., \Upsilon_n$, TP first check whether pk_R in each $\Upsilon_1, ..., \Upsilon_n$ are the same. Next, check whether $\forall i \in \{1, ..., n\}$:

$$\hat{e}(E_{1,\,i}, g) \cdot \hat{e}(E_{2,\,i}, u_0 \prod_{j=1}^{k} u_j^{m_j})^{-1} \cdot \hat{e}(E_{3,\,i}, R)^{-1} \cdot \hat{e}(E_{4,\,i}, Y)^{-1} \stackrel{?}{=} \mathbf{X}_i$$

and let $\lambda_i = (K_{1,\ i} = E_{1,\ i} \cdot E_{4,\ i}^{-y},\ K_{2,\ i} = E_{2,\ i},\ K_{3,\ i} = E_{3,\ i})$. Then check whether

$$\hat{e}(K_{1,\ i}, g) \cdot \hat{e}(K_{2,\ i}, u_0 \prod_{j=1}^{k} u_i^{m_j})^{-1} \cdot \hat{e}(K_{3,\ i}, R)^{-1} \stackrel{?}{=} \mathbf{X}_i$$

holds or not. Finally, if the above holds then TP outputs a vector $\bar{\lambda} = \{\lambda_1, ..., \lambda_n\}$ and sends it to all the signers.

- Extract the multi-signature: each signer computes $\forall i \in \{1, ..., n\} : \sigma_i = (\theta_i = K_{1,\ i} \cdot K_{3,\ i}^{-\gamma},\ \phi_i = K_{2,\ i})$. Finally, each signer computes a multi-signature on message M, which is $\Theta = (\theta = \prod_{i=1}^{n} \theta_i,\ \ \phi = \prod_{i=1}^{n} \phi_i)$.

Verify: Given $\mathfrak{LS} = \{pk_{S_1}, ..., pk_{S_n}\}$, Θ and a message M, a verifier V checks whether

$$\hat{e}(\theta, g) \cdot \hat{e}(\theta, u_0 \prod_{j=1}^{k} u_j^{m_j})^{-1} \stackrel{?}{=} \prod_{j=1}^{n} \mathbf{X}_j$$

holds or not. If the equation holds, then V accepts the signature. Otherwise, V rejects it.

6.7 Conclusion

In this chapter, the notion of fair multi-signature schemes was proposed to capture the need for fairness in multi-signature schemes. In other words, the authenticity of the message produced by signers is fairly distributed. By enabling the fairness property, where all honest signers can be assured of the authenticity of the message and the distribution of the multi-signature, the notion of fair multi-signature schemes bridges the gap between theory and practice. Using the existing verifiably encrypted signatures, a generic construction of FMS schemes and its instantiations in the random oracle model and the standard model have been provided. We have presented a security model for fair multi-signature schemes and proofs of their security for generic construction. On the whole, if readers give the matter careful consideration, it is not difficult to come to the conclusion that this cryptographic primitive is useful in many practical applications, in particular those that involve a multi-party signing process.

Chapter 7

Identification Schemes

In this chapter, a new primitive algorithm called "escrowed deniable identification" is described. Part of this chapter appeared in SecTech'09 [THS+09], International Journal of Security and Its Application [THS+10] and WISA'09 [TSM09a]. Additionally, we also contribute to the notion of identity-based identification scheme that is secure against concurrent-reset attacks in the standard model and efficient identity-based identification that is secure against passive attack in the standard model [TSM09a].

7.1 Introduction

Invented by Fiat and Shamir in [FS86], an identification scheme allows a party called a prover to prove his/her identity to another party called a verifier; hence, it is a scheme that prevents impersonation. There are many types of settings for identification schemes, such as a public key setting and a symmetric setting. In this chapter, we are interested in a public key setting, where the prover holds the private key for proving himself/herself and the verifier holds the corresponding public key for verifying the prover identity during the protocol. Generally, the prover's public key appears to be a random bit string; however, it is generated after the associated private key has been selected. Nevertheless, an identity-based identification scheme allows a prover to select a public key that represents the identity of the prover, such as an email address. An example of an identity-based identification's application would be where a web service can determine from the user's identity whether to grant or deny an access to a resource.

It is essential to consider the security of identity-based identification schemes against active and concurrent attacks. For example, a well-known attack that is the man-in-the-middle attacks is the alternative name of the active and concurrent

attacks. In this attack, an adversary pretends to be a verifier and interacts with the honest prover before impersonating the honest prover. Moreover, the device used for executing the interactive protocol is often a resettable device such as Personal Computer (PC), whereby a computer can be reset to *any* state other than the initial state. Bellare, Fischlin, Goldwasser and Micali [BFGM01] formalised concurrent-reset attacks where an adversary has the power to reset the prover to the initial state and obtains information that leads to the associated private key before attempting to impersonate. A smart card is vulnerable to the reset attack, as mentioned in [BFGM01] and [CGGM00]. To reset a smart card's state to the initial state, an adversary only has to disconnect and reconnect its power source. Based on the study by Canetti, Goldwasser, Goldreich and Micali in [CGGM00] and the security analysis in [BFGM01], reset attacks play an important part in the security of (identity-based) identification protocols.

The need for privacy-preserving techniques has become essential due to increasing concern about the erosion of privacy in our society. Consider the following scenario in today's society. These days, politicians' privacy is threatened by *paparazzi*, who always relentlessly shadow them in their public and private activities. Suppose that such a politician would like to enter a building that is equipped with a smart card identification system. In this case, the politician will act as the prover and the smart card reader is the verifier. If the identification system employed does not ensure deniability, then as soon as the politician has identified himself to the system, the smart card verification system can be used to convince a *paparazzo* that the politician is indeed in the building. In the worst case scenario, this could be used by a terrorist for malicious purposes. Therefore, it is clear that it is essential to equip identification schemes with a deniability property. Generally, an identification scheme based on a zero knowledge protocol provides the deniability property. The idea behind deniability is to allow a transcript of the identification protocol to be simulated by any verifier. Therefore, there is no an actual evidence that the identification protocol has been executed between a prover and a verifier. On the other hand, if a terrorist enters the building with malicious intent, then there is no evidence to seize the terrorist. Hence, identification schemes also should be equipped with an escrowed deniability property. Let us consider another scenario where a verifier needs evidence to prove the existence of a transcript of a conversation between the prover and himself. Let Peggy be an online software seller and let Victor be an online software buyer. Assume that Peggy has sold some software to Victor through

the internet. Later, Victor has been caught using illegal software that he bought from Peggy. Victor seems to be a victim of Peggy's crime since he didn't know that the software that he bought from her is illegal software. In order to prove that he is innocent, Victor has to provide a proof of the money transaction for this software and the transcript of a conversation between himself and Peggy for that transaction. Why is the transcript of a conversation needed? The money transaction could have been conducted off-line and hence it might not be traceable to Peggy. Also, Peggy could deny the relationship between the money transaction and the illegal software, since the transcript of a conversation has ceased to exist. Therefore, a proof of the transcript of a conversation is needed to verify the co-existence of money transaction and the conversation between Victor and Peggy. The example above shows that an escrowed deniability property for an identification scheme based on a zero knowledge protocol is required. In our escrowed deniable identification scheme, a trusted party that can produce evidence to prove that a prover has participated in the generation of the identification transcript with a verifier is introduced to achieve both deniability and escrowed deniability. Moreover, the verifier alone cannot create a proof of the generation of the identification transcript. In the following subsections, a review of related works and a summary of our contribution are given to support this concept.

7.1.1 Related Work

Bellare, Fischlin, Goldwasser and Micali [BFGM01] provided four paradigms for constructing an identification protocol that would be secure under reset attack. The first three paradigms are based on cryptographic primitives, which are stateless signature schemes, encryption schemes and a combination of trapdoor commitment schemes and standard identification protocols secured against non-resetting attacks. The fourth paradigm is based on the resettable zero knowledge proof of membership, which was introduced in [CGGM00]. However, they did not provide a solution against the reset attack for identity-based identification. The well-known identity-based identification schemes, which were proposed by Kurosawa and Heng [KH05], are only secure under passive and concurrent-active attacks in the standard model. An identity-based identification scheme that is secure against reset attacks does not exist yet prior to this work.

For the deniability property, Dwork, Naor and Sahai were the first to introduce

the concept of deniability in authentication [DNS98]. Later in Crypto'03 [Pas03], Pass gave a formal definition of the deniable zero knowledge. Later, the deniability in authentication and key exchange were proposed in [RGK06, RG05].

For the 'escrow' property, it has been comprehensively studied in other areas, such as the "verifiable escrowed signatures" by Mao [Mao97], and the "escrowed linkability of ring signatures" by Chow, Susilo and Yuen [CSY06]. Moreover, the recent notion of 'ambiguous fair exchange' by Huang et al. [HYWS08] shares some commonalities with our notion of escrowed deniable identification. However, to transform these studies so as to obtain an escrowed deniable identification scheme is not a trivial matter.

The following works are used as tools to construct escrowed deniable identification scheme presented in this chapter. A publicly verifiable encryption (fair exchange) protocol was first proposed in Eurocrypt'98 by Asokan, Shoup and Waidner [ASW98]. A signature is encrypted with a trusted third party's public key. This encrypted signature is fairly exchanged and verifiable before each party reveals their decrypted signature in the last round. Any party can request the trusted third party to reveal the signature when another party is malicious. This concept is applied in escrowed deniable identification schemes to achieve deniable and openable proof of the transcript.

The second technique used to construct escrowed deniable identification schemes is a technique that transforms a weakly unforgeability secured signature scheme into a fully unforgeability secured signature scheme in the standard model proposed by Huang et al. [HWLZ08] in 2008. Their technique is to use a strong one-time signature to sign a message concatenated with a regular signature that signs a one-time public key. This eliminates the use of the random oracle in the proof of security of the signature scheme. Hence, with the technique above we can construct an escrowed deniable identification scheme and prove its security in the standard model.

7.1.2 Our Contribution

Our contribution is divided into two parts in this chapter. The first part outlines our contribution towards the identity-based identification scheme. For our contribution to the security model for identification and identity-based identification scheme, we give a definition called "$\mathsf{CR1}^{+}$", which improves a definition of $\mathsf{CR1}$. Compared to $\mathsf{CR1}$, our definition improves an attacker's power. In the $\mathsf{CR1}$ attack model, an

adversary is only allowed to reset the state of a prover, including its clones, to its initial state. In contrast, in the $\mathsf{CR1}^+$ attack model, an adversary is allowed to reset the state of a prover, including its clones, to any state that he/she chooses. Hence, we claim that $\mathsf{CR1}$ is a special case of $\mathsf{CR1}^+$. It is obvious that a $\mathsf{CR1}^+$ attacker's execution time is less than a $\mathsf{CR1}$ attacker's execution time. Moreover, in terms of our contribution to identity-based identification, we are the first to propose an identity-based identification scheme that is provably secure against impersonation attack under concurrent-reset attacks named $\mathsf{CR1}$ as well as the concurrent-reset attacks named $\mathsf{CR1}^+$. Based on the q-Strong Diffie-Hellman (SDH) assumption in [BB04] (where q equals to 2), our scheme is provable as secure under a $\mathsf{CR1}^+$ attack.

The second part of this chapter presents our contribution towards escrowed identification. The notion of escrowed deniable identification schemes was first introduced in [THS+09, THS+10]. In this notion, the identity of a prover is protected from being revealed to the public by a verifier. Meanwhile, if the prover misbehaves, then a trusted party can non-interactively reveal the prover's identity from an identification transcript. The transferability property of our primitive provides for the verifier to interactively prove to another party that the prover has identified himself to the verifier. This interactive proof is undertaken without revealing the prover's signature, which the verifier possesses. The formal security definitions for escrowed deniable identification schemes are also provided. These include impersonation, deniability and transferability/escrowed deniability. Moreover, based on certain standard number-theoretic assumptions, a concrete and efficient construction of an escrowed deniable identification scheme and its security proof in the standard model are proposed.

Chapter Organisation

The rest of the chapter is organised as follows. In Section 7.2, we define the identity-based identification scheme model, types of attack, the security of identity-based identification scheme under the passive and the concurrent-reset attack settings. A secure identity-based identification scheme under the passive attacks and its security proof is presented in Section 7.3. In Section 7.4, an identity-based identification scheme secure against impersonation under the concurrent-reset attacks with its proof of security and its experiment is presented. Next, we present the comparison of our presented schemes with the state-of-the-art of identity-based identification schemes in the literature in Section 7.5. In Section 7.6, we present the definition

of the escrowed deniable identification scheme and its security model. The concrete construction scheme of an escrowed deniable identification scheme and its security analysis present in Section 7.6 and Section 7.8. Finally, we conclude this chapter.

7.2 Definition of Identity-based Identification Scheme

In this section, we present a definition of identity-based identification schemes and their security models as outlined below.

7.2.1 Outline of Identity-based Identification Schemes

In an identity-based identification scheme (IBI-scheme), the algorithms can be classified into two PPT algorithms and one interactive protocol as follows.

1. **Key Generation ($KeyGen$):**
   Take a security parameter 1^ℓ as input and generate a pair of a public parameter pk_K and a master private key sk_K. That is, $KeyGen(1^\ell) \rightarrow (pk_K, sk_K)$.

2. **Key Extraction ($Extract$):**
   Given the identity of a prover ID and the master private key sk_K, $Extract$ takes ID and sk_K as input and computes a witness instance (a prover's private key) sk_P and gives it to the prover. That is, $Extract(ID, sk_K) \rightarrow sk_P$.

3. **Identification Protocol ($\langle P, V \rangle$):**
   A canonical protocol of an identification scheme can be denoted by $CID = (Commit, Challenge, Response, Check)$, where $Commit$, $Challenge$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier:

   - Step 1. P chooses r at random from a certain domain and computes $x = Commit(r)$. P then sends x to V.
   - Step 2. V chooses a challenge c at random from a certain set and sends $Challenge = c$ to P.
   - Step 3. P computes a response $y = Response(pk_K, sk_P, x, c)$ and sends y to V.
   - Step 4. V checks if $x = Check(pk_K, ID, x, c, y)$. V accepts P if only the prior equation holds.

The above protocol (P, V) in both standard and identity-based identification schemes is often called a canonical protocol. We say that (x, c, y) is a valid transcript for pk_K if it satisfies the equation in the step 4 as specified above. Observe that most identification schemes are transformable from or to digital signature schemes [KH04, KH05]. This is a fast track to construct an identification scheme, but nonetheless these schemes are insecure against concurrent-active or reset attacks.

7.2.2 Security of Identity-based Identification Schemes against Impersonation under Passive Attack

An *imp-pa* adversary $\mathcal{A} = (\hat{U}, \hat{P})$ is a pair of randomised polynomial-time algorithms that consist of the cheating identity-based user and cheating prover, respectively. We consider the following game that comprises three phases:

Phase 1: KGC runs on input 1^ℓ to produce a master public key and private key (pk_K, sk_K). A random tape and a master public key are given to $\hat{U}$. Then, it interacts with KGC initialised with pk_K, sk_K. We can define precisely that $Extract$ is a function of KGC that takes an incoming user's identity ID, pk_K, sk_K, and the current state, and it returns a user's private key sk_P associated with pk_K and the identity of a user ID. Therefore, the cheating identity-based user $\hat{U}$ can issue a request for the form (ID_i, i) where i is an index of each request form. As a result, the operation $(sk_{Pi}) \leftarrow Extract(ID_i, sk_K)$ is executed and sk_{Pi} is returned to $\hat{U}$ as the private key associated with the user's identity ID_i. These requests can be arbitrarily interleaved and the next chosen user's identity ID_{i+1} may be relevant for KGC public key and/or the previous user's identities $(ID_1, ..., ID_i)$ and the user's secret keys $(sk_{P1}, ...sk_{Pi})$.

Eventually, $\hat{U}$ outputs the user's identity ID_*, which $\mathcal{A}$ decides to impersonate. This ends the first phase. Note that it is not critical whether or not $\mathcal{A}$ can output sk_{P*}. The most important thing is that $\mathcal{A}$ can impersonate the user's identity ID_* so that the game can continue to the next phase.

Phase 2: $\mathcal{A}$ makes a request for conversation transcripts between the honest prover (the user's identity ID_*) and the honest verifier. A fresh random tape R is chosen for the honest verifier and the prover ID_*. Let $\mathtt{St}_i = (pk_K, ID_*, R, i)$ be the state of information of each request and let each transcript contain $\{Commit_i, Ch_i, Response_i, Check_i\}$ where Ch_i is a random challenge number for each request. These requests can be arbitrarily interleaved and, eventually, $\mathcal{A}$ outputs a set of state information

$\mathtt{St}$ and then stops, ending the second phase.

Phase 3: $\mathcal{A}$ now acts as a cheating prover $\hat{P}$, which attempts an impersonation on ID_*. $\mathcal{A}$ is initialised with $\mathtt{St}$, the verifier V is initialised with pk_K, ID_* and freshly chosen states (or coins) and $\hat{P}$ interacts with V. We say that adversary $\mathcal{A}$ wins if V accepts this interaction. The *imp-pa* advantage of $\mathcal{A}$, denoted by $\mathrm{Adv}^{imp-pa}_{IBI,\mathcal{A}}(\ell)$, is the probability that $\mathcal{A}$ wins, having taken over the coins of ℓ, the coins of $\hat{U}$, the coins of the prover ID_*, and the coins of V. We say that the IBI is secure against impersonation under a passive-reset attack ($IMP\text{-}PA$ secure) if the function $\mathrm{Adv}^{imp-pa}_{IBI,\mathcal{A}}(.)$ is negligible for all *imp-pa* adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter ℓ. Furthermore, the time of $\mathcal{A}$ is defined as the execution time of the entire three-phase game, including the time taken by the key generation, extraction, initialisation and computation of all the queries in each phase.

7.2.3 Security of Identity-based Identification Schemes against Impersonation under $\mathsf{CR1}^+$ Attack

An *imp-cra* adversary $\mathcal{A} = (\hat{U}, \hat{V}, \hat{P})$ is a triple of randomised polynomial-time algorithms, a cheating identity-based user, a cheating verifier and a cheating prover, respectively. The first phase is identical to the *imp-pa* game, and in this section, we consider the last two phases of the game as follows.

