2010

Contributions to pairing-based cryptography

Tsz Hon Yuen
University of Wollongong

Contributions to Pairing-based Cryptography

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Tsz Hon Yuen

School of Computer Science and Software Engineering
November 2010

Dedicated to
My mother and my father

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.

Tsz Hon Yuen
November 15, 2010

Abstract

Pairing-based cryptography is an active research area in cryptography in the last decade. Pairings are bilinear mappings defined over cyclic groups wherein the discrete logarithm problem is hard. The bilinear property of pairings enables researchers to solve open problems like the construction of practical identity-based encryption, or short signatures without random oracles. Pairings can also be used to construct new cryptographic primitives.

This thesis contributes to the pairing-based cryptography in three areas. Firstly, we show that pairings can be used to construct efficient and provably secure digital signature schemes. We give the first convertible undeniable signatures without random oracles, and the first concrete sanitisable signatures without random oracles. We also construct a new signature primitive called *concinnous signatures*, which is designed to facilitate fair exchange of digital signatures without any trusted third party.

Secondly, we analyse the identity-based cryptosystems which extensively use pairings. We mainly focus on the key escrow problem of identity-based cryptography. We propose the notion of *escrow-free identity-based signatures*. Furthermore, we discuss the impossibility of ideal escrow-free identity-based encryption. After that, we investigate the best defence against the key escrow problem of identity-based encryption. We categorise the existing solutions into *preventive measure* and *blaming mechanism*. In the category of preventive measure, we propose the notion of *fully anonymous identity-based encryption*. In the category of blaming mechanism, we also construct a new accountable-authority identity-based encryption.

Finally, we construct new cryptographic primitives and frameworks using pairings. We give new instantiations and applications of lossy trapdoor function. We give a new cryptographic primitive called *two-tier trapdoor functions*. From two-tier trapdoor functions, we construct a new encryption primitive called *two-tier encryption*. It is a generalisation of a number of encryption schemes, including identity-based encryption. We also propose a cryptographic treatment of publish/subscribe systems.

Acknowledgement

First of all, I would like to thank my supervisors Willy Susilo and Yi Mu. They are very supportive to both my research and give me valuable advices and innovative ideas to my works. I am also grateful for their assistant to help me adapting the life in the university.

I would like to express my gratitude to the researchers who collaborate with me during my Ph.D. study, including Man Ho Au, Qiong Huang, Joseph K. Liu, Duncan S. Wong and Guomin Yang.

Last but certainly not least, I would like to thank my family for their love and support throughout my study.

Publications

In this thesis, the following publications published in refereed conference are included:

- T. H. Yuen, W. Susilo, and Y. Mu. Cryptographic treatment of publish/subscribe systems. In S.-H. Heng, R.N. Wright, and B.-M. Goi, editors, *CANS 2010*, volume 6467 of LNCS, pages 201220. Springer, 2010.
- T. H. Yuen, W. Susilo, and Y. Mu. How to construct identity-based signatures without the key escrow problem. In F. Martinelli and B. Preneel, editors, *EuroPKI 2009*, volume 6391 of LNCS, pages 286301. Springer, 2010.
- T. H. Yuen, W. Susilo, J. K. Liu, and Y. Mu. Sanitizable signatures revisited. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *CANS 2008*, volume 5339 of LNCS, pages 80-97. Springer, 2008.
- T. H. Yuen, M. H. Au, J. K. Liu, and W. Susilo. (Convertible) undeniable signatures without random oracles. In S. Qing, H. Imai, and G. Wang, editors, *ICICS 2007*, volume 4861 of LNCS, pages 83-97. Springer, 2007.

The following paper is published in journal:

- T. H. Yuen, W. Susilo, and Y. Mu. How to construct identity-based signatures without the key escrow problem: Formal definitions and constructions. *Int. J. Inf. Secur.*, 9(4):297-311, 2010.

The following papers are currently under review by the editors of journals:

- T. H. Yuen, W. Susilo, and Y. Mu. Lossy trapdoor functions: New instantiations and applications. Submitted to *Information and Computation*.
- T. H. Yuen, W. Susilo, and Y. Mu. Two-tier encryption and two-tier trapdoor functions. Submitted to *Theoretical Computer Science*.