Phase 2: $\mathcal{A}$ now plays the role of a cheating verifier $\hat{V}$. To initialise the setting for this phase, a resettable random tape R_s is first provided to the prover ID_*. The cheating verifier $\hat{V}$ can then issue either a request for the form $(\mathtt{ch}, j, 0)$ for the *initial* state or the form $(\mathtt{ch}, j, i)$ for any other state i, where $\mathtt{ch}$ is a chosen challenge number and j is the index of the prover (or clone) that will be reset. For the form $(\mathtt{ch}, j, 0)$, the initial state $\mathtt{St}_{j,0}$ of the clone j is set to (pk_K, ID_*, R_s), the operation $(Commit_{out}, Resp_{out}) \leftarrow P(0, \mathtt{St}_{j,0}, \mathtt{ch})$ is executed, $(Commit_{out}, Resp_{out})$ is returned to $\hat{V}$, and remains at the $\mathtt{St}_{j,0}$ state. In the other form $(\mathtt{ch}, j, i)$, $\hat{V}$ can issue a request for the form $(\mathtt{ch}, j, i)$ where the selected challenge number $\mathtt{ch}$ is sent to the j-th clone with the chosen current state at i. The j-th clone computes $(Commit_{out}, Resp_{out}) \leftarrow P(i, \mathtt{St}_{j,i}, \mathtt{ch})$ and returns $(Commit_{out}, Resp_{out})$ to $\hat{V}$, and it also remains at the $\mathtt{St}_{j,i}$ state. These requests can be arbitrarily interleaved and $\hat{V}$ may eventually output a set of state information $\mathtt{St}$ and then stop, ending the second phase.

Phase 3: $\mathcal{A}$ now plays the role of a cheating prover $\hat{P}$. $\hat{P}$ is initialised with $\mathtt{St}$ (note

that it is not compulsory to give $\hat{P}$ separate coins, or even pk_K, since these coins can be obtained from $\hat{V}$ from the previous phase via $\mathtt{St}$.), the honest verifier V is initialised with pk_K, ID_* and a freshly chosen coin, and $\hat{P}$ interacts with V. We say that adversary $\mathcal{A}$ wins if V accepts this interaction. The *imp-cra* advantage of $\mathcal{A}$, denoted by $\mathrm{Adv}^{imp\text{-}cra}_{IBI,\mathcal{A}}(\ell)$ is the probability that $\mathcal{A}$ wins, having taken over the coins of ℓ, the coins of $\hat{V}$, the coins of the prover clones, and the coins of V. It is said that the IBI is secure against impersonation under a $\mathsf{CR1}^+$ attack ($IMP\text{-}CRA$ secure) if the function $\mathrm{Adv}^{imp\text{-}cra}_{IBI,\mathcal{A}}(.)$ is negligible for all *imp-cra* adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter ℓ. Moreover, the time of $\mathcal{A}$ is defined as in the passive attack model.

7.3 Identity-based Identification Schemes against Impersonation under Passive Attack ($IBI\text{-}PA$)

In this section, we first present our scheme that resists impersonation under passive attack. Then, in the next section, we modify our scheme to resist against impersonation under $\mathsf{CR1}^+$ attack. The scheme is described as follows.

1. **Key Generation ($KeyGen$):**
   Let $(\mathbb{G}_1, \mathbb{G}_T)$ be two multiplicative cyclic groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$ for some prime p. g is a generator of $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is a bilinear pairing function. Given a security parameter 1^ℓ, which is a positive integer, $KeyGen$ works as follows.

   - Run the public parameter generator to obtain the system parameter $\mathsf{param} = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, p, g\}$.
   - Select random numbers $\alpha, a, c, \tau \in \mathbb{Z}_p^*$ and compute $P_{Pub} = \hat{e}(g,g)^\alpha$, $A = g^a$, $B = g^\tau$, $C = g^c$, $D = g^{a \cdot c}$, $E = g^{\tau \cdot c}$.

   In this case, a pair of private key and public parameter of KGC is generated where $pk_K = (\mathbb{G}_1, \mathbb{G}_T, \hat{e}, p, g, P_{pub}, A, B, C, D, E)$ is the set of public parameters and $sk_K = (\alpha, a, c, \tau)$ is the KGC's master private key.

2. **Key Extraction ($Extract$):**
   Given the identity of a prover ($ID \in \mathbb{Z}_p^*$), the public parameter pk_K and

the master private key sk_K, $Extract$ takes ID and (α, a, c, τ) as input and computes a user P's witness instance sk_P as follows.

- U_i chooses random integers $k, n \in \mathbb{Z}_p^*$ such that $(ID + n \cdot a + \tau) \bmod p \neq 0$ and $k \neq a^{-1}$ and computes $\varpi_1 = g^{\frac{\alpha}{(ID+n\cdot a+\tau)}}$, $\varpi_2 = g^{k\cdot a}$, $\varpi_3 = g^k$, $\varpi_4 = k + ID \cdot c$, $\varpi_5 = g^{k\cdot\tau}$.

After this point, the user's public key ID and the user's private key (a witness instance) $sk_P = (\varpi_1, \varpi_2, \varpi_3, \varpi_4, \varpi_5, n)$ are provided to the prover.

3. **Identification Protocol ($\langle P, V\rangle$):**
A canonical protocol of an identity-based identification scheme can be denoted by CID =($Commit$, $Challenge$, $Response$, $Check$), where $Commit$, $Challenge$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier:

- Step 1. P chooses a random integer $r \in \mathbb{Z}_p^*$ and computes $Commit(\varpi_2, \varpi_3, \varpi_5, n, r) = (\varpi_2, \varpi_3, \varpi_5, n, r)$, and then sends $(\varpi_2, \varpi_3, \varpi_5, n, r)$ to V.
- Step 2. V chooses a random challenge integer $ch \in \mathbb{Z}_p^*$ and sends $Challenge = ch$ to P.
- Step 3. P computes a response $Response(\varpi_1, \varpi_4, n, r, ch) = z$, where $z = \varpi_1^{\frac{1}{r\cdot\varpi_4+ch}} \in \mathbb{G}_1$, and sends z to V.
- Step 4. V checks if $P_{pub}(= \hat{e}(g,g)^\alpha) \equiv \hat{e}(z, \varpi_2^{n\cdot r} \cdot \varpi_3^{ID\cdot r} \cdot A^{n\cdot ch} \cdot C^{r\cdot ID^2} \cdot D^{n\cdot r\cdot ID} \cdot g^{ID\cdot ch} \cdot \varpi_5^r \cdot E^{r\cdot ID} \cdot B^{ch})$.
  $Check(pk_K, ID, \varpi_2, \varpi_3, \varpi_5, n, r, ch, z) = 1$ if the above holds.
- Step 5. V accepts P only if $Check$ holds.

7.3.1 An Experiment on Identity-based Identification Schemes against Impersonation under Passive Attack

Given a random tape R, let $\hat{U}$ denote a cheating user that issues a request form (ID_i, i) to KGC, and let KGC return with a private key sk_{Pi} associated with a public key ID_i. Let $\mathcal{A}$ be an adversary that can break the $IBI\text{-}PA$ identity-based identification scheme with an advantage $\mathrm{Adv}^{imp-pra}$ and let $\mathcal{A}$ play a role in the following game as $\hat{U}$, an eavesdropper and $\hat{P}$ in the associated phases as defined earlier. Let an adversary's algorithm $\mathcal{S}$, attempting to solve the q-SDH problem

with $h, h^x, h^{x^2}, ..., h^{x^q}$ as input and output $(h^{\frac{1}{x+r}}, r)$, be the challenger. $\mathcal{S}$ runs the experiment as follows.

Experiment: Adversary $\mathcal{S}(h, h^x, h^{x^2}, ..., h^{x^q})$

Since $\mathcal{A}$ can make a request for a transcript of the conversation for at most q_{IP} queries, where $q_S < q$, we may then assume that $\mathcal{A}$ issues exactly $q-1$ queries. If the actual number of requests is less, we can always virtually reduce the value of q so that $q = q_S + 1$. However, $\mathcal{A}$ is required to reveal a number of queries up front. $\mathcal{S}$ randomly chooses integer $l_1, ..., l_{q-1} \in \mathbb{Z}_p^*$.

Let $f(y)$ be the polynomial $f(y) = \prod_{i=1}^{q-1}(y+l_i)$. Reform $f(y)$ by expanding it to $f(y) = \sum_{i=0}^{q-1} \beta_i y^i$ where $\beta_0, ..., \beta_{q-1} \in \mathbb{Z}_p$ are the coefficients of the polynomial $f(y)$. Let $K_1, ..., K_q$ denote as $h^x, h^{x^2}, ..., h^{x^q}$. Compute as follows.

$$g \leftarrow \prod_{i=0}^{q-1}(K_i)^{\beta_i} = h^{f(x)} \in \mathbb{G}_1 \text{ and } g^x \leftarrow \prod_{i=1}^{q}(K_i)^{\beta_{i-1}} = h^{xf(x)} \in \mathbb{G}_1.$$

$\mathcal{S}$ randomly chooses an integer $a, c, \alpha, \tau \in \mathbb{Z}_p^*$. Then, let $\mathcal{S}$ set $A = g^a$, $B = g^\tau$, $C = g^c$, $D = g^{a.c}$, $E = g^{\tau \cdot c}$ and $P_{Pub} = \hat{e}(g, g)^\alpha$ be a public key $pk_K = (g, A, B, C, D, E, P_{Pub})$. Initialise with (pk_K, R) and set i to 0. Let $\mathfrak{ID}$ be the list of user's identities that $\mathcal{A}$ makes a request for a private key associated with a public key pk_K and user identity ID.

Phase 1: $\mathcal{S}$ answers $\mathcal{A}$'s requests as follows. $\mathcal{A}$ issues a request for the form (ID_i, i) where ID_i is an adaptively chosen integer in $\mathbb{Z}_p^*$ by $\mathcal{A}$. Without losing generality, assume that $\mathcal{A}$ will never issue a request for the same user's identity again. $\mathcal{A}$ then sends a query to $\mathcal{S}$. Then, $\mathcal{S}$ computes as follows: set $i \leftarrow i + 1$ and then $\mathcal{S}$ randomly selects integers $n_i, k_i \in \mathbb{Z}_p^*$ such that $(ID_i + n_i \cdot a + \tau) \bmod p \neq 0$ and $k_i \neq a^{-1}$ and computes $\varpi_{i,1} = g^{\frac{\alpha}{(ID_i + n_i \cdot a + \tau)}}$, $\varpi_{i,2} = g^{k_i \cdot a}$, $\varpi_{i,3} = g^{k_i}$, $\varpi_{i,4} = k_i + ID_i \cdot c$, $\varpi_{i,5} = g^{k_i \cdot \tau}$ and keeps $(\varpi_{i,1}, \varpi_{i,2}, \varpi_{i,3}, \varpi_{i,4}, \varpi_{i,5}, k_i, n_i, ID_i)$ in the $\mathfrak{ID}$. $\mathcal{S}$ then returns $(\varpi_{i,1}, \varpi_{i,2}, \varpi_{i,3}, \varpi_{i,4}, \varpi_{i,5}, n_i)$ as a private key associated with the public key ID_i to $\mathcal{A}$.

Phase 2: Eventually, after at most $q_{ID} < (p-2)$ queries, $\mathcal{A}$ outputs ID^* as the public key, which $\mathcal{A}$ will attempt to impersonate. Then $\mathcal{A}$ now

acts as an eavesdropper that can issue a request for a transcript of the conversation between the prover ID_* and the honest verifier.

If ID_* is in the $\mathfrak{ID}$, then $\mathcal{S}$ terminates and returns failure. Otherwise, initialise $\mathcal{A}$ with $(pk_P(=(pk_K, ID_*)), R)$; set i to 0; $\mathcal{S}$ selects a random integer $n \in \mathbb{Z}_p^*$ such that $\tau+n\cdot a+ID_* \neq 0$ and then constructs the partial ID_*'s witness instance as follows: $\varpi_{*,2} = g^{k\cdot a} = g^{a\cdot x}$, $\varpi_{*,3} = g^k = g^x$, $\varpi_{*,5} = g^{k\cdot\tau} = g^{\tau\cdot x}$. For the above context, k is x. Let $\mathfrak{TC}$ be the list of conversation transcripts that $\mathcal{A}$ has requested and it is initialised with an empty list. $\mathcal{S}$ then answers $\mathcal{A}$'s requests as follows: $\mathcal{A}$ issues a request for a transcript of the conversation between the user ID_* and the honest verifier and then $\mathcal{S}$ performs as follows. set $i \leftarrow i+1$ and then $\mathcal{S}$ randomly chooses $r_i \in \mathbb{Z}_p^*$ such that r_i is not contained in the $\mathfrak{TC}$. If $i \geq q$, then $\mathcal{S}$ terminates and returns failure. $\mathcal{S}$ sets $(\varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r_i)$ as a commitment. $\mathcal{S}$ computes $ch_i = r_i(l_i - c \cdot ID_*)$ as a challenge number. $\mathcal{S}$ must generate a response with $g^{\frac{1}{x+l_i}}$. To do so, let $f_i(y)$ be the polynomial $f_i(y) = \frac{f(y)}{(y+l_i)} = \prod_{j=1, j\neq i}^{q-1}(y+l_j)$. As before, we reform $f_i(y)$ into $f_i(y) = \sum_{j=0}^{q-2} \mathbb{y}_j y^j$. Compute $g^{\frac{1}{x+l_i}} \leftarrow \prod_{i=0}^{q-2} K_j^{\mathbb{y}_i} = h^{\frac{f(x)}{(x+l_i)}} \in \mathbb{G}_1$. Let $z_i = \overset{\alpha}{g^{\frac{}{r_i(\tau+n\cdot a+ID_*)\cdot(x+l_i)}}}$ be the response. If z_i was in the $\mathfrak{TC}$, then $\mathcal{S}$ terminates and returns failure. $\mathcal{S}$ then keeps (r_i, ch_i, z_i) in the $\mathfrak{TC}$ and returns $(\varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r_i, ch_i, z_i)$ as a conversation transcript to $\hat{V}$, until $\hat{V}$ outputs the state information $\mathtt{St}$ on at most $q_{IP} < q$ queries and then stops.

Phase 3: Now, $\mathcal{A}$ changes its status to $\hat{P}$ and can no longer issue a request for a transcript of the conversation between the user ID_* and the honest verifier to $\mathcal{S}$.

$\hat{P}$ starts the impersonation process.

- $\hat{P}$ first randomly chooses an integer $r \in \mathbb{Z}_p^*$ and then sends $(\varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r)$ to V.
- $\mathcal{S}$ randomly selects an integer $ch \in \mathbb{Z}_p^*$ such that $l_* = (ch + r \cdot c \cdot ID_*)/r$ and $l_* \notin \{l_1, ..., l_{q-1}\}$ and sends ch to $\hat{P}$.
- $\hat{P}$ returns z_*.

Define $Check(pk_K, z_*, ch, \varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r, ID_*)$ as:

$$P_{pub} \stackrel{?}{=} \hat{e}(z_*, \varpi_{*,2}^{n \cdot r} \cdot \varpi_{*,3}^{ID_* \cdot r} \cdot A^{n \cdot ch} \cdot C^{r \cdot ID_*^2} \cdot D^{n \cdot r \cdot ID_*} \cdot g^{ID \cdot ch} \cdot \varpi_{*,5}^{r} \cdot E^{r \cdot ID_*} \cdot B^{ch}),$$

where $P_{pub} = \hat{e}(g, g)^{\alpha}$. If the equality holds, then the output of $Check(pk_K$, $z_*, ch, \varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r, ID_*)$ is 1. Otherwise, it outputs 0. If $Check(pk_K$, $z_*, ch, \varpi_{*,2}, \varpi_{*,3}, \varpi_{*,5}, n, r, ID_*) = 1$ then $\mathcal{S}$ computes the output as follows.

$$\begin{aligned} \text{Since } l_* &= \frac{r \cdot ID_* \cdot c + ch}{r}; \\ \text{hence, } Z &= z_*^{\frac{r(\tau + n \cdot a + ID_*)}{\alpha}} \\ &= g^{(\frac{\alpha}{(\tau + n \cdot a + ID_*)(r(x + ID_* \cdot c) + ch)})(\frac{r(\tau + n \cdot a + ID_*)}{\alpha})} \\ &= g^{\frac{1}{x + (ID_* \cdot c + ch/r)}}. \\ &= g^{\frac{1}{x + l_*}} \end{aligned}$$

Since $g = h^{f(x)}$, $Z = h^{\frac{f(x)}{x+l_*}}$. Let us denote by $Z^* = h^{\frac{1}{x+l_*}}$. Let $f_*(y)$ be the polynomial $f_*(y) = \frac{f(y)}{(y+l_i)}$ such that there exists some polynomial $\gamma(y) = \sum_{i=0}^{q-2} \gamma_i y^i$ and some $\gamma_{-1} \in \mathbb{Z}_p^*$. Then we reform $f_*(y)$ into $\gamma(y)$ as: $f(y) = \gamma(y)(y + l_*) + \gamma_{-1}$. The exponent of Z, where Z is $h^{\frac{f(x)}{x+l_*}}$, can then be written as $\frac{f(x)}{(x+l_*)} = \frac{\gamma_{-1}}{x+l_*} + \sum_{i=0}^{q-2} \gamma_i x^i$. Since $f(x) = \prod_{i=1}^{q-1}(x + l_i)$ and $l_* \notin \{l_1 ..., l_{q-1}\}$; hence, $(x + l_*)$ does not divide $f(x)$ and $\gamma_{-1} \neq 0$. Then $\mathcal{S}$ computes $Z^* \leftarrow \left(Z. \prod_{i=1}^{q-1} K_i^{-\gamma_i}\right)^{1/\gamma_{-1}} = g^{\frac{1}{(x+l_*)}}$. $\mathcal{S}$ wins the game and returns (Z^*, l_*) as the solution of q-SDH. Otherwise, $\mathcal{S}$ returns failure.

7.3.2 Proof of Security

Theorem 7.1 *Let $IBI = (KeyGen, Extract, P, V)$ be an IBI-PA identity-based identification scheme associated with the q-SDH assumption. Let $\mathcal{A} = (\hat{U}, \hat{P})$ be an imp-pa adversary of time complexity $\mathtt{t}(.)$ attacking IBI. Then there exists a q-SDH adversary $\mathcal{S}$ of time complexity $\mathtt{t}'(.)$ solving the q-SDH problem such that for every security parameter ℓ*

$$\begin{aligned} \mathrm{Adv}_{IBI,\mathcal{A}}^{imp-pa}(\ell) &\geq e^2 \cdot \mathrm{Adv}_B^{q-SDH}(\ell) \text{ or, put in a simple terms,} \\ \epsilon &\geq e^2 \cdot \epsilon', \end{aligned}$$

where e is the natural logarithm. Moreover, the time complexity $\mathtt{t}$ of $\mathcal{A}$ is $\mathtt{t} \leq \mathtt{t}' - ((6 + 3q_{ID} + q_{IP})C_G + 2C_P)$, where C_G is a computation time of group exponential

operation and C_P is a computation time of bilinear group pairing operation, and $\mathcal{A}$ can request queries at most $q_{ID} < p - 1$ queries and at most $q_{IP} < q$ queries.

Corollary 7.2 *If the $(q, \mathtt{t}', \epsilon')$-SDH assumption holds, then the IBI-PA identity-based identification scheme is $(q_{ID}, q_{IP}, \mathtt{t}, \epsilon)$-secure against impersonation under passive attack.*

Proof: The experiment $\mathcal{S}(h, h^x, h^{x^2}, ..., h^{x^q})$ in Section 7.3.1 shows that the $IBI\text{-}PA$ identity-based identification scheme that is $(\mathtt{t}, q_{ID}, q_{IP}, \epsilon)$-secure against impersonation under passive attack can be reduced to the $(q, \mathtt{t}', \epsilon')$-SDH problem. For any input $h, h^x, h^{x^2}, ...h^{x^q} \in \mathbb{G}_1$, $\mathcal{S}(h, h^x, h^{x^2}, ...h^{x^q})$ returns $(h^{\frac{1}{x+r}}, r)$ where these outputs are computed in one unit of time from outputs (z_*, l_*). From these outputs, it is suggested that the adversary $\mathcal{A}$ breaks the $IBI\text{-}PA$ identity-based identification scheme with the passive attack setting.

However, to show that the probability that the experiment $\mathcal{S}$ can succeed in solving the q-SDH problem with an advantage of at least ϵ', we analyse three events needed for $\mathcal{S}$ to succeed.

E1: $\mathcal{S}$ does not abort as a result of the queries in phase 1.

E2: $\mathcal{S}$ does not abort as a result of the queries in phase 2 and phase 3.

E3: $\mathcal{A}$ can impersonate ID^* with a (r, ch, z_*)-transcript where (r, z_*) is generated by $\mathcal{A}$ and ch is chosen by $\mathcal{S}$.

The probability that $\mathcal{S}$ will succeed, if all of the above events happen, is

$$\begin{aligned}\mathbf{Pr}[E1 \wedge E2 \wedge E3] &= \mathbf{Pr}[E1] \cdot \mathbf{Pr}[E2|E1] \cdot \mathbf{Pr}[E3|E1 \wedge E2] \\ &= \mathbf{Pr}[E1] \cdot \mathbf{Pr}[E2] \cdot \mathbf{Pr}[E3|E1 \wedge E2].\end{aligned}$$

The following claims give an upper bound of the probability for each of the above terms.

Claim 7.3 (1) *The probability that $\mathcal{S}$ does not abort as a result of the queries in phase 1 is at least $1/e$ where e is the natural logarithm. Hence, $\mathbf{Pr}[E1] \geq 1/e$.*

Proof: In general, we assume that $\mathcal{A}$ does not query a private key for the same identity twice. Subsequently, the only result of a user's private key queries that causes an abortion is a collision of responses in the $\mathfrak{ID}$, which $\mathcal{A}$ can use to compute

a KGC's private key. Therefore, using the inductive hypothesis to prove the above claim, the probability that, for each request for a user's private key, $\mathcal{S}$ does not abort as a result of a collision of a user's private key queries on the $\mathfrak{ID}$ is $1-1/(p-1)$. Consequently, the probability that $\mathcal{S}$ does not abort after issuing responses for a user's private key queries at most q_{ID} queries is at least $\prod_{i=1}^{q_{ID}}(1-1/(p-1))$. Therefore, since q_{ID} approached $p-2$, the probability that $\mathcal{S}$ does not abort as a result of all the private key queries is at least $(1-1/(p-1))^{p-2} \geq 1/e$. □

Claim 7.4 (2) *After phase 1 is complete, the probability that $\mathcal{S}$ does not abort as a result of the queries in phase 2 and phase 3 is at least $1/e$. Hence,* $\mathbf{Pr}[E2|E1] = \mathbf{Pr}[E2] = (1-1/(q_{IP}+1))^{q_{IP}} \geq 1/e$.

Proof: As a result of the queries in phase 1, $\mathcal{S}$ requires only one user's identity out of the $\mathfrak{ID}$ to challenge $\mathcal{A}$, where the limit of q_{ID} can be set at most $q_{ID} \leq p-2$ queries. Therefore, the results of the queries are not influenced by the conversation transcript queries in phase 2, which means $\mathbf{Pr}[E2|E1] = \mathbf{Pr}[E2]$.

For the conversation transcript queries in phase 2, it is assumed that $\mathcal{S}$ does not need to respond with the same challenge numbers twice and $\mathcal{A}$ must reveal the number of queries up front where $q_{IP} < q$. Thus, the only result of all the conversation transcript queries that causes a termination is a collision of the responses in the $\mathfrak{TC}$, which $\mathcal{A}$ can use to compute a private key. We then again use the inductive hypothesis to prove the above claim, where the probability that, for each query for a transcript of conversation, $\mathcal{S}$ does not abort as a result of a collision of the conversation transcript queries on the $\mathfrak{TC}$ is $1-1/(q_{IP}+1)$. Consequently, the probability that $\mathcal{S}$ does not abort after issuing a request for a conversation transcript at most q_{IP} queries is at least $\prod_{i=1}^{q_{IP}}(1-1/(q_{IP}+1))$. Therefore, since $q_{IP} = q-1$, the probability that $\mathcal{S}$ does not abort as a result of all the conversation transcript queries is at least $(1-1/q)^{q-1} \geq 1/e$.

For phase 3, $\mathcal{S}$ does not abort after $\mathcal{A}$ has identified itself as a user ID_*. Hence, the probability that $\mathcal{S}$ does not abort as a result in phase 3 is 1.

Therefore, the probability that $\mathcal{S}$ does not abort as a result of all the conversation transcript queries in phase 2 after phase 1 has finished is $\mathbf{Pr}[E2] = (1-1/(q_{IP}+1))^{q_{IP}} \geq 1/e$ for at most $q_{IP} < q$ queries. □

Claim 7.5 (3) *If algorithm $\mathcal{A}$ can impersonate a user ID^* after phases 1, 2 and 3 have finished, then algorithm $\mathcal{A}$'s view is identical to its view in a real attack. Hence,* $\mathbf{Pr}[E3|E1 \wedge E2] \geq \epsilon$.

Proof: The public key provided to $\mathcal{A}$ is obtained from the same distribution as the public key produced by the *KeyGen* algorithm. Responses to identity queries and conversation transcripts are the same as in a real attack. Moreover, each response for identity queries and conversation transcripts is uniformly and independently distributed in $\mathbb{Z}_p^*$ and all responses to queries are valid. Therefore, $\mathcal{A}$ impersonates an honest member with a probability of at least ϵ. Hence, $\mathbf{Pr}[E3|E1 \wedge E2] \geq \epsilon$. □
Using the bounds from all of the above claims, it is shown that $\mathcal{S}$ can generate a solution for the q-SDH problem with the probability of at least $\epsilon \cdot \frac{1}{e^2}$ as required. Moreover, $\mathcal{S}$'s running time is the same as $\mathcal{A}$'s running time, plus the time that it takes to respond to q_{ID} queries for the request of a private key associated with a public key (pk_K, ID) and to q_{IP} queries for the request of ID_*'s conversation transcripts. Let C_G be a computation time of group exponential operation and C_P be a computation time of bilinear group pairing operation. The total running time is consequently at most $\mathtt{t}' \geq \mathtt{t} + (6 + 3q_{ID} + q_{IP})C_G + 2C_P$ as required. Note that the computation of multiplication in the finite field order p is not taken into account. This completes the proof of Theorem 7.1. □

7.4 Identity-based Identification Scheme against Impersonation under $\mathsf{CR1}^+$ Attack (*IBI-CRA*)

In this section, we present our identity-based identification scheme, which is secure against $\mathsf{CR1}^+$ attack. The *KeyGen* phase is the same as the *IBI-PA* scheme from Section 7.3, and therefore it is omitted. We will describe the rest of the scheme as follows.

1. **Key Extraction (*Extract*):**
   Given an identity of the prover ($ID \in \mathbb{Z}_p^*$), the public parameter pk_K and the master private key sk_K, *Extract* takes ID and (α, a, c, τ) as input and computes an user P's private key (or a user P's witness instance) sk_P as follows.

   - U_i chooses random integers $v, k, n_1, n_2 \in \mathbb{Z}_p^*$ such that $(ID + n_1 \cdot a + \tau) \bmod p \neq 0$, $(ID + n_2 \cdot a + \tau) \bmod p \neq 0$, and $k \neq a^{-1}$ and computes $\varpi_1 = g^{\frac{\alpha - v}{(ID + n_1 \cdot a + \tau)}}$, $\varpi_2 = g^{\frac{v}{(ID + n_2 \cdot a + \tau)}}$, $\varpi_3 = g^{k \cdot a}$, $\varpi_4 = g^k$, $\varpi_5 = k + ID \cdot c$, $\varpi_6 = g^{k \cdot \tau}$.

Thereafter, the user's public key ID and the user's private key (a witness instance) $sk_P = (\varpi_1, \varpi_2, \varpi_3, \varpi_4, \varpi_5, \varpi_6, n_1, n_2)$ are provided to the prover.

2. **Identification Protocol ($\langle P, V \rangle$):**
A canonical protocol of an identity-based identification scheme can be denoted by $CID = (Commit, Challenge, Response, Check)$, where $Commit$, $Challenge$, $Response$ and $Check$ are PPT algorithms used in the following protocol, where P is the prover and V is the verifier:

- Step 1. P chooses a random integer $r \in \mathbb{Z}_p^*$ and computes $Commit(\varpi_3, \varpi_4, \varpi_6, n_1, n_2, r) = (\varpi_3, \varpi_4, \varpi_6, n_1, n_2, r)$ and then sends $(\varpi_3, \varpi_4, \varpi_6, n_1, n_2, r)$ to V.
- Step 2. V chooses random challenge integers ch_1, $ch_2 \in \mathbb{Z}_p^*$ and sends $Challenge = (ch_1, ch_2)$ to P.
- Step 3. P computes a response $Response(\varpi_1, \varpi_2, \varpi_5, n, r, ch_1, ch_2) = (z_1, z_2)$, where $z_1 = \varpi_1^{\frac{1}{ch_1 \cdot \varpi_5 + r}}$, $z_2 = \varpi_2^{\frac{1}{ch_2 \cdot \varpi_5 + r}} \in \mathbb{G}_1$, and sends (z_1, z_2) to V.
- Step 4. V checks if $P_{pub} (= \hat{e}(g, g)^{\alpha}) \equiv \hat{e}(z_1, \varpi_3^{n_1 \cdot ch_1} \cdot \varpi_4^{ID \cdot ch_1} \cdot A^{n_1 \cdot r} \cdot C^{ch_1 \cdot ID^2} \cdot D^{n_1 \cdot ch_1 \cdot ID} \cdot g^{ID \cdot r} \cdot \varpi_6^r \cdot E^{r \cdot ID} \cdot B^{ch_1}) \cdot \hat{e}(z_2, \varpi_3^{n_2 \cdot ch_2} \cdot \varpi_4^{ID \cdot ch_2} \cdot A^{n_2 \cdot r} \cdot C^{ch_2 \cdot ID^2} \cdot D^{n_2 \cdot ch_2 \cdot ID} \cdot g^{ID \cdot r} \cdot \varpi_6^r \cdot E^{r \cdot ID} \cdot B^{ch_2})$. Then, $Check(pk_K, ID, \varpi_3, \varpi_4, \varpi_6, n_1, n_2, r, ch_1, ch_2, z_1, z_2) = 1$ if the above equation holds.
- Step 5. V accepts P only if$Check$ is 1.

7.4.1 An Experiment on Identity-based Identification Schemes against Impersonation under CR1$^+$ Attack

Let $\hat{U}$ denote a cheating user who issues a request form (ID_i, i) to KGC. Given a random tape R and a resettable random tape R_s, KGC returns with a private key sk_{Pi} associated with a public key ID_i. Let $\mathcal{A}$ be an adversary that can break the $IBI\text{-}CRA$ identification scheme with an advantage $\mathrm{Adv}^{imp\text{-}cra}$ and let $\mathcal{A}$ play a role in the following game as $\hat{U}$, $\hat{V}$ and $\hat{P}$, respectively. Let an adversary's algorithm $\mathcal{S}$, attempting to solve the 2-SDH problem with h, h^x, h^{x^2} as input and output $(h^{\frac{1}{x+r}}, r)$, be the challenger. $\mathcal{S}$ executes the experiment as follows.

Experiment: Adversary $\mathcal{S}(h, h^x, h^{x^2})$

$\mathcal{S}$ first randomly chooses an integer $a, c, \alpha, \tau \in \mathbb{Z}_p^*$ and sets $g^x = h^{x^2}$, $g = h^x$, $g^{\frac{1}{x}} = h$. Then, $\mathcal{S}$ sets $A = g^a$, $B = g^\tau$, $C = g^c$, $D = g^{a.c}$, $E = g^{\tau.c}$ and $P_{Pub} = \hat{e}(g,g)^\alpha$. Let $pk_K = (g, A, B, C, D, E, P_{Pub})$ be public key. Perform the initialisation with (pk_K, R) and set i to 0. Let $\mathfrak{ID}$ be the list of user's identities that $\mathcal{A}$ makes a request for a private key associated with public key (pk_K, ID).

Phase 1: $\mathcal{S}$ answers $\mathcal{A}$'s requests as follows: $\mathcal{A}$ issues a request for the form (ID_i, i) where ID_i is an adaptively chosen integer in $\mathbb{Z}_p^*$ by $\mathcal{A}$. Without losing generality, assume that $\mathcal{A}$ will never issue a request for the same user's identity again. Then, $\mathcal{A}$ sends a query to $\mathcal{S}$. $\mathcal{S}$ responds as follows. Set $i \leftarrow i + 1$ and then $\mathcal{S}$ randomly selects integers $n_{i,1}, n_{i,2}, k_i, v_i \in \mathbb{Z}_p^*$ such that $(ID_i + n_{i,1} \cdot a + \tau) \bmod p \neq 0$ and $(ID_i + n_{i,2} \cdot a + \tau) \bmod p \neq 0$ and $k_i \neq a^{-1}$ and computes $\varpi_{i,1} = g^{\frac{\alpha - v_i}{(ID_i + n_{i,1} \cdot a + \tau)}}$, $\varpi_{i,2} = g^{\frac{v_i}{(ID_i + n_{i,2} \cdot a + \tau)}}$, $\varpi_{i,3} = g^{k_i \cdot a}$, $\varpi_{i,4} = g^{k_i}$, $\varpi_{i,5} = k_i + ID_i \cdot c$, $\varpi_{i,6} = g^{k_i \cdot \tau}$ and keeps $(\varpi_{i,1}, \varpi_{i,2}, \varpi_{i,3}, \varpi_{i,4}, \varpi_{i,5}, \varpi_{i,6}, k_i, n_{i,1}, n_{i,2}, ID_i)$ in the $\mathfrak{ID}$. $\mathcal{S}$ then returns $(\varpi_{i,1}, \varpi_{i,2}, \varpi_{i,3}, \varpi_{i,4}, \varpi_{i,5}, \varpi_{i,6}, n_{i,1}, n_{i,2})$ as a private key associated with the public key ID_i to $\mathcal{A}$.

Phase 2: Eventually after at most $q_{ID} < (p-2)$ queries, $\mathcal{A}$ outputs ID^* as the public key, which $\mathcal{A}$ will attempt to impersonate. $\mathcal{A}$ now acts as a cheating verifier $\hat{V}$. Simultaneously, $\mathcal{S}$ acts as the honest prover of user ID^*. If ID_* is in the $\mathfrak{ID}$, then $\mathcal{S}$ terminates and returns failure. Otherwise, initialise $\mathcal{A}$ with $(pk_P (= (pk_K, ID_*)), R)$; set i to 0; $\mathcal{S}$ selects random integers $n_1, n_2 \in \mathbb{Z}_p^*$ such that $\tau + n_1 \cdot a + ID_* \neq 0$ and $\tau + n_2 \cdot a + ID_* \neq 0$. $\mathcal{S}$ then constructs a partial ID_*'s witness instance as follows: $\varpi_{*,3} = g^{k \cdot a} = g^{a \cdot x}, \varpi_{*,4} = g^k = g^x$, $\varpi_{*,6} = g^{k \cdot \tau} = g^{\tau \cdot x}$. For the above context, k is x. Let $\mathfrak{TC}$ be the list of conversation transcripts that $\mathcal{A}$ has requested and it is initialised with an empty list. $\mathcal{A}$ also sets a counter for a number of provers (clones) $\nu = 0$. $\mathcal{S}$ then answers $\hat{V}$'s requests as follows.

- If $\hat{V}$ issues a request for the form $(\mathtt{ch}, j, 0)$, where $\mathtt{ch} \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and $j = \nu$ then $\mathcal{S}$ computes as follows: set $r_{0,j} \in R_s \leftarrow r$; $i \leftarrow 0$; $j \leftarrow j$. If $i \geq (p-2)$ then $\mathcal{S}$ terminates and returns failure. $\mathcal{S}$ sends $(\varpi_{*,3}, \varpi_{*,4}, n_1, n_2, r_{0,j})$ as a commitment. Then, $\hat{V}$ issues challenge numbers $(ch_{0,j,1}, ch_{0,j,2})$ to $\mathcal{S}$. Next, $\mathcal{S}$ computes $k_3 =$

$-\frac{(ch_{0,j,2}\cdot ID_*\cdot c+r_{0,j})(\tau+n_2\cdot a+ID_*)}{(ch_{0,j,1}\cdot ID_*\cdot c+r_{0,j})(\tau+n_1\cdot a+ID_*)}$. If $k_3\cdot ch_{0,j,1}(\tau+n_1\cdot a+ID_*)+ch_{0,j,2}(\tau+n_2\cdot a+ID_*)=0$ or $(ch_{0,j,2}\cdot ID_*\cdot c+r_{0,j})(\tau+n_2\cdot a+ID_*)=0$ or $(ch_{0,j,1}\cdot ID_*\cdot c+r_{0,j})(\tau+n_1\cdot a+ID_*)=0$ then $\mathcal{S}$ terminates and returns failure. Otherwise, $z_{0,j,1}=g^{\frac{k_3\cdot\alpha}{x\cdot(k_3\cdot ch_{0,j,1}(\tau+n_1\cdot a+ID_*)+ch_{0,j,2}(\tau+n_2\cdot a+ID_*))}}$ and $z_{0,j,2}=g^{\frac{\alpha}{x\cdot(k_3\cdot ch_{0,j,1}(\tau+n_1\cdot a+ID_*)+ch_{0,j,2}(\tau+n_2\cdot a+ID_*))}}$ are the response. $\mathcal{S}$ then keeps $(r_{0,j}, ch_{0,j,1}, ch_{0,j,2}, z_{0,j,1}, z_{0,j,2})$ in the $\mathfrak{TC}$ and returns $(z_{0,j,1}, z_{0,j,2})$ to $\hat{V}$.