- T. H. Yuen, W. Susilo, and Y. Mu. Cryptographic treatment of publish/subscribe systems. Submitted to *IEEE Transactions on Information Forensics and Security*.

The following manuscripts are currently under submission to conferences:

- T. H. Yuen, W. Susilo, Duncan S. Wong and Qiong Huang. Concinnous signatures: Fair exchange of digital signatures.
- T. H. Yuen, W. Susilo, and Y. Mu. Impossibility to ideal escrow-free identity-based encryption.
- T. H. Yuen, W. Susilo, and Y. Mu. On the anonymity of identity-based encryption.
- T. H. Yuen, W. Susilo, and Y. Mu. Black-box accountable authority IBE revisited.

Other publications published during my Ph.D. study which are not included in this thesis:

- T. H. Yuen, Q. Huang, Y. Mu, W. Susilo, D. S. Wong, and G. Yang. Efficient non-interactive range proof. In H.Q. Ngo, editor, *COCOON 2009*, volume 5609 of LNCS, pages 138-147. Springer, 2009.
- M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Certificate based (linkable) ring signature. In E. Dawson and D. S. Wong, editors, *ISPEC 2007*, volume 4464 of LNCS, pages 79-92. Springer, 2007.
- J. Li, T. H. Yuen, K. Kim. Practical threshold signatures without random oracles. In W. Susilo, J. K. Liu, and Y. Mu, editors, *ProvSec 2007*, volume 4784 of LNCS, pages 198-207. Springer, 2007.

Contents

List of Tables

List of Figures

List of Notations

Below introduces the notations commonly used through out the rest of the thesis. Some notation will also be defined locally near its first use, while other notation will be used without further definition.

$S_1 \cup S_2$	union of sets S_1 and S_2
$S_1 \backslash S_2$	difference of sets S_1 and S_2
$S_1 \subseteq S_2$	S_1 is a subset of S_2
$x \in S, x \notin S$	element x (not in) set S
$x \in_R S$	sampling element x uniformly random in set S
$L_1 \prec L_2$	L_1 is polynomial-time many-one reducible to L_2
$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	sets of natural numbers, integers, and real numbers
$\mathbb{Z}^+$	set of positive integers
$\mathbb{Z}_n$	integers modulo n
$\mathbb{Z}_n^*$	multiplicative group of integers modulo n
$a \pmod b$	modulo operation: remainder of a divided by b
$\forall$	for all
$\exists$	there exists
$\vee, \wedge, \neg$	boolean operators OR, AND, and NOT
$\mathrm{ord}(\mathbb{G})$	order of a group $\mathbb{G}$
$\Pr[E]$	probability of event E occurring
$E_1 \vert E_2$	event E_1 occurring given event E_2
$\vert s \vert$	number of elements in s if s is a finite set, or the length of s if s is a string, or the bit-length/size of s if s is an integer.
1^k	the string of k ones.
$s_1 \vert\vert s_2$	string s_1 concatenate with string s_2.
$\binom{n}{r}, C_r^n$	binomial coefficient.

Abbreviations

Below introduces the abbreviations commonly used through out the rest of the thesis. They will be defined locally near its first use.

ABO:	All-but-one
A-IBE:	Accountable Authority Identity-based Encryption
CBPS:	Content-based Publish/subscribe System
CCA:	Chosen Ciphertext Attack
CDH:	Computational Diffie-Hellman
CRS:	Common Reference String
DL:	Discrete Logarithm
DLIN:	Decision Linear
IBE:	Identity-based Encryption
IBS:	Identity-based Signatures
NIZK:	Non-interactive Zero Knowledge
PKG:	Private Key Generator
PoK:	Proof of Knowledge
Pub/sub	Publish/subscribe
ROM:	Random Oracle Model
SDH:	Strong Diffie-Hellman
SoK:	Signature of Knowledge
TDF:	Trapdoor Function

Chapter 1

Introduction

Public key cryptography plays an important role in the security of the Internet. Some public key cryptosystems allow confidential communication over the Internet, and some provide the authenticity of messages over the Internet. Public key cryptography is employed in Internet standards such as Transport Layer Security (TLS) and Pretty Good Privacy (PGP).