- If $\hat{V}$ issues a request for the form $(\mathtt{ch}, j, i)$, where $\mathtt{ch}\overset{\$}{\leftarrow}\mathbb{Z}_p^*$ and j is any value such that $j\leq\nu$, then $\mathcal{S}$ acts as follows. First, $\mathcal{S}$ sets $r_{i,j}\in R_s\leftarrow r$; $i\leftarrow i+1$. If $i\geq(p-2)$ then $\mathcal{S}$ terminates and returns failure. $\mathcal{S}$ sends $(\varpi_{*,3}, \varpi_{*,4}, n_1, n_2, r_{i,j})$ as a commitment $(ch_{i,j,1}, ch_{i,j,2})\leftarrow\mathtt{ch}\overset{\$}{\leftarrow}\mathbb{Z}_p^*$. Then, $\hat{V}$ issues challenge numbers $(ch_{i,j,1}, ch_{i,j,2})$ to $\mathcal{S}$. Next, $\mathcal{S}$ computes $k_3=-\frac{(ch_{i,j,2}\cdot ID_*\cdot c+r_{i,j})(\tau+n_2\cdot a+ID_*)}{(ch_{i,j,1}\cdot ID_*\cdot c+r_{i,j})(\tau+n_1\cdot a+ID_*)}$. If $k_3\cdot ch_{i,j,1}(\tau+n_1\cdot a+ID_*)+ch_{i,j,2}(\tau+n_2\cdot a+ID_*)=0$ or $(ch_{i,j,2}\cdot ID_*\cdot c+r_{i,j})(\tau+n_2\cdot a+ID_*)=0$ or $(ch_{i,j,1}\cdot ID_*\cdot c+r_{i,j})(\tau+n_1\cdot a+ID_*)=0$ then $\mathcal{S}$ terminates and returns failure. Otherwise, $z_{i,j,1}=g^{\frac{k_3\cdot\alpha}{x\cdot(k_3\cdot ch_{i,j,1}(\tau+n_1\cdot a+ID_*)+ch_{i,j,2}(\tau+n_2\cdot a+ID_*))}}$ and $z_{i,j,2}=g^{\frac{\alpha}{x\cdot(k_3\cdot ch_{i,j,1}(\tau+n_1\cdot a+ID_*)+ch_{i,j,2}(\tau+n_2\cdot a+ID_*))}}$ are the response. $\mathcal{S}$ then keeps $(r_{i,j}, ch_{i,j,1}, ch_{i,j,2}, z_{i,j,1}, z_{i,j,2})$ in the $\mathfrak{TC}$ and returns $(z_{i,j,1}, z_{i,j,2})$ to $\hat{V}$.

This continues until $\hat{V}$ outputs the state information $\mathtt{St}$ or makes at most $q_{IP}<(p-1)$ queries and then it stops.

Phase 3: Now, A changes status from $\hat{V}$ to $\hat{P}$. $\hat{P}$ begins the impersonation process.

- $\hat{P}$ first randomly chooses an integer $r\in\mathbb{Z}_p^*$ and then sends $(\varpi_{*,3}, \varpi_{*,4}, \varpi_{*,6}, n_1, n_2, r)$ to V.
- $\mathcal{S}$ first sets $ch_1=ch_2$ and randomly selects an integer $ch_1\in\mathbb{Z}_p^*$ such that $ch_1\neq\frac{-r}{c\cdot ID_*}$ and (r, ch_1, ch_2) are not in the $\mathfrak{TC}$ and then sends it ch_1, ch_2 to $\hat{P}$.
- $\hat{P}$ returns z_1, z_2.

Define $Check(pk_K, ID, z_1, z_2, ch_1, ch_2, \varpi_{*,3}, \varpi_{*,4}, \varpi_{*,6}, n_1, n_2, r, ID_*)$ as: $P_{pub}\overset{?}{=}\hat{e}(z_1,\ \varpi_{*,3}^{n_1\cdot ch_1}\cdot\varpi_{*,4}^{ID_*\cdot ch_1}\cdot A^{n_1\cdot r}\cdot C^{ch_1\cdot ID_*^2}\cdot D^{n_1\cdot ch_1\cdot ID_*}\cdot g^{ID\cdot r}\cdot\varpi_{*,6}^{r}\cdot$

$E^{r\cdot ID_*} \cdot B^{ch_1}) \cdot \hat{e}(z_2, \varpi_{*,3}^{n_2\cdot ch_2} \cdot \varpi_{*,4}^{ID_*\cdot ch_2} \cdot A^{n_2\cdot r} \cdot C^{ch_2\cdot ID_*^2} \cdot D^{n_2\cdot ch_2\cdot ID_*} \cdot g^{ID\cdot r} \cdot \varpi_{*,6}^{r} \cdot E^{r\cdot ID_*} \cdot B^{ch_2})$, where $P_{pub} = \hat{e}(g,g)^{\alpha}$. If the equality holds, then it outputs 1; otherwise, it outputs 0. If *Check* outputs 1 then $\mathcal{S}$ checks if $P_{pub} \neq \hat{e}(z_1^{\tau+n_1\cdot a+ID_*}, (g^x \cdot g^{ID_*\cdot c})^{ch_1} g^r) \cdot \hat{e}(z_2^{\tau+n_2\cdot a+ID_*}, (g^x \cdot g^{ID_*\cdot c})^{ch_2} g^r)$, then $\mathcal{S}$ returns failure. Otherwise, $\mathcal{S}$ computes the solution for the 2-SDH problem as follows: let $m = ID_* \cdot c + \frac{r}{ch_1}$. Then compute $Z = (z_1^{ch_1(\tau+n_1\cdot a+ID_*)} \cdot z_2^{ch_2(\tau+n_2\cdot a+ID_*)})^{\frac{1}{\alpha}} = g^{(\frac{(\alpha-v)\cdot ch_1(\tau+n_1\cdot a+ID_*)}{\alpha(\tau+n_1\cdot a+ID*)(ch_1(x+ID_*\cdot c)+r)})} \cdot g^{(\frac{v\cdot ch_2(\tau+n_2\cdot a+ID_*)}{\alpha(\tau+n_2\cdot a+ID*)(ch_2(x+ID_*\cdot c)+r)})}$. Since $ch_1 = ch_2$ and $g = h^x$, $Z = g^{\frac{1}{x+ID_*\cdot c+r/ch_1}} = g^{\frac{1}{x+m}} = h^{\frac{x}{x+m}} = h^{\frac{-m}{x+m}} \cdot h$. Hence, let $Z^* = (Z^{-1} \cdot h)^{\frac{1}{m}} = h^{\frac{1}{x+m}}$. $\mathcal{S}$ wins the game and returns (Z^*, m) as the solution to the 2-SDH problem. Otherwise, $\mathcal{S}$ returns failure. Note that v is assumed to be a variable that does not need to be known for solving the 2-SDH problem.

7.4.2 Proof of Security

Theorem 7.6 *Let $IBI = (KeyGen, Extract, P, V)$ be an IBI-CRA identity-based identification scheme associated with the 2-SDH assumption. Let $\mathcal{A} = (\hat{U}, \hat{V}, \hat{P})$ be an imp-cra adversary of time complexity $\mathtt{t}(.)$ attacking IBI. Then there exists a 2-SDH adversary $\mathcal{S}$ of time complexity $\mathtt{t}'(.)$ solving the 2-SDH problem such that for every security parameter ℓ*

$$\begin{aligned}
\mathrm{Adv}_{IBI,\mathcal{A}}^{imp\text{-}cra}(\ell) &\geq (1/(1-\frac{1}{p-1})^{q_{ID}})(1/(1-\frac{1}{p-1})^{q_{IP}}) \cdot \frac{(p-1)^2}{p^2-2p} \cdot \mathrm{Adv}_B^{2-SDH}(\ell) \\
\mathrm{Adv}_{IBI,\mathcal{A}}^{imp\text{-}cra}(\ell) &\geq e^2 \cdot \frac{(p-1)^2}{p^2-2p} \cdot \mathrm{Adv}_B^{2-SDH}(\ell) \text{ or, put in simple terms,} \\
\epsilon &\geq e^2 \cdot \frac{(p-1)^2}{p^2-2p} \cdot \epsilon',
\end{aligned}$$

where e is the natural logarithm. Moreover, the time complexity $\mathtt{t}$ of $\mathcal{A}$ is $\mathtt{t} \leq \mathtt{t}' - ((8+4q_{ID}+2q_{IP})C_G + 5C_P)$, where C_G is a computation time of group exponential operation and C_P is a computation time of bilinear group pairing operation, and $\mathcal{A}$ can request queries at most $q_{ID} < p-2$ queries and at most $q_{IP} < p-1$ queries. For simplicity, we assume that the entire computation time of the multiplication in $\mathbb{Z}_p^*$ for all the phases is one unit of time.

Corollary 7.7 *If the 2-SDH assumption is $(2, \mathtt{t}', \epsilon')$-secure then the IBI-CRA identity-based identification scheme associated with the 2-SDH assumption is $(q_{ID}, q_{IP}, \mathtt{t}, \epsilon)$-secure against impersonation under CR1$^+$ attack.*

Proof: The experiment $\mathcal{S}(h, h^x, h^{x^2})$ in Section 7.4.1 shows that the $IBI\text{-}CRA$ identity-based identification scheme that is $(q_{ID}, q_{IP}, \mathtt{t}, \epsilon)$-secure against impersonation under $\mathsf{CR1}^+$ attack can be reduced to the $(2, \mathtt{t}', \epsilon')$-SDH problem. For any input $h \in \mathbb{G}_1$, $h^x \in \mathbb{G}_1$ and $h^{x^2} \in \mathbb{G}_1$, $\mathcal{S}(h, h^x, h^{x^2})$ returns $(h^{\frac{1}{x+r}}, r)$ where these outputs are computed in one unit of time from outputs (z_1, z_2, r). These outputs are obtained if the adversary $\mathcal{A}$ breaks the $IBI\text{-}CRA$ identity-based identification scheme with the $\mathsf{CR1}^+$ attack setting.

However, to show the probability that the experiment $\mathcal{S}$ can succeed in solving the 2-SDH problem with an advantage of at least ϵ', we analyse four events required for $\mathcal{S}$ to succeed.

E1: $\mathcal{S}$ does not abort as a result of the queries in phase 1.

E2: $\mathcal{S}$ does not abort as a result of the queries in phase 2.

E3: $\mathcal{A}$ can impersonate user ID^*.

E4: $\mathcal{S}$ does not abort as a result of $\mathcal{A}$'s output in phase 3.

The probability that $\mathcal{S}$ will succeed, if all of the above events happen, is

$$\begin{aligned}\mathbf{Pr}[E1 \wedge E2 \wedge E3 \wedge E4] &= \mathbf{Pr}[E1] \cdot \mathbf{Pr}[E2|E1] \cdot \mathbf{Pr}[E3|E1 \wedge E2] \\ &\quad \cdot \mathbf{Pr}[E4|E1 \wedge E2 \wedge E3] \\ &= \mathbf{Pr}[E1] \cdot \mathbf{Pr}[E2] \cdot \mathbf{Pr}[E3|E1 \wedge E2] \cdot \mathbf{Pr}[E4].\end{aligned}$$

The following claims give a lower bound for each of the above terms.

Claim 7.8 (1) *The probability that $\mathcal{S}$ does not abort as a result of the queries in phase 1 is at least $1/e$ where e is the natural logarithm. Hence, $\mathbf{Pr}[E1] \geq (1 - 1/(p-1))^{q_{ID}} \approx 1/e$.*

Proof: In general, we assume that $\mathcal{A}$ does not make a query request for a private key for the same identity twice. Subsequently, the only result of a user's private key queries that causes a termination is a collision of secret keys in the $\mathfrak{ID}$, which $\mathcal{A}$ can use to compute a KGC's private key. Therefore, using the inductive hypothesis to prove the above claim, the probability that, for each query, $\mathcal{S}$ does not abort as the result of a collision of a user's private key queries on the $\mathfrak{ID}$ is $1 - 1/(p-1)$. Consequently, the probability that $\mathcal{S}$ does not abort after issuing responses for

a user's private key queries at most q_{ID} queries is at least $\prod_{i=1}^{q_{ID}}(1-1/(p-1))$. Therefore, since q_{ID} approached $p-2$, the probability that $\mathcal{S}$ does not abort as a result of all the private key queries is at least $(1-(1/(p-1)))^{p-2} \geq 1/e$. □

Claim 7.9 (2) *The probability that $\mathcal{S}$ does not abort as a result of the queries in phase 2, after finishing the queries in phase 1, is at least $1/e$. Hence, $\mathbf{Pr}[E2|E1] = \mathbf{Pr}[E2] \geq (1-1/(p-1))^{q_{IP}} \approx 1/e$.*

Proof: First, in phase 1, $\mathcal{S}$ requires only one user's identity from the $\mathfrak{ID}$ for a challenge, where the limit of q_{ID} can be set at most $q_{ID} < (p-2)$. Therefore, the results of the private key queries in phase 1 do not interfere with the identification queries in phase 2, so that $\mathbf{Pr}[E2|E1] = \mathbf{Pr}[E2]$.

Second, we assume that $\mathcal{A}$ does not issue a query for the same pair of challenge numbers twice. We then again use the inductive hypothesis to prove this claim. The probability that, for each query, $\mathcal{S}$ does not abort as the result of a collision of identification queries on the $\mathfrak{TC}$ is $1-1/(p-1)$. Consequently, the probability that $\mathcal{S}$ does not abort after issuing responses for all the requests for identification at most q_{IP} queries is at least $\prod_{i=1}^{q_{IP}}(1-1/(p-1))$. Therefore, since q_{IP} approached $p-2$, the probability that $\mathcal{S}$ does not abort as a result of all the identification queries is at least $(1-1/(p-1))^{p-2} \geq 1/e$.

Finally, the probability that $\mathcal{S}$ does not abort as a result of all the conversation transcript queries in phase 2, after phase 1 has finished, is $\mathbf{Pr}[E2] \geq (1-1/(p-1))^{q_{IP}} \approx 1/e$ for at most $q_{IP} < (p-2)$ queries. □

Claim 7.10 (3) *If algorithm $\mathcal{A}$ can impersonate a user ID^* after phases 1 and 2 have finished, then algorithm $\mathcal{A}$'s view is identical to its view in a real attack. Hence, $\mathbf{Pr}[E3|E1 \wedge E2] \geq \epsilon$.*

Proof: The public key given to $\mathcal{A}$ is from the same distribution as a public key produced by the *KeyGen* algorithm. Responses to identity queries and $\mathsf{CR1}^+$ attacks are the same as in a real attack. Nonetheless, each response for identity queries and conversation transcripts is uniformly and independently distributed in $\mathbb{Z}_p^*$ and all responses to queries are valid. Hence, $\mathcal{A}$ will impersonate an honest user with a probability of at least ϵ. Hence, $\mathbf{Pr}[E3|E1 \wedge E2] \geq \epsilon$. □

Claim 7.11 (4) *The probability that $\mathcal{S}$ does not abort during phase 3, after phases 1 and 2 have finished and $\mathcal{A}$ can identify itself as a user ID_* is at least $1-1/(p-1)^2$. Hence, $\mathbf{Pr}[E4|E1 \wedge E2 \wedge E3] = \mathbf{Pr}[E4|E3] \geq 1-1/(p-1)^2$*

Proof:

In phase 3, $\mathcal{S}$ aborts the simulation only if $\mathcal{A}$ can impersonate the user ID_* by generating two random generators $z_1^*, z_2^* \stackrel{\$}{\leftarrow} \mathbb{G}_1$, which satisfy the equation below.

$$\begin{aligned} P_{pub} &= \hat{e}(g,g)^{\alpha} \\ &= \hat{e}(z_1^*, \varpi_{*,3}^{n_1 \cdot ch_1} \cdot \varpi_{*,4}^{ID_* \cdot ch_1} \cdot A^{n_1 \cdot r} \cdot C^{ch_1 \cdot ID_*^2} \cdot D^{n_1 \cdot ch_1 \cdot ID_*} \cdot g^{ID \cdot r}) \cdot \\ &\quad \hat{e}(z_2^*, \varpi_{*,3}^{n_2 \cdot ch_2} \cdot \varpi_{*,4}^{ID_* \cdot ch_2} \cdot A^{n_2 \cdot r} \cdot C^{ch_2 \cdot ID_*^2} \cdot D^{n_2 \cdot ch_2 \cdot ID_*} \cdot g^{ID \cdot r}) \end{aligned}$$

Therefore, the probability that $\mathcal{A}$ can succeed in choosing two random generators to fulfil the equation above is $1/(p-1)^2$. It is also noted that termination in this phase is not influenced by the private key and identification queries, since a was not needed to generate the above z_1^*, z_2^*. Consequently, the probability that $\mathcal{S}$ does not abort as a result in phase 3, after phases 1 and 2 have finished and $\mathcal{A}$ can identify itself as a user ID_*, is at least $\mathbf{Pr}[E4|E3] \geq 1 - 1/(p-1)^2$. □

Using the bounds from all of the above claims, it is shown that $\mathcal{S}$ can generate a solution for the 2-SDH problem with a probability of at least $\epsilon \cdot \frac{1}{e^2} \cdot (1 - 1/(p-1)^2)$ as required. Moreover, $\mathcal{S}$'s running time is the same as $\mathcal{A}$'s running time, plus the time that it takes to respond to q_{ID} queries for the request of a private key associated with a public key (pk_K, ID) and to q_{IP} queries for the request of ID_*'s identification transcripts. Let C_G be a computation time of group exponential operation and C_P be a computation time of bilinear group pairing operation. Note that the computation of multiplication in the finite field order p is not taken into account for simplicity. The total running time is consequently at most $\mathtt{t}' \geq \mathtt{t} + (8 + 4q_{ID} + 2q_{IP})C_G + 5C_P$ as required. This completes the proof of Theorem 7.6. □

7.5 Efficiency

The performance comparison between our identity-based identification scheme and the state-of-the-art identity-based identification scheme secure against passive attack and concurrent-active attack proposed by Kurosawa and Heng [KH05] is provided in Table 7.1. Let C_E be a computation of group exponential operation, C_P be a computation of bilinear group pairing operation and C_M be a computation time of bilinear group multiplicative operation. Note that $KH\text{-}IBI\text{-}PA$ and $KH\text{-}IBI\text{-}CA$ are Kurosawa-Heng IBI secured against the passive and concurrent attacks, respectively.

Computation	$KH\text{-}IBI\text{-}PA$	$IBI\text{-}PA$
Prover	$3C_E+C_P+4C_M$	C_E
Verifier	$3C_E+C_P+4C_M$	$9C_E+C_P+8C_M$
Total	$6C_E+2C_P+8C_M$	$10C_E+1C_P+8C_M$
Bandwidth	$KH\text{-}IBI\text{-}PA$	$IBI\text{-}PA$
Public(bits)	3403	2210
Secret(bits)	331	2721
Communi-cation (bits)	1515	2019
Computation	$KH\text{-}IBI\text{-}CA$	$IBI\text{-}CRA$
Prover	$6C_E+2C_P+6C_M$	$2C_E$
Verifier	$6C_E+2C_P+6C_M$	$18C_E+2C_P+16C_M$
Total	$12C_E+4C_P+12C_M$	$20C_E+2C_P+16C_M$
Bandwidth	$KH\text{-}IBI\text{-}CA$	$IBI\text{-}CRA$
Public(bits)	5451	2210
Secret(bits)	662	3052
Communi-cation (bits)	3190	2337

Table 7.1: Table: Bandwidth and Computation Comparison.

7.6 Definition of Escrowed Deniable Identification Schemes

In this section, we provide a formal model of an escrowed deniable identification scheme and its security model. Unlike previous definitions of identification schemes [BP02, Oka92, Sch89, Sho99, AABN02, BFGM01], we introduce a trusted third party into the escrowed deniable identification schemes, a party that has the power to invoke the deniability of a prover. Therefore, in our security models of escrowed deniable identification schemes, we consider a new property, '*transferability*'. This provides a security guarantee for the trusted third party and the prover, which preserves privacy for all other cases except the case in dispute.

7.6.1 Outline of Escrowed Deniable Identification Schemes

We introduce a notion called an *escrowed deniable identification scheme* (EDID) that balances both the need for deniability and the need for undeniability in identification schemes. In an escrowed deniable identification scheme, in addition to

the fact that a prover can deny an identification transcript, a trusted authority can convert a deniable identification transcript into an undeniable one, enabling anyone to verify ownership of the transcript.

Formally, an EDID scheme involves a prover P, a verifier V, a trusted authority TA and any third party $AnyV$. It consists of the following algorithms and protocols:

System Parameter Generation ($Setup$):

On input of 1^ℓ, where ℓ is the security parameter, the algorithm generates a system parameter, i.e. $Setup(1^\ell) \rightarrow \mathsf{param}$.

Trusted Authority Key Generator ($TKeyGen$):

On input of param, $TKeyGen$ generates a public-private key pair (pk_T, sk_T) for the trusted authority, i.e. $TKeyGen(\mathsf{param}) \rightarrow (pk_T, sk_T)$.

Prover Key Generator ($PKeyGen$):

On input of param, $PKeyGen$ generates a public-private key pair (pk_P, sk_P) for the prover, i.e. $PKeyGen(\mathsf{param}) \rightarrow (pk_P, sk_P)$.

Identification protocol ($\langle P, V \rangle$):

This is an interactive protocol between the prover P and the verifier V. It consists of four rounds of communication and six PPT algorithms, ($\mathtt{Cmt}_V, \mathtt{Cmt}_P, \mathtt{Ch}$, $\mathtt{Rsp}, \mathtt{Check}_P, \mathtt{Check}_V$), where ($\mathtt{Cmt}_P$, $\mathtt{Check}_P$) and ($\mathtt{Check}_V$, $\mathtt{Cmt}_V$) are sets of algorithms to generate commitments and to verify the commitment run by the prover P and the verifier V, respectively. $\mathtt{Ch}$ is an algorithm to disclose the challenge and $\mathtt{Rsp}$ is an algorithm run by the prover P to generate a response after the process of commitment generation, challenge disclosure and commitment verification.

- Step 1. V chooses a challenge c at random from a certain domain, and computes $T \leftarrow \mathtt{Cmt}_V(c)$. V then sends T to P.
- Step 2. P chooses r at random from a certain domain, and computes $a \leftarrow \mathtt{Cmt}_P(r)$. P then sends a to V.
- Step 3. V runs $\mathtt{Ch}$ to reveal a random challenge c, and sends it to P.
- Step 4. After receiving V's challenge c, P then runs $b \leftarrow ck_P(c, T)$. If $b = 0$, P aborts; otherwise, it computes its response by running $z \leftarrow \mathtt{Rsp}(sk_P, r, c)$, and sends z to V.

- Step 5. V checks the validity of P's response by running $\texttt{Check}_V(pk_T, pk_P, a, c, z)$. If the output is '1', V accepts it; otherwise, it rejects it.

Open protocol ($\langle TA, V\rangle$):

An open protocol can be formalised by two (probabilistic) polynomial-time algorithms $\texttt{Open}, \texttt{Verf}$, where $\texttt{Open}$ is invoked by TA, and $\texttt{Verf}$ is executed by the verifier V. On input of a transcript tr and the private key of TA, $\texttt{Open}$ outputs an evidence to affirm the authenticity of tr. $\texttt{Verf}$ is an algorithm for validating the evidence with respect to tr and pk_P. It takes as input pk_T, pk_P, tr and the evidence, and outputs 1 to accept or 0 to reject the evidence.

Transfer protocol ($\langle V, AnyV\rangle$):

A transfer protocol is an interactive protocol between any third party $AnyV$ and the verifier V that possesses a transcript tr and its affirmative evidence from the trusted authority (TA). The aim of the protocol is to convince $AnyV$ that tr indeed represents an execution of the identification protocol between P and V.

The completeness can be defined in a natural way. Next we define other security properties in an escrowed deniable identification scheme.

7.6.2 Deniability

Roughly speaking, deniability indicates that given a transcript of an execution of an identification protocol, a prover is able to deny that he is the prover in the execution. To achieve deniability, we require that the verifier can generate this transcript itself. Formally, we consider the following definition, which share similarity with that of zero knowledge.

Definition 7.1 (Deniability) *An EDID scheme is* deniable *if for any* $\mathsf{param} \leftarrow Setup(1^\ell)$, $(pk_T, sk_T) \leftarrow TKeyGen(\mathsf{param})$ *and* $(pk_P, sk_P) \leftarrow PKeyGen(\mathsf{param})$, *for any PPT algorithm* $\mathcal{D}$, *or for any verifier strategy* V^*, *there exists a PPT algorithm* S, *which has oracle access to* V^*, *such that*

$$|\Pr[\text{Expt}_1(\ell) = 1] - \Pr[\text{Expt}_2(\ell) = 1]| = \epsilon(\ell),$$

where $\epsilon(\cdot)$ *is a negligible function in* ℓ, *and* $Expt_1(\ell)$ *and* $Expt_2(\ell)$ *are defined as follows.*

$\text{Expt}_1(\ell)$:	$\text{Expt}_2(\ell)$:
$tr \leftarrow \langle P(sk_P), V^* \rangle (pk_T, pk_P)$	$tr' \leftarrow S^{V^*}(pk_T, pk_P)$
$b \leftarrow \mathcal{D}(pk_T, pk_P, tr)$	$b' \leftarrow \mathcal{D}(pk_T, pk_P, tr)$
return b	*return* b

where the probabilities are taken over the random bits used in Setup, TKeyGen, PKeyGen, and the random bits consumed by P, V^*, S and $\mathcal{D}$.

7.6.3 Impersonation

An identification scheme is secure against impersonation if no-one except the prover P with its public key pk_P can identify itself to others as P. In this section, we consider the most common impersonation attacks, i.e. passive attacks and active attacks, which are described below:

- **Passive Attack** (imp-pa): This is the weakest form of attack considered for impersonation. An adversary can only listen to the interaction between a prover and a verifier, and then begin to impersonate the prover after the interaction.

- **Active Attack** (imp-aa): This attack is stronger than the one above. In an active attack, the adversary, acting as a (cheating) verifier, actively interacts with prover clones in sequence. After the last execution of the identification protocol is over, it starts to impersonate the prover to others.

Impersonation under Active Attack: An imp-aa adversary $\mathcal{A}$ is a pair of PPT algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ acts as V^* and $\mathcal{A}_2$ acts as P^*. Let `St` denote the state of information. The active attack is initialised by first calling *Setup*, *TKeyGen* and *PKeyGen* to generate public-private key pairs (pk_T, sk_T) and (pk_P, sk_P) for the trusted authority and the prover respectively. Taking public keys pk_T and pk_P as input, the adversary $\mathcal{A}$ then performs its attack in the following two phases:

- Phase 1. (Learning Phase) Given input pk_T, pk_P, the adversary $\mathcal{A}_1$ is allowed to interact with P's clones sequentially. When each of P's clones interacts with $\mathcal{A}_1$, it is initialised with (pk_P, sk_P), pk_T and fresh random coins. Later, $\mathcal{A}_1$ outputs `St` to be passed onto $\mathcal{A}_2$. This completes phase 1.

- Phase 2. (Impersonation Phase) At the beginning of phase 2, V is initialised with the public keys pk_T, pk_P, while the adversary $\mathcal{A}_2$ is given `St`. Then $\mathcal{A}_2$

tries to impersonate P to V. At the end of this phase, V outputs a decision bit b, indicating *Accept* or *Reject*.

The adversary $\mathcal{A}$ is said to be *successful* in the attack if V outputs 1 at the end of Phase 2. Formally, we consider the following experiment:

$\text{Expt}_{\mathcal{A}}^{\mathsf{imp-aa}}(\ell)$:
  $\mathsf{param} \leftarrow Setup(1^{\ell})$
  $(pk_T, sk_T) \leftarrow TKeyGen(\mathsf{param})$
  $(pk_P, sk_P) \leftarrow PKeyGen(\mathsf{param})$
  $(\perp, \mathtt{St}) \leftarrow {\mathcal{A}_1}^{P(sk_P)}(pk_T, sk_T, pk_P)$
  $(\perp, b) \leftarrow \langle \mathcal{A}_2(sk_T, \mathtt{St}), V \rangle (pk_T, pk_P)$
  return b

where an oracle call to $P(sk_P)$ results in an execution of the identification protocol with the prover P and a transcript tr is returned.

Definition 7.2 (Security against Impersonation under Active Attack) *We say an escrowed deniable identification scheme EDID is* secure against impersonation under active attack*, if there is no PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that the probability* $\Pr[\text{Expt}_{\mathcal{A}}^{\mathsf{imp-aa}}(\ell) = 1]$ *is non-negligible in ℓ.*

Note that in the definition above the adversary can be the trusted authority. That is, even TA cannot impersonate the prover in an active attack.

7.6.4 Transferability

Intuitively, the notion of *transferability* in escrowed deniability identification schemes is aimed at revealing the transcript confirmation or evidence that proves the validity of the prover of the transcript. The idea is that a verifier is provided with evidence for a case in dispute to prove to another party who would like to be convinced of the validity of the transcript. To complete this idea, a trusted authority is involved to process an opening (transferring) algorithm. Unlike the deniability property in a general identification scheme (for example, zero-knowledge protocol based identification schemes), the verifier can now convince another party that the transcript of an identification scheme is actually due to an interaction with the claimed prover with the help of or evidence from a trusted party. In the experiment below, the adversary is modelled as a malicious verifier who tries to convince any third party to

accept the transcript without the help of the trusted authority. Hence, the trusted authority is viewed as an opening oracle $\mathcal{OPN}$ who answers queries for opening the chosen transcript. We provide a formal definition of *transferability* as follows.

Let V^* be any verifier strategy (honest or malicious). Let (pk_P, sk_P) be the prover's public key and private key generated by the key generation algorithm of the identification scheme, and let (pk_T, sk_T) be the TA's public key and private key, respectively, generated by the key generation algorithm of the identification scheme. Let $tr \leftarrow \langle P(sk_P), V^* \rangle (pk_P)$ be the transcript of an interaction between P and V^*, and let $\theta \leftarrow \langle TA(sk_{TA}), V^* \rangle (pk_P)$ be the confirmation evidence θ of an interaction between TA and V^*. Let $\mathtt{Verf}$ be the verifier's decision algorithm, which takes a transcript tr and its confirmation evidence θ as input and outputs 1 or 0, which indicate '*Accept*' or '*Reject*', respectively. Let S be a probabilistic polynomial-time algorithm. We consider the following experiment:

$\mathrm{Expt}^{\mathsf{tran}}_{\mathcal{A}}(\ell)$:
$(pk_P, sk_P) \leftarrow PKeyGen(1^\ell)$
$(pk_T, sk_T) \leftarrow TKeyGen(1^\ell)$
$(\perp, \mathtt{St}) \leftarrow \mathcal{A}_1{}^{\mathcal{OID}, \mathcal{OPN}}(pk_P, pk_T)$
$(tr^*, \theta^*) \leftarrow \mathcal{A}_2(\mathtt{St})$
If (tr^*, θ^*) has been queried to $\mathcal{OID}, \mathcal{OPN}$ then $\perp$,
otherwise, in the transfer protocol (or any other protocol for transferring the proof (tr^*, θ^*)),
$(\perp, b) \leftarrow \langle \mathcal{A}_2(tr^*, \theta^*), AnyV \rangle (pk_P, pk_T)$
Return b

Adversary $\mathcal{A}$ is said to be *successful* in the attack if $AnyV$ outputs $b = Accept$.

Definition 7.3 (Security against Transferability Attack) *An identification scheme* $\mathsf{ID} = (PKeyGen,\ P,\ V)$ *is said to be secure against transferability attacks if there is no probabilistic polynomial-time* tran *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *such that the probability* $\Pr[\mathrm{Expt}^{\mathsf{tran}}_{\mathcal{A}}(\ell) = 1]$ *is non-negligible in* ℓ.

Oracle $\mathcal{OPN}(tr)$:
$\theta \leftarrow \mathtt{Open}(sk_T, pk_T, tr)$
$b \leftarrow \mathtt{Verf}(tr, \theta, pk_T)$
Return θ iff $b =$ accept
Otherwise, return $\perp$

Oracle $\mathcal{HO}(str)$:
$m \leftarrow H(str)$
Return m

Oracle $\mathcal{OID}$:
$tr = (a, b, z) \leftarrow \langle \mathcal{OID}(sk_P), V \rangle (pk_T, pk_P)$
Return tr

Figure 7.1: Oracle for Adversary Attacking Transferability of Escrowed Deniability Identification Scheme

7.7 Our Construction

7.7.1 High Level Idea

Before presenting our construction of escrowed deniable identification schemes, we will first describe the idea and intuition behind our construction. Let TA's pair of public/secret keys be (pk_T, sk_T). First, P generates a commitment $\mathtt{Cmt}$, and V replies with a random challenge c. Then, P signs both $\mathtt{Cmt}$ and c to obtain θ. Next, P will verifiably encrypt θ using the TA's public key pk_T. That is, $\phi \leftarrow VE_{pk_T}(\theta)$. Then, P sends ϕ to V. V can check the validity of ϕ with respect to pk_T and pk_Pr, but V cannot transfer this conviction to anyone else (due to the *indistinguishability* property of the verifiable encryption used). When mischievous behaviour occurs, TA converts the transcript and makes it undeniable. TA can decrypt ϕ using sk_T to obtain θ, and since θ is a regular digital signature generated by P; hence, it is undeniable.

7.7.2 The Construction

In this section, we present our scheme based on the idea outlined above. The construction uses a Boneh-Boyen short signature scheme and verifiable encryption scheme developed by Boneh et al. [BB04, BGLS03]. We incorporate the technique described in [HWLZ08] to construct our EDID scheme in the standard model. The scheme works as follows.

System Parameter Generation (*Setup*):

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be two multiplicative cyclic groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$ for some large prime p. g, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is a bilinear pairing. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. The system parameter param then consists of $(\mathbb{G}_1, \mathbb{G}_T, \hat{e}, p, g, g_1, g_2, H)$.

Trusted Authority Key Generator ($TKeyGen$):

Given the public parameter param, $TKeyGen$ selects random numbers $x, y \in \mathbb{Z}_p; \mathbb{W} \in \mathbb{G}_1$ and computes $\mathbb{V} = \mathbb{W}^y$, $\mathbb{U} = \mathbb{V}^x$. The public key and private key of the trusted authority are $pk_T = (\mathbb{U}, \mathbb{V}, \mathbb{W})$ and $sk_T = (x, y)$ respectively.

Prover Key Generator ($PKeyGen$):

Given the public parameter param, $PKeyGen$ selects a random number $s \in \mathbb{Z}_p$ and computes $S_P = g_1^s$. The public key and private key of the prover are $pk_P = S_P$ and $sk_P = s$ respectively.

Identification protocol ($\langle P, V \rangle$):

The protocol comprises two parts. The first part is a 4-round zero-knowledge proof protocol of the Schnorr Identification, in which the prover P proves to the verifier V that it knows the private key s, which is the discrete logarithm of the public key S_P to base g_1. In the second part of the identification protocol, prover P generates a $BB04$ short signature θ on the 4-round Schnorr Identification transcript it just carried out with the verifier. P then computes ϕ, which is the verifiable encryption of θ under the TA's public key, and sends it to the verifier. Finally, P proves, in an interactive manner, to the verifier that ϕ is correctly formed. Following the description above, the protocol will be more than four rounds. Optimisation of the round efficiency of the protocol can be done by setting θ to be the signature on the first two rounds of the 4-round Schnorr Identification protocol and conducting the proof-of-correctness of ϕ in parallel with the Schnorr Identification with the verifier. The resulting protocol remains four rounds and it is shown as follows.

1. 1^{st} Round (V to P). (Commitment of Challenge.) V randomly generates $c, d \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\mathbb{T} = g_1^c g_2^d$ and sends $\mathbb{T}$ to P.
2. 2^{nd} Round (P to V).

(a) P randomly generates $r_s \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\mathtt{T} = g_1^{r_s}$. Now, P runs the $KeyGen$ of the $BB04$ signature for the one-time public key pk_{OT} and the one-time private key sk_{OT}. However, P can simply use a common parameter from param, such as $p, \hat{e}$, for the $BB04$ signature. Hence, on input of param, P randomly selects $g_a, g_b \in \mathbb{G}_1$; $\alpha, \eta \in \mathbb{Z}_p$ and computes the one-time public key $pk_{OT} = (g_a, g_b, \mathcal{U} = g_b^{\alpha}, \mathcal{V} = g_b^{\eta}, \mathcal{Z} = \hat{e}(g_a, g_b))$ and the one-time private key $sk_{OT} = (\alpha, \eta)$.

(b) P randomly selects $a, b \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $E_1 = \mathbb{U}^a$ and $E_2 = \mathbb{V}^b$. Then, P randomly generates $r_a, r_b \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $A_1 = \mathbb{U}^{r_a}$ and $A_2 = \mathbb{V}^{r_b}$.

(c) Let $m = H(pk_{OT})$. Now, P computes a signature $\theta = g^{\frac{1}{s+m}}$. Then, P computes $E_3 = \theta \mathbb{W}^{a+b}$ and $A_3 = \hat{e}(\mathbb{W}, S_P g_1^m)^{r_a + r_b}$. Parse $\hat{A}$ as (A_1, A_2, A_3) and $\hat{E}$ as (E_1, E_2, E_3).

(d) Let $\bar{m} = H(\mathtt{T}, \mathbb{T}, pk_P, E_1, E_2, E_3, A_1, A_2, A_3)$. On input of pk_{OT}, sk_{OT}, P randomly chooses $\kappa \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\vartheta = g_a^{\frac{1}{\alpha + \kappa \cdot \eta + \bar{m}}}$. Then P sends $(\mathtt{T}, pk_{OT}, \vartheta, \kappa, \hat{E}, \hat{A})$ to V.

3. 3^{rd} Round (V to P). (Challenge.) V sends c, d to P.

4. 4^{th} Round (P to V). (Response.) P checks if $T \stackrel{?}{=} g_1^c g_2^d$. Output $\perp$ means the check fails. Otherwise, compute $z_s = r_s - cs$, $z_a = r_a - ca$ and $z_b = r_b - cb$. Set $\hat{Z}$ as (z_s, z_a, z_b) and send $\hat{Z}$ to V.

5. (Verification.) V computes $\bar{m} = H(\mathtt{T}, \mathbb{T}, pk_P, E_1, E_2, E_3, A_1, A_2, A_3)$ and $m = H(pk_{OT})$ and outputs *Accept* if the following holds:

$$T_1 \stackrel{?}{=} S_P^c g_1^{z_s},\ \hat{e}(g_a, g_b) \stackrel{?}{=} \hat{e}(\theta, \mathcal{U} \cdot \mathcal{V}^{\kappa} \cdot g_2^{\bar{m}}),\quad A_1 \stackrel{?}{=} E_1^c \mathbb{U}^{z_a},$$

$$A_2 \stackrel{?}{=} E_2^c \mathbb{V}^{z_b},\ A_3 \stackrel{?}{=} \left(\frac{\hat{e}(E_3, S_P g_1^m)}{\hat{e}(g, g_1)} \right)^c \hat{e}(\mathbb{W}, S_P g_1^m)^{z_a + z_b}.$$

Otherwise, it outputs *Reject*.

Open protocol ($\langle TA, V \rangle$):

A protocol can be denoted by $OP = (\ \mathtt{Open}, \mathtt{Verf})$, where `Open` and `Verf` are PPT algorithms used in the protocol detailed in Figure 7.2.

Transfer protocol ($\langle V, AnyV \rangle$):

A protocol can be denoted by $TP = (\mathtt{Cmt}, \mathtt{Ch}, \mathtt{Rsp}, \mathtt{Check})$, where `Cmt`, `Ch`, `Rsp` and `Check` are PPT algorithms used in the following protocol, where the

verifier V proves that a transcript denoted as tr is indeed generated by P to any third party verifier. This protocol is illustrated in Figure 7.2.

7.8 Security Analysis

In this section, we provide a proof of the security of our proposed EDID scheme, which includes deniability, security against impersonation and transferability (escrowed deniability). First, we define the following notations, which we will use throughout the rest of this section.

7.8.1 Deniability

We will now present proof that the identification protocol and the transfer protocol in our EDID scheme are zero knowledge protocols. First, the completeness of the identification and transfer protocols in our EDID scheme are straightforward; hence, this will be omitted. Second, zero knowledge proof of the identification and transfer protocols in our EDID scheme are outlined in the theorems below.

Theorem 7.12 *The identification protocol in our identification scheme EDID is deniable.*

Proof: Let $\mathcal{S}$ be a simulator and V^* be any verifier. Given the public keys $pk_T = (\mathbb{U}, \mathbb{V}, \mathbb{W})$, and $pk_P = S_P = g_1^s$, algorithm $\mathcal{S}$ simulates transcripts as follows.

1. First, $\mathcal{S}$ receives T from V^*, and then computes its response as follows.
   - $\mathcal{S}$ first selects random generators $\mathtt{T}', A_1', A_2', A_3', \theta' \xleftarrow{\$} \mathbb{G}_1$.
   - Then, $\mathcal{S}$ runs $KeyGen$ of the $BB04$ signature scheme and obtains $pk_{OT}' = (g_{a'}, g_{b'}, \mathcal{U} = g_{b'}^{\alpha'}, \mathcal{V} = g_{b'}^{\eta'}, \mathcal{Z}' = \hat{e}(g_{a'}, g_{b'}))$ and $sk_{OT}' = (\alpha', \eta')$.
   - Next, $\mathcal{S}$ chooses integers $a', b', \kappa' \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $E_1' = \mathbb{U}^{a'}$, $E_2' = \mathbb{U}^{b'}$. $\mathcal{S}$ computes $m' = H(pk_{OT}')$.
   - $\mathcal{S}$ computes $E_3' = \theta' \mathbb{W}^{a+b}$ and $\bar{m}' = H(\mathtt{T}', \mathbb{T}, pk_P, E_1', E_2', E_3', A_1', A_2', A_3')$. Then it computes $\vartheta' = g_{a'}^{\frac{1}{\alpha' + \kappa' \cdot \eta' + \bar{m}'}}$.
   - $\mathcal{S}$ responds to $\mathcal{A}$ with $(\mathtt{T}', pk_{OT}', \vartheta', \kappa', E_1', E_2', E_3', A_1', A_2', A_3')$.

Open

TA		**V**
	$\xleftarrow{tr}$	$tr \stackrel{\texttt{def}}{=} (\mathbb{T}, \mathtt{T}, pk_{OT}, \vartheta, \kappa, \hat{A}, \hat{E}, c, d, \hat{Z})$
$\bar{m} \leftarrow H(\mathbb{T}, \mathtt{T}, pk_P, E_1,$ $E_2, E_3, A_1, A_2, A_3)$; $m \leftarrow H(pk_{OT})$; iff not $(T_1 \stackrel{?}{=} S_P^c g_1^{z_s} \wedge$ $\hat{e}(g_a, g_b) \stackrel{?}{=} \hat{e}(\theta, \mathcal{U} \cdot \mathcal{V}^\kappa \cdot g_2^{\bar{m}}) \wedge$ $A_1 \stackrel{?}{=} E_1^c \mathbb{U}^{z_a} \wedge A_2 \stackrel{?}{=} E_2^c \mathbb{V}^{z_b}$ $\wedge A_3 \stackrel{?}{=} \left(\frac{\hat{e}(E_3, S_P g_1^m)}{\hat{e}(g, g_1)}\right)^c$ $\hat{e}(\mathbb{W}, S_P g_1^m)^{z_a + z_b})$ then $\perp$. Otherwise, $\theta \leftarrow E_3/(E_1^{1/xy} E_2^{1/x})$; iff $\hat{e}(g, g_1) \stackrel{?}{=} \hat{e}(\theta, S_P g_1^m)$ then $\texttt{Open}(sk_{TA}) \stackrel{\texttt{def}}{=} \theta$, Otherwise, $\perp$	$\xrightarrow{\theta}$	
		$\texttt{Verf}$: iff $\hat{e}(g, g_1) \stackrel{?}{=} \hat{e}(\theta, S_P g_1^m)$ then $\texttt{Verf} = \texttt{accept}$. Otherwise, $\texttt{Verf} = \texttt{reject}$.

Transfer

V		**AnyV**
	$\xleftarrow{\mathfrak{T}}$	$c', d' \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$; $\mathfrak{T} = g_1^{c'} g_2^{d'}$
$a', r_{a'} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$; $D_1 \leftarrow \theta g_2^{a'}$; $D_2 \leftarrow \hat{e}(g_2, S_P g_1^m)^{r_{a'}}$; $\texttt{Cmt}(pk_P, tr, r_{a'}, a') \stackrel{\texttt{def}}{=} (D_1, D_2)$	$\xrightarrow{tr, D_1, D_2}$	
	$\xleftarrow{c', d'}$	$\texttt{Ch} \stackrel{\texttt{def}}{=} (c', d')$
iff $(\mathfrak{T} \stackrel{?}{=} g_1^{c'} g_2^{d'})$ then $z \leftarrow r_{a'} + c'a'$; $\texttt{Rsp}(pk_P, sk_P, \texttt{Cmt}, \texttt{Ch}) \stackrel{\texttt{def}}{=} z$, otherwise, $\perp$	$\xrightarrow{z}$	
		$\texttt{Check}$: $\bar{m} \leftarrow H(\mathbb{T}, \mathtt{T}, pk_P, E_1, E_2,$ $E_3, A_1, A_2, A_3)$ and $m \leftarrow H(pk_{OT})$. iff $(\hat{e}(D_1, S_P g_1^m)/\hat{e}(g, g_1))^{c'} \stackrel{?}{=}$ $D_2^{-1} \hat{e}(g_2, S_P g_1^m)^z$ $\wedge \hat{e}(g_a, g_b) \stackrel{?}{=} \hat{e}(\theta, \mathcal{U} \cdot \mathcal{V}^\kappa \cdot g_2^{\bar{m}})$, then the transcript tr was generated by P

Figure 7.2: Open & Transfer Protocols

2. After V^* replies with c' and d', $\mathcal{S}$ checks the validity of (c', d') with respect to $\mathbb{T}$ and then rewinds V^* to its previous state and computes a new response as follows.

   - Run $KeyGen$ of the $BB04$ signature scheme and obtain $pk_{OT} = (g_a, g_b, \mathcal{U} = g_b^{\alpha}, \mathcal{V} = g_b^{\eta}, \mathcal{Z} = \hat{e}(g_a, g_b))$ and $sk_{OT} = (\alpha, \eta)$.
   - Select $\kappa, z_s, a, b, z_a, z_b \xleftarrow{\$} \mathbb{Z}_p^*$; $E_3 \xleftarrow{\$} \mathbb{G}_1$; $E_1 = \mathbb{U}^a$; $E_2 = \mathbb{V}^b$; $T_1 = S_P^c g_1^{z_s}$; $A_1 = E_1^{c'}\mathbb{U}^{z_a}$; $A_2 = E_2^c \mathbb{V}^{z_b}$.
   - Then, it computes $m = H(pk_{OT})$, $A_3 = \left(\frac{\hat{e}(E_3, S_P g_1^m)}{\hat{e}(g, g_1)}\right)^c \hat{e}(\mathbb{W}, S_P g_1^m)^{z_a + z_b}$.
   - Compute $\bar{m} = H(\mathtt{T}, \mathbb{T}, pk_P, E_1, E_2, E_3, A_1, A_2, A_3)$, $\vartheta = g_a^{\frac{1}{\alpha + \kappa \cdot \eta + \bar{m}}}$.
   - Return $(\mathtt{T}, pk_{OT}, \vartheta, \kappa, \hat{E}, \hat{A})$ to V^*.

3. Finally, upon receiving c and d from V^*, $\mathcal{S}$ aborts if $c \neq c'$ or $d \neq d'$. Otherwise, $\mathcal{S}$ replies with (z_s, z_a, z_b).

From the structure of the protocol and the simulation process, the difference between the distribution of the real transcripts $\Im = \{tr\}$ and the simulated transcripts $\hat{\Im} = \{\widehat{tr}\}$ lies only in the event that $c \neq c'$ or $d \neq d'$. We denote this event by INEQ. We can see the probability that event INEQ happens only with negligible probability. If this is not the case, we can compute the discrete logarithm of g_2 with respect to the base g_1. If $c \neq c'$, Then $d \neq d'$ as well. Since $g_1^c g_2^d = T = g_1^{c'} g_2^{d'}$, we obtain $g_2 = g_1^{(c-c')/(d'-d)}$. As the group order p is known, we can obtain $\log_{g_1} g_2 = (c - c')/(d' - d) \bmod p$. This contradicts the discrete logarithm assumption, hence, we can run this experiment to solve the discrete logarithm problem by setting $g_1 = h$ and $g_2 = h^a$, where h, h^a are instances of the discrete logarithm problem. Therefore, we conclude that for any PPT algorithm $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}(pk_T, pk_P, tr) = 1] - \Pr[\mathcal{D}(pk_T, pk_P, \widehat{tr}) = 1]\right| \leq \Pr[\text{INEQ}],$$

which is also negligible in ℓ. □

Theorem 7.13 *The transfer protocol in our identification scheme* EDID = (*Setup*, *TKeyGen*, *PKeyGen*, *P*, *V*, *TA*, *AnyV*) *is a zero knowledge protocol.*

Proof: Let $\mathcal{S}$ be a simulator and $\mathcal{A}$ play the role of any arbitrary verifier V^*. Let tr be a transcript that $\mathcal{S}$ wants to prove a possession of a proof of transcript tr $\mathcal{S}$ simulates transcripts as follows.

1. First, $\mathcal{S}$ receives $\mathbb{T}$ from $\mathcal{A}$. Upon access to the random tape used by V^*, $\mathcal{S}$ obtains c' and d', where $T = g_1^{c'} g_2^{d'}$.

2. Then, $\mathcal{S}$ computes a response as follows.

   - Select $z \xleftarrow{\$} \mathbb{Z}_p^*$ and $D_1 \xleftarrow{\$} \mathbb{G}_1$.
   - Compute $m = H(pk_{OT}, \vartheta)$. Then compute $D_2 = \left(\frac{\hat{e}(g,g_1)}{\hat{e}(D_1, S_P g_1^m)}\right)^{c'} \hat{e}(g_2, S_P g_1^m)^z$.
   - Then $\mathcal{S}$ returns (D_1,D_2) to $\mathcal{A}$.

3. Upon receiving c and d, $\mathcal{S}$ aborts if $c \neq c'$ or $d \neq d'$. Otherwise, $\mathcal{S}$ replies with z.

Let tr_t be a transcript of the real transfer transcripts and $\widehat{tr_t}$ be a transcript of the simulated transfer transcripts. From the structure of protocol and the simulation process, the difference between the distribution of the real transcripts $\Im = \{tr_t\}$ and the simulated transcripts $\hat{\Im} = \{\widehat{tr_t}\}$ only happen when $c \neq c'$ or $d \neq d'$ as the challenge steps are revealed. Since both $c, d \xleftarrow{\$} \mathbb{Z}_p$, the probability that $c \neq c'$ or $d \neq d'$ but $T = g_1^{c'} g_2^{d'} = g_1^c g_2^d$ is equal to $1/p$. Therefore, the distance between the probability distribution of $\Im$ and $\hat{\Im}$ as a result of this is no more than $1/2^\ell$ where ℓ is a security parameter and $\ell \approx |p|$.
Therefore, the distance between the probability distribution of $\Im$ and $\hat{\Im}$ is

$$\left|\Pr[\langle P(sk_P), V^*\rangle(pk_P) = 1] - \Pr[S^{V^*}(pk_P) = 1]\right| < 1/2^\ell,$$

which is *negligible* for sufficiently large ℓ. □

7.8.2 Security Analysis for Impersonation

The following theorem presents a security analysis of our escrowed deniability identification scheme against impersonation under active attack. Before we describe this, we recall that confirmation evidence generated in the 2^{nd} round of the protocol is in fact a Boneh-Boyen basic short signature.

Theorem 7.14 *Our identification scheme EDID is secure against impersonation under active attack in the standard model, if the q-DL assumption holds.*

Proof: Suppose that there exists a PPT imp-aa adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for an EDID scheme such that the probability $\Pr[\mathrm{Expt}_{\mathcal{A}}^{\mathsf{imp-aa}}(\ell) = 1]$ is non-negligible. Then we can show that there exists a PPT adversary $\mathcal{F}$ to solve the q-DL problem using $\mathcal{A}$ as a subroutine. $\mathcal{F}$ is given $g, g^s, g^{s^2}, \cdots, g^{s^q} \in \mathbb{G}_1$ as input. $\mathcal{F}$ computes g_1 and S_P in the same way as in the proof of Lemma 1 in [BB04]. $\mathcal{F}$ then sets $g = g_1^{\gamma}$ and $g_2 = g^{\beta}$, where $\gamma, \beta \xleftarrow{\$} \mathbb{Z}_p$. Let $\mathfrak{OT} = \{pk_{OT,1}, sk_{OT,1}, ..., pk_{OT,q_H}, sk_{OT,q_H}\}$ be the list of pre-computed one-time public keys and secret keys. Let $\mathfrak{LM} = \{m_1 = H(pk_{OT,1}), ..., m_{q_H} = H(pk_{OT,q_H})\}$ be the list of hash values of the one-time public keys.

Identification oracle $\mathcal{OID}$: On input of a call, $\mathcal{F}$ simulates the prover as follows.

1. Obtain $\mathbb{T}$ from $\mathcal{A}_1$.
2. First, select random generators $\mathtt{T}', A_1', A_2', A_3' \xleftarrow{\$} \mathbb{G}_1$. Second, choose integers $a', b' \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $E_1' = \mathbb{U}^{a'}$; $E_2' = \mathbb{U}^{b'}$. Let $m \xleftarrow{\$} \mathfrak{LM}$. Next, compute $\theta = g_1^{1/(s+m)}$ as in the proof of Lemma 1 in [BB04]. Then compute $E_3' = \theta \mathbb{W}^{a+b}$. Obtain $pk_{OT} \in \mathfrak{LM}$, where $m = H(pk_{OT})$ then select $\kappa' \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $\bar{m}' = H(\mathtt{T}, \mathbb{T}, pk_P, E_1, E_2, E_3, A_1, A_2, A_3)$; $\vartheta' = g_a^{\frac{1}{\alpha+\kappa'\cdot\eta+\bar{m}'}}$. Finally, $\mathcal{OID}$ returns $(\mathtt{T}', pk_{OT}, \vartheta', \kappa', E_1', E_2', E_3', A_1', A_2', A_3')$.
3. $\mathcal{A}_1$ replies with c, d.
4. $\mathcal{F}$ checks the validity of c, d with respect to $\mathbb{T}$. If it is not valid, $\mathcal{F}$ aborts the current execution. Otherwise, it rewinds $\mathcal{A}_1$ to the second step and then computes as follows.
   (a) $z_s, a, b, z_a, z_b \xleftarrow{\$} \mathbb{Z}_p^*$; $E_1 = \mathbb{U}^a$; $E_2 = \mathbb{V}^b$; $\mathtt{T} = S_P^c g_1^{z_s}$; $A_1 = E_1^c \mathbb{U}^{z_a}$; $A_2 = E_2^c \mathbb{V}^{z_b}$.
   (b) Set $E_3 = E_3'$ and compute $A_3 = \left(\frac{\hat{e}(E_3, S_P g_1^m)}{\hat{e}(g, g_1)}\right)^c \hat{e}(\mathbb{W}, S_P g_1^m)^{z_a + z_b}$.
   (c) Select a random integer $\kappa \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $\bar{m} = H(\mathtt{T}, \mathbb{T}, pk_P, E_1, E_2, E_3, A_1, A_2, A_3)$; $\vartheta = g_a^{\frac{1}{\alpha+\kappa\cdot\eta+\bar{m}}}$.
   (d) Finally, $\mathcal{OID}$ returns $(\mathtt{T}, pk_{OT}, \vartheta, \kappa, E_1, E_2, E_3, A_1, A_2, A_3)$ to $\mathcal{A}$.
5. $\mathcal{A}$ replies with c', d'.
6. Check the validation of c', d' with $\mathbb{T}$ and check whether $c = c'$ and $d = d'$. If the above does not hold, $\mathcal{F}$ aborts the current execution. Otherwise, it returns $\hat{Z} = (z_s, z_a, z_b)$ to $\mathcal{A}$.

7. Finally, $\mathcal{F}$ records a transcript tr and a signature θ.

Now, $\mathcal{F}$ runs the eimp-aa experiment with $\mathcal{A}$. First, in the *Learning Phase*, the entire parameter is first initialised. The public-private key pair of provers is initially set to $pk_P = S_P$ and $sk_P = s$, then a public-private key pair of The TA is generated by running $(pk_T, sk_T) \leftarrow TKeyGen(1^\ell)$. In this phase, $\mathcal{A}$ plays the role of $\mathcal{A}_1$. $\mathcal{A}_1$ is given pk_P, pk_T, sk_T and access to $\boldsymbol{\mathcal{OID}}$. At the end of this phase, $\mathcal{A}_1$ outputs a state of information $\mathtt{St}$ and passes it to $\mathcal{A}_2$.

Now we move to the *Impersonation Phase*. On input of $\mathtt{St}$ from the *Learning Phase*, $\mathcal{A}_2$ runs the identification protocol to convince V (played by $\mathcal{F}$) to accept $\mathcal{A}_2$ as P. Note that the public parameter for $\mathcal{A}_2$ can be obtained from $\mathcal{A}_1$ in the previous phase. $\mathcal{A}_2$ then interacts with $\mathcal{F}$ as the following protocol:

- $(\mathcal{F} \rightarrow \mathcal{A}_2)$ Select random integers $c, d \in \mathbb{Z}_p^*$ and compute $\mathbb{T} = g_1^c g_2^d$. $\mathcal{F}$ sends $\mathbb{T}$ to $\mathcal{A}_2$.
- $(\mathcal{A}_2 \rightarrow \mathcal{F})$ Reply with $(\mathtt{T}, pk_{OT}, \vartheta, \kappa, E_1, E_2, E_3, A_1, A_2, A_3)$.
- $(\mathcal{F} \rightarrow \mathcal{A}_2)$ Respond with c, d.
- $(\mathcal{A}_2 \rightarrow \mathcal{F})$ Return $\hat{Z} = (z_s, z_a, z_b)$.

$\mathcal{A}$ wins the game if a transcript $tr = (\mathbb{T}, \mathtt{T}, pk_{OT}, \vartheta, \kappa, E_1, E_2, E_3, A_1, A_2, A_3, c, d, \hat{Z})$ from the above protocol passes validation. Due to the fact that $\mathcal{F}$ possesses γ, β, which are secret keys to solve the relationship between g, g_1 and g_2, with an overwhelming probability, $\mathcal{F}$ then rewinds $\mathcal{A}_2$ to the third step and replies to $\mathcal{A}_2$ with $c', d' \in \mathbb{Z}_p^*$ such that $c' \neq c$ and $d' \neq d$. $\mathcal{A}_2$ responds with $\hat{Z}' = (z'_s, z'_a, z'_b)$. Let $tr_2 = (\mathbb{T}, \mathtt{T}, pk_{OT}, \vartheta, \kappa, E_1, E_2, E_3, A_1, A_2, A_3, c', d', \hat{Z}')$ denote the second transcript. From these two transcripts, $\mathcal{F}$ compute s as the answer to q-DL problem.

Next, we conclude by presenting the probability of success of $\mathcal{A}$ succeeding in the impersonation. There are two events that trigger $\mathcal{F}$ to abort. We will show that $\mathcal{F}$ will abort the simulation with negligible probability. These two events occur in steps 4 and 6 of the identification oracle. The first event is that $c \neq c'; d \neq d'$ but $\mathbb{T} = g_1^c g_2^d = g_1^{c'} g_2^{d'}$. If this event occurs, then $\mathcal{A}_1$ can be used to solve the discrete logarithm problem, where it is given g, g^x as input to find x. We simply set $g_1 = g; g_2 = g^x$ then, upon receiving c, d, c' and d', we compute $x = (c - c')/(d' - d)$. Hence, this event occurs with negligible probability, underlying the fact that the discrete logarithm problem is difficult. The second event is when c, d is dishonestly

generated and is consequently not valid. Since the correctness of $\mathbb{T} = g_1^c g_2^d$ holds with negligible probability of error with regard to the first event, we can conclude that if $\mathcal{A}_1$ is correctly interacting with the identification oracle, then the second event will not occur. Therefore, we claim that $\mathcal{F}$ solves the q-DL problem with non-negligible probability by using $\mathcal{A}$.

To conclude, the explanation above shows that if there exists an adversary that breaks the impersonation of an EDID scheme under active attack, then we can use this adversary to solve the q-DL problem with non-negligible probability. Conversely, if the q-DL problem holds, then the EDID scheme is secure against impersonation under active attack. □

7.8.3 Security Analysis for Transferability

Theorem 7.15 *Our identification scheme* EDID $= (Setup, TKeyGen, PKeyGen, P, V, TA, AnyV)$ *is secure against transferability attacks only if the q-SDH problem hold under in the standard model.*

Proof: From the proof presented in Theorem 7.12 and Theorem 7.13, the identification protocol and the transfer protocol in our EDID scheme (computational) are zero knowledge protocols. Based on this, we can exclude a PPT tran adversary that can distinguish the simulated transcripts from the actual transcripts. We also exclude an adversary that, without help from a trusted third party, uses any means (or any protocol) to convince another party that the adversary has actually interacted with the prover to generate a transcript. If there exists an adversary as described above then that adversary seems in fact to be an adversary against deniability where proof has been provided in Theorem 7.12. Hence, the rest of our proof will show that, assuming that there exists a PPT tran adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT algorithm $\mathcal{F}$ that uses $\mathcal{A}$ to solve the q-SDH problem. We begin with $\mathcal{F}$ constructing oracles and setting up public parameters as follows.

Parameter setup: First, $\mathcal{F}$ is given $g, g^s, g^{s^2}, \cdots, g^{s^q} \in \mathbb{G}_1$ as input. Then, $\mathcal{F}$ computes g_1 and S_P in the same way as in the proof of Lemma 1 in [BB04]. $\mathcal{F}$ then sets $g = g_1^\gamma$ and $g_2 = g^\beta$, where $\gamma, \beta \xleftarrow{\$} \mathbb{Z}_p$. Next, $\mathcal{F}$ runs $TKeyGen$ to get the public key and private key of the trusted authority, which are $pk_T = (\mathbb{U}, \mathbb{V}, \mathbb{W})$ and $sk_T = (x, y)$ respectively. Let $\mathfrak{OT} = \{pk_{OT,1}, sk_{OT,1}, ..., pK_{OT,q_H-1}, sk_{OT,q_H-1}, pk^*_{OT}, sk^*_{OT}\}$ be the list of pre-computed one-time public

keys and secret keys. Let $\mathfrak{LM} = \{m_1 = H(pk_{OT,1}), ..., m_{q_H-1} = H(pk_{OT,q_H-1}), m^* = H(pk^*_{OT})\}$ be the list of hash values of the one-time public keys.

Identification oracle $\mathcal{OID}$: For every request for a transcript to $\mathcal{OID}$, except when $m = m_*$, $\mathcal{F}$ constructs identification oracle in the same way as the proof in Theorem 7.14. In the case of m_*, $\mathcal{F}$ changes the procedure for identification oracle and processes them as follows.

1. Obtain $\mathbb{T}$ from $\mathcal{A}_1$.
2. First, select random generators $\mathtt{T}', A'_1, A'_2, A'_3, E'_3 \xleftarrow{\$} \mathbb{G}_1$. Second, choose integers $a', b' \xleftarrow{\$} \mathbb{Z}^*_p$ and compute $E'_1 = \mathbb{U}^{a'}$; $E'_2 = \mathbb{U}^{b'}$. Obtain $pk^*_{OT} \in \mathfrak{LM}$ then select $\kappa' \xleftarrow{\$} \mathbb{Z}^*_p$ and compute $\bar{m}' = H(\mathtt{T},\mathbb{T},pk_P,E_1,E_2,E_3,\ A_1,A_2,A_3)$; $\vartheta' = g_a^{\frac{1}{\alpha+\kappa'\cdot\eta+\bar{m}'}}$. Finally, $\mathcal{OID}$ returns $(\mathtt{T}', pk_{OT}, \vartheta', \kappa', E'_1, E'_2, E'_3, A'_1, A'_2, A'_3)$.
3. $\mathcal{A}_1$ replies with c, d.
4. $\mathcal{F}$ checks the validity of c, d with respect to $\mathbb{T}$. If it is not valid, $\mathcal{F}$ aborts the current execution. Otherwise, it rewinds $\mathcal{A}_1$ to the second step and then computes as follows.
   (a) $z_s, a, b, z_a, z_b \xleftarrow{\$} \mathbb{Z}^*_p$, $E_3 \xleftarrow{\$} \mathbb{G}_1 : E_1 = \mathbb{U}^a$, $E_2 = \mathbb{V}^b$, $\mathtt{T} = S_P^c g_1^{z_s}$, $A_1 = E_1^c \mathbb{U}^{z_a}$; $A_2 = E_2^c \mathbb{V}^{z_b}$.
   (b) Compute $A_3 = \left(\frac{\hat{e}(E_3, S_P g_1^m)}{\hat{e}(g,g_1)}\right)^c \hat{e}(\mathbb{W}, S_P g_1^m)^{z_a+z_b}$.
   (c) Select a random integer $\kappa \xleftarrow{\$} \mathbb{Z}^*_p$ and compute $\bar{m} = H(\mathtt{T},\mathbb{T},pk_P,E_1,E_2,E_3,A_1,A_2,A_3)$; $\vartheta = g_a^{\frac{1}{\alpha+\kappa\cdot\eta+\bar{m}}}$.
   (d) Finally, $\mathcal{OID}$ returns $(\mathtt{T}, pk_{OT}, \vartheta, \kappa, E_1, E_2, E_3, A_1, A_2, A_3)$ to $\mathcal{A}$.
5. $\mathcal{A}$ replies with c', d'.
6. Check the validation of c', d' with $\mathbb{T}$ and check whether $c = c'$ and $d = d'$. If the above does not hold, $\mathcal{F}$ aborts the current execution. Otherwise, it returns $\hat{Z} = (z_s, z_a, z_b)$ to $\mathcal{A}$.
7. Finally, $\mathcal{F}$ records a transcript tr^* and a signature θ^*.

Open oracle $\mathcal{OPN}$: $\mathcal{F}$ constructs open oracle as follows.

- If tr is in the list of queried transcripts, then $\mathcal{OPN}$ returns an associated signature θ, except where $m = m^*$, it returns $\perp$.

- Otherwise, decrypt tr with sk_T and obtain θ and then check whether $\hat{e}(g, g_1) = \hat{e}(\theta, S_P g_1^m)$. If the answer is yes, then $\mathcal{OPN}$ returns θ. If it is not, then $\mathcal{OPN}$ returns $\perp$.

Let $\mathsf{param} = (\hat{e}, p, g, g_1, g_2)$ and $pk_S = S_P = g_1^s$. The private parameters for $\mathcal{F}$ are sk_{TA}, γ and β. Next, $\mathcal{F}$ simulates the *Learning Phase* by running $\mathcal{A}$ on input of (param,pk_P, pk_T,$\mathcal{OID}$, $\mathcal{OPN}$), and then operates as follows.

- On the request of tr, $\mathcal{A}_1$ runs the identification protocol with $\mathcal{OID}$ to obtain tr.
- On being requested to open tr, $\mathcal{A}_1$ arbitrarily sends tr to $\mathcal{OPN}$. $\mathcal{OPN}$ returns a signature θ on a message m from tr, if tr and θ are valid and tr is generated by $\mathcal{OPN}$. Otherwise, it returns a failure.

At the end of this phase, $\mathcal{A}_1$ outputs a state of information $\mathtt{St}$ and passes it on to $\mathcal{A}_2$. Now, we move to the *Convincing Phase*, where $\mathcal{F}$ plays the role of $AnyV$. Note that the public parameter for $\mathcal{A}_2$ can be obtained from $\mathcal{A}_1$ in the previous phase. $\mathcal{A}_2$ then interacts with $AnyV$ as the following protocol:

- $(AnyV \rightarrow \mathcal{A}_2)$ Select random integers $c, d \in \mathbb{Z}_p^*$ and compute $\mathfrak{T} = g_1^c g_2^d$. $\mathcal{F}$ sends $\mathfrak{T}$ to $\mathcal{A}_2$.
- $(\mathcal{A}_2 \rightarrow AnyV)$ Run $\{tr, D_1, D_2\} \leftarrow \mathcal{A}_2(\mathtt{St}, pk_P, pk_T, \mathsf{param})$ and reply with (tr, D_1, D_2).
- $(AnyV \rightarrow \mathcal{A}_2)$ Respond with c, d.
- $(\mathcal{A}_2 \rightarrow AnyV)$ Return $z \leftarrow \mathcal{A}_2(\mathtt{St}, pk_P, pk_T, \mathsf{param}, tr, D_1, D_2, c, d)$.

$\mathcal{A}$ wins the game if a transfer transcript $tr_t = (\mathfrak{T}, tr, D_1, D_2, c, d, z)$ from the above protocol passes the validation. Due to the fact that $\mathcal{F}$ possesses γ, β, which are secret keys to solve the relationship between g, g_1 and g_2, with a overwhelming probability, $\mathcal{F}$ then rewinds $\mathcal{A}_2$ to the third step and replies to $\mathcal{A}_2$ with $c', d' \in \mathbb{Z}_p^*$ such that $c' \neq c$ and $d' \neq d$. Then $\mathcal{A}_2$ responds with z', with $tr_t' = (\mathfrak{T}, tr, D_1, D_2, c', d', z')$ denoting the second transfer transcript.

Let q_H, q_O be the number of queries that $\mathcal{A}$ makes to the identification oracle and open oracle, respectively. Within the probability $\frac{1}{q_H}$, $\mathcal{A}$ processes the second phase with m_*. If $\mathcal{A}$ wins the game with m_*, then, from these two transcripts, $\mathcal{F}$ computes a signature θ^* on a message m_* as the answer to the q-SDH problem.

There are certain events that cause $\mathcal{F}$ to abort the simulation. We will now show that such events happen with negligible probability or that some are expected to occur with non-negligible probability. First, in the open oracle, the first event (E_1) is that $\mathcal{F}$ aborts the simulation when $m = m_*$. This event is already expected to happen within the probability $(1-1/(q_H))^{q_O} \geq 1/e$ where e is the natural logarithm. The other event (E_2) is also in the open oracle when tr is not in the list of transcripts produced by $\boldsymbol{\mathcal{OID}}$ but it passes verification. This means that $\mathcal{A}$ can produce a valid transcript. Then $\mathcal{A}$ can indeed be used to break the impersonation of our EDID scheme. Hence, from the proof presented in Theorem 7.14, the probability that $\mathcal{F}$ aborts in this event is negligible. For the events above, the probability that $\mathcal{F}$ does not abort the simulation is non-negligible, where $\Pr[E_1] + \Pr[E_2] \approx 1/e$. Hence, we claim that the probability that $\mathcal{A}$ wins the game is non-negligible if the probability of solving the q-SDH problem is non-negligible.

7.9 Conclusion

We started this chapter by providing a stronger definition of $\mathsf{CR1}^+$ attack for an identity-based identification scheme. The $\mathsf{CR1}^+$ attack is a stronger variant than $\mathsf{CR1}$ attack proposed in [BFGM01]. In $\mathsf{CR1}^+$ attack, an adversary is allowed to reset the prover (or its clones) to *any* state. In $\mathsf{CR1}$ attack, an adversary is allowed to reset the prover only to the initial state. Hence, the $\mathsf{CR1}$ attack in [BFGM01] is a special case of our $\mathsf{CR1}^+$ attacks. Then, we provided two identity-based identification schemes, which are secure under passive attack and secure against the $\mathsf{CR1}^+$ attack. The complexity assumption used in our proof is weaker than the state-of-the-art identification scheme by Kurosawa and Heng [KH05]. We also provided the computation comparison between our identity-based identification schemes and Kurosawa and Heng's identity-based identification schemes. We also introduced a new notion called *escrowed deniability* in an identification scheme. This notion bridges the gap between deniability and non-deniability in an identification scheme. We also provided a concrete scheme that satisfies this new notion. The security of our identification scheme ensures impersonation and transferability (escrowed deniability). Proof of these was also presented. In short, we believe the escrowed deniability property is an essential feature for identification schemes where the need for incorporation and disaffirmation is crucial.

Chapter 8

Conclusions and Further Works

This chapter concludes our contributions in this thesis. We answer the research questions mentioned in Chapter 1 and divide them in two main contributions, which are contributions to signature schemes and contributions to identification schemes.

8.1 Contribution to Signature schemes

Our contributions to signature schemes comprise three aspects.

First, we proposed a signature scheme that provides the privacy and anonymity of the signer. We also provided an algorithm to control the privacy of the signer.

Second, we proposed algorithms that allow a signer to assigns a set of verifiers such that only these verifiers can verify the signature on a message signed by the signer while others cannot do so.

Finally, we proposed an algorithm that provides fairness to the honest signers in multi-signatures.

8.1.1 Universal Designated Verifier Signature Schemes

A universal designated verifier signature scheme allows a signature holder to delegate a signature on a message that he obtains from a signer to any designated verifier. In other words, a signature holder is given the privilege of designating the signature to any verifier of his choice. In Chapter 4, we proposed two new notions: a "one-time universal designated verifier signature" and a "universal designated verifier signature with threshold-signers". One-time universal designated verifier signature schemes allow a signature holder to delegate a signature on a message that he obtains from a signer to only one designated verifier of his choice. If a signature holder generates a designated verifier signature more than once then the original signature that was

signed by the signer can be obtained by any party, and hence, the privacy of the signature is no longer guarantied. We presented a definition of a one-time universal designated verifier signature scheme and its security model in Section 4.2. We also provided a concrete construction of a one-time universal designated verifier signature scheme in Section 4.3 and its security analysis in Section 4.4. Universal designated verifier signatures with threshold-signers allow the privacy of both the signer and the signature holder to be preserved. The signature holder is allowed to provide anonymity for the signer(s) and the signature(s) that he has in his possession. The designated verifier only assures that a designated verifier signature is generated from the signature(s) that have been signed by a signer (or t signers) in the list of n signers. We presented a definition of a universal designated verifier signature with threshold-signers and its security model in Section 4.5. We presented a concrete construction of a universal designated verifier signature with threshold-signers in Section 4.6 and its security analysis in Section 4.7.

8.1.2 Policy Controlled Signature Schemes

In Chapter 5, we proposed three new notions: a "policy-controlled signature", a "universal policy-controlled signature" and a "multi-level controlled signature". A policy-controlled signature scheme allows multiple verifiers to verify a signature on certain messages signed by a signer; however, these verifiers must satisfy a policy assigned by the signer in order to verify this signature. In other words, the signature produced by the signer is verifiable only to those verifiers that satisfy certain policies specified by the signer. Meanwhile, the other party that does not satisfy the policies should not be able to verify this signature. A definition of a policy-controlled signature scheme and its security model were presented in Section 5.2. The proposed scheme was presented in Section 5.3. Its security analysis was described in Section 5.4.

The extension of policy-controlled signatures, namely, universal policy-controlled signatures and multi-level controlled signatures, were described in Sections 5.5 and 5.8. In universal policy-controlled signatures, a party called the policy holder is allowed to generate proof (a policy-controlled signature) of the possession of a signer's signature on certain messages. This signature is verifiable only to a verifier that satisfies the policy assigned by the policy signer. Our concrete scheme of universal policy-controlled signatures was presented in Section 5.6. Its security analysis was

described in Section 5.7.

The notion of multi-level controlled signatures eliminates the unnecessary chain of attributes in the policy when this can be assigned in a simple way, such as the number of the security level. Let "POLICY=(11-security level or 12-security level or 13-security level)" be an example of policies assigned by a signer in a policy-controlled signature scheme. In multi-level controlled signatures, we can simply assign the policy as "POLICY= more than or equal to 11-security level". Multi-level controlled signatures reduce the size of the signature and the number of attributes specified in the policy. A definition of a multi-level controlled signature scheme and its security model were presented in Section 5.8. The first proposed scheme with a constant size for a signer's private key was presented in Section 5.9. Its security analysis was described in Section 5.10. The second proposed scheme with a constant size for a verifier's credentials was presented in Section 5.11. Its security analysis was described in Section 5.12.

8.1.3 Fair Multi-Signature Schemes

Multi-signature schemes allow a number of parties together to generate a signature on some messages. This signature is the so-called "multi-signature". In Chapter 6, we constructed a new notion for multi-signature schemes called "fair multi-signatures". In this notion a group of signers together fairly generate a signature. In other words, every signer should be able to output a multi-signature if the protocol is complete. Otherwise, none of the signers can output a multi-signature. It can be seen clearly that this notion is an extension of a multi-signature scheme and it further strengthens the notion of multi-signature schemes. We pointed out that this cryptographic primitive is useful in many applications, in particular the ones that involve a multi-party signing process. We presented a security model for fair multi-signature schemes. We also provided a generic construction of a fair multi-signature scheme, which can be constructed from the verifiable encrypted signature scheme based on aggregate signatures. Two instantiations of our generic construction scheme with their security analysis were presented in Sections 6.5 and 6.6.

8.2 Contribution to Identification Schemes

An identification scheme allows a party called the prover to prove his/her identity to another party called the verifier. However, the other party cannot claim to be a prover. Hence, the security of an identification scheme aims to prevent impersonation at the least. In a public key setting, the private key is chosen and then the prover's public key (appearing as a random bit string) is randomly generated and this is used to represent the prover in a digital way. In an identity-based setting, a prover is allowed to select a public key that represents the identity of the prover, such as an email address. In Chapter 7, we proposed identity-based identification schemes, which are secure under passive attack and secure against $\mathsf{CR1}^+$ attack. We demonstrated that our identity-based identification schemes are more efficient than the state-of-the-art identification scheme proposed by Kurosawa and Heng [KH05] in Section 7.5.

We also introduced a new notion called "escrowed deniable identification" in Section 7.6. In escrowed deniable identification schemes, a trusted party is introduced who can provide evidence proving that a prover has participated in the generation of the identification transcript with a verifier. However, without the help of the trusted party, the verifier cannot provide this evidence. Thus, our identification scheme provides both deniability in general and non-deniability when this is required. We provided a concrete scheme in Section 7.7 and the security of our proposed scheme in Section 7.8.

8.3 Further works

The privacy of the signature holder in universal designated verifier signature schemes always holds even though the signature holder might act maliciously. Future works will focus on how to control or revoke the privacy of the signature holder when he misuses the signer's signature.

A policy-controlled signature scheme offers a huge impact in controlling and limiting the number of verifiers to verifier the signer's signature. One might hope to provide a generic construction for policy-controlled signatures, which are secure in the standard model.

For future works on identification schemes, we shall focus on the more efficient construction of the identification scheme secure against reset attack and the escrowed

identification scheme.

Bibliography

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002.

[AGH10] Esma Aïmeur, Sébastien Gambs, and Ai Ho. Towards a privacy-enhanced social networking site. In *ARES*, pages 172–179. IEEE Computer Society, 2010.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, pages 591–606, 1998.

[Ate04] Giuseppe Ateniese. Verifiable encryption of digital signatures and applications. *ACM Trans. Inf. Syst. Secur.*, 7(1):1–20, 2004.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Cachin and Camenisch [CC04], pages 56–73.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BC05] Carlo Blundo and Stelvio Cimato, editors. *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, volume 3352 of *Lecture Notes in Computer Science*. Springer, 2005.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *ACM Conference on Computer and Communications Security*, pages 449–458. ACM, 2008.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BFGM01] Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. Identification protocols secure against reset attacks. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, 2001.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. In *SCN*, volume 5229 of *LNCS*, pages 218–235. Springer, 2008.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[BM05] Walid Bagga and Refik Molva. Policy-based cryptography and applications. In Andrew S. Patrick and Moti Yung, editors, *Financial Cryptography*, volume 3570 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2005.

[BM06] Walid Bagga and Refik Molva. Collusion-free policy-based encryption. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 233–245. Springer, 2006.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Juels et al. [JWdV06], pages 390–399.

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Cachin and Camenisch [CC04], pages 268–286.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *LNCS*, pages 31–46. Springer, 2003.

[BP02] Mihir Bellare and Adriana Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Yung [Yun02], pages 162–177.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Yung [Yun02], pages 465–480.

[BSSC05] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. Cambridge University Press, New York, NY, USA, 2005.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *STOC*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CHY04] Sherman S. M. Chow, Lucas Chi Kwong Hui, and Siu-Ming Yiu. Identity based threshold ring signature. In Park and Chee [PC05], pages 218–232.

[CHYC04] Sherman S. M. Chow, Lucas Chi Kwong Hui, Siu-Ming Yiu, and K. P. Chow. Secure hierarchical identity based signature and its application. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *ICICS*, volume 3269 of *LNCS*, pages 480–494. Springer, 2004.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In Blundo and Cimato [BC05], pages 134–148.

[CSY06] Sherman S. M. Chow, Willy Susilo, and Tsz Hon Yuen. Escrowed linkability of ring signatures and its applications. In Phong Q. Nguyen,

editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2006.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. IT-22(6):644–654, November 1976.

[DHP07] Catherine Dwyer, Starr R. Hiltz, and Katia Passerini. Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. In *Proceedings of the Thirteenth Americas Conference on Information Systems*, August 2007.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[DP06] Yevgeniy Dodis and Prashant Puniya. On the relation between the ideal cipher and the random oracle models. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 184–206. Springer, 2006.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.

[Fre05] David Freeman. Pairing-based identification schemes. technical report HPL-2005-154, Hewlett-Packard Laboratories, August 2005.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 181–200. Springer, 2007.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT*, pages 123–139, 1999.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. 17(2):281–308, April 1988. Special issue on cryptography.

[Gol90] Shafi Goldwasser, editor. *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*. Springer, 1990.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[Gol05] Oded Goldreich. Foundations of cryptography: a primer. *Found. Trends Theor. Comput. Sci.*, 1(1):1–116, 2005.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *EUROCRYPT*, pages 123–128, 1988.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 466–481. Springer, 2002.

[HMSZ05] Xinyi Huang, Yi Mu, Willy Susilo, and Futai Zhang. Short designated verifier proxy signature from pairings. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuan-Shun Dai, and Laurence Tianruo Yang,

editors, *EUC Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 835–844. Springer, 2005.

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[HRL09] Lein Harn, Jian Ren, and Changlu Lin. Efficient identity-based gq multisignatures. *Int. J. Inf. Sec.*, 8(3):205–210, 2009.

[HS03] Javier Herranz and Germán Sáez. Forking lemmas for ring signature schemes. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279. Springer, 2003.

[HSMW06] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Universal designated verifier signature without delegatability. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2006.

[HSMW08] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Secure universal designated verifier signature without random oracles. *Int. J. Inf. Sec.*, 7(3):171–183, 2008.

[HSMZ06] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Restricted universal designated verifier signature. In Jianhua Ma, Hai Jin, Laurence Tianruo Yang, and Jeffrey J. P. Tsai, editors, *UIC*, volume 4159 of *Lecture Notes in Computer Science*, pages 874–882. Springer, 2006.

[HWLZ08] Qiong Huang, Duncan S. Wong, Jin Li, and Yiming Zhao. Generic transformation from weakly to strongly unforgeable signatures. *J. Comput. Sci. Technol.*, 23(2):240–252, 2008.

[HYWS08] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. *RSA Conference 2008, Cryptographers' Track (CT-RSA 2008), Lecture Notes in Computer Science 4964*, pages 106 – 120, 2008.

[IN83] K Itakura and K Nakamura. A public key cryptosystem suitable for digital multisignatures. *NEC Research and Development*, 71:1–8, 1983.

[IT05] Toshiyuki Isshiki and Keisuke Tanaka. An (n-t)-out-of-n threshold ring signature scheme. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 406–416. Springer, 2005.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 1997.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. *Advances in Cryptology - Eurocrypt '96, Lecture Notes in Computer Science 1070*, pages 143 – 154, 1996.

[JWdV06] Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors. *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*. ACM, 2006.

[KH04] Kaoru Kurosawa and Swee-Huay Heng. From digital signature to id-based identification/signature. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 2004.

[KH05] Kaoru Kurosawa and Swee-Huay Heng. Identity-based identification without random oracles. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *ICCSA (2)*, volume 3481 of *Lecture Notes in Computer Science*, pages 603–613. Springer, 2005.

[KH06] Kaoru Kurosawa and Swee-Huay Heng. The power of identification schemes. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 364–377. Springer, 2006.

[KK02a] Myungsun Kim and Kwangjo Kim. A new identification scheme based on gap diffie-hellman problem. In *The 2002 Symposium on Cryptography and Information Security*, 2002.

[KK02b] Myungsun Kim and Kwangjo Kim. A new identification scheme based on the bilinear diffie-hellman problem. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 2002.

[KL10] Jon M. Kleinberg and Katrina Ligett. Information-sharing and privacy in social networks. *CoRR*, abs/1003.0469, 2010.

[LLP05] Yong Li, Helger Lipmaa, and Dingyi Pei. On delegatability of four designated verifier signatures. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *ICICS*, volume 3783 of *Lecture Notes in Computer Science*, pages 61–71. Springer, 2005.

[LLQ06] Fabien Laguillaumie, Benoît Libert, and Jean-Jacques Quisquater. Universal designated verifier signatures without random oracles or non-black box assumptions. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Cachin and Camenisch [CC04], pages 74–90.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, volume 4004 of *LNCS*, pages 465–485. Springer, 2006.

[LV04] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In Blundo and Cimato [BC05], pages 105–119.

[LV07a] Fabien Laguillaumie and Damien Vergnaud. Multi-designated verifiers signatures: anonymity without encryption. *Inf. Process. Lett.*, 102(2-3):127–132, 2007.

[LV07b] Fabien Laguillaumie and Damien Vergnaud. On the soundness of restricted universal designated verifier signatures and dedicated signatures. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC*, volume 4779 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2007.

[LW04] Joseph K. Liu and Duncan S. Wong. On the security models of (threshold) ring signature schemes. In Park and Chee [PC05], pages 204–217.

[LW06] Jin Li and Yanming Wang. Universal designated verifier ring signature (proof) without random oracles. In Xiaobo Zhou, Oleg Sokolsky, Lu Yan, Eun-Sun Jung, Zili Shao, Yi Mu, Dong Chun Lee, Daeyoung Kim, Young-Sik Jeong, and Cheng-Zhong Xu, editors, *EUC Workshops*, volume 4097 of *LNCS*, pages 332–341. Springer, 2006.

[LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005.

[LWW03] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. A separable threshold ring signature scheme. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 12–26. Springer, 2003.

[Mao97] Wenbo Mao. Verifiable escrowed signature. In Vijay Varadharajan, Josef Pieprzyk, and Yi Mu, editors, *ACISP*, volume 1270 of *Lecture Notes in Computer Science*, pages 240–248. Springer, 1997.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2001.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[Oka88] Tatsuaki Okamoto. A digital multisignature schema using bijective public-key cryptosystems. *ACM Trans. Comput. Syst.*, 6(4):432–441, 1988.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[OO88] Kazuo Ohta and Tatsuaki Okamoto. A modification of the fiat-shamir scheme. In Goldwasser [Gol90], pages 232–243.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.

[OS90] H. Ong and Claus-Peter Schnorr. Fast signature generation with a fiat shamir-like scheme. In *EUROCRYPT*, pages 432–440, 1990.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer, 2003.

[PC05] Choonsik Park and Seongtaek Chee, editors. *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of *Lecture Notes in Computer Science*. Springer, 2005.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[RG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 112–121. ACM, 2005.

[RGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Juels et al. [JWdV06], pages 400–409.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 228–245. Springer, 2007.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2003.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[Sho99] Victor Shoup. On the security of a practical identification scheme. *J. Cryptology*, 12(4):247–260, 1999.

[SMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In Goldwasser [Gol90], pages 269–282.

[SPS10] Anna Cinzia Squicciarini, Federica Paci, and Smitha Sundareswaran. Prima: an effective privacy protection mechanism for social networks.

In Dengguo Feng, David A. Basin, and Peng Liu, editors, *ASIACCS*, pages 320–323. ACM, 2010.

[Sti06] Douglas R. Stinson. Some observations on the theory of cryptographic hash functions. *Des. Codes Cryptography*, 38(2):259–277, 2006.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2007.

[SZM04] Willy Susilo, Fangguo Zhang, and Yi Mu. Identity-based strong designated verifier signature schemes. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2004.

[THS+09] Pairat Thorncharoensri, Qiong Huang, Willy Susilo, Man Ho Au, Yi Mu, and Duncan Wong. Escrowed deniable identification schemes. In Dominik Ślęak, Tai hoon Kim, Wai-Chi Fang, and Kirk P. Arnett, editors, *Security Technology*, volume 58 of *Communications in Computer and Information Science*, pages 234–241. Springer, November 2009.

[THS+10] Pairat Thorncharoensri, Qiong Huang, Willy Susilo, Man Ho Au, Yi Mu, and Duncan Wong. Escrowed deniable identification schemes. *International Journal of Security and Its Applications*, 4(1):49–67, January 2010.

[TSM08] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. How to balance privacy with authenticity. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 184–201. Springer, 2008.

[TSM09a] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. Identity-based identification scheme secure against concurrent-reset attacks without random oracles. In Heung Youl Youm and Moti Yung, editors, *WISA*, volume 5932 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2009.

[TSM09b] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. Policy-controlled signatures. In Sihan Qing, Chris J. Mitchell, and Guilin Wang, editors, *ICICS*, volume 5927 of *Lecture Notes in Computer Science*, pages 91–106. Springer, 2009.

[TSM09c] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. Universal designated verifier signatures with threshold-signers. In Tsuyoshi Takagi and Masahiro Mambo, editors, *IWSEC*, volume 5824 of *Lecture Notes in Computer Science*, pages 89–109. Springer, 2009.

[TWC+04] Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 384–398. Springer, 2004.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[Yun02] Moti Yung, editor. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[YWW04] Gang Yao, Guilin Wang, and Yong Wang. An improved identification scheme. In *Coding, Cryptography and Combinatorics*, volume 23 of *Progress in Computer Science and Applied Logic*, Birkhauser Verlag, Basel, Switzerland, 2004. Birkhauser Verlag.

[ZFI05] Rui Zhang, Jun Furukawa, and Hideki Imai. Short signature and universal designated verifier signature without random oracles. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 483–498, 2005.

[ZK02] Fangguo Zhang and Kwangjo Kim. ID-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2002.

Index