The first public key cryptosystem is the Diffie-Hellman key exchange proposed in 1976 [71]. The RSA cryptosystem was published by Rivest, Shamir and Adleman in 1978 [180]. One of the differences between these two systems is that the Diffie-Hellman cryptosystem works in cyclic groups with known order, while the RSA cryptosystem works with groups which are not cyclic and whose order is unknown to the attacker.

Informally speaking, a pairing, or bilinear map, is a map from two cyclic groups into another cyclic group. The cyclic groups equipped with pairings are called the *bilinear groups*. The most important property of pairings is the *bilinearity*. In cryptography, pairings were firstly introduced for cryptanalytic purposes. Original research by Menezes, Okamoto and Vanstone [148], and by Frey, Müller and Rück [85], pointed out that the Weil and Tate pairings could be used for cryptanalytic purposes, undermining the security of certain types of elliptic curves. The bilinear property of pairings also implies that the decisional Diffie-Hellman problem is easy in bilinear groups.

Nevertheless, Sakai, Ohgishi and Kasahara [185] and Joux [124] independently observed that these very same condemned elliptic curves had in fact useful cryptographic properties. Practical identity-based encryption was an open problem since the concept of identity-based cryptography was proposed by Shamir [189] in 1984. In 2001, Boneh and Franklin [34] came up with an elegant simple solution to the problem of identity-based encryption using pairings. Since then, there are lots of new cryptographic protocols, regardless whether they are identity-based or not, which use pairings. There are a few reasons for the popularity of pairings in cryptography:

- The key size is small when comparing to the traditional RSA-based cryptosystem. For 80-bit symmetric security level, pairing-based cryptosystems use 171-bit bilinear groups while the RSA-based cryptosystems use 1024-bit modulus. As a result, the signature size and the ciphertext size in pairing-based cryptosystems can be very small. For example, the BLS signatures [38] are about half the size of the Digital Signature Algorithm (DSA) signatures. The short group signatures in [31] are approximately the size of a standard RSA signature with the same security.

- The bilinear property is useful in decryption and verification algorithms. The bilinear property is crucial to construct the first practical identity-based encryption in [34]. In addition, pairings are often used to verify the validity of digital signatures in pairing-based cryptography. Prior signature schemes often use hash functions to verify signatures. Therefore, researchers can construct new digital signatures in pairing-based cryptography without using the hash-and-sign paradigm or the Fiat-Shamir paradigm [78].

- There are some pairing-related intractability assumptions which are helpful to construct cryptosystems without random oracles. For example, the q-Strong Diffie-Hellman assumption is used to construct short signatures without random oracles [29]; the Decisional Bilinear Diffie-Hellman assumption is used to construct identity-based encryption without random oracles [206].

On the other hand, there are some concerns for the use of pairing-based cryptosystems:

- The pairing computation requires more time and computational power. Therefore systems with limited computational power, such as wireless sensor network, cannot perform the pairing computation efficiently.

- The pairing-related intractability assumptions are less examined than the traditional RSA or discrete logarithm assumption. In 2006, Cheon [64] proved that the security of the q-Strong Diffie-Hellman problem on an abelian group of order p can be reduced up to $O(\log p \cdot p^{1/3})$ (resp. $O(\log p \cdot p^{1/3})$) for large q if $p-1$ (resp. $p+1$) has a divisor $d = O(p^{1/2})$ (resp. $d = O(p^{1/3})$). Because of this, Cheon suggested to use 220-bit prime p for 80-bit symmetric security level.

1.1 Motivations of the Thesis

In this thesis, we explore the pairing-based cryptosystems in three different directions. Firstly, pairings are useful to construct new digital signature schemes, which can be very efficient (e.g. the BLS short signatures [38]), or secure in the standard model (e.g. the BB short signatures [29]). There are a number of variants of signature schemes which use pairings to achieve security in the standard model, such as group signatures [46] and ring signatures [67]. It is natural to ask if pairings can be used to construct other types of digital signatures, which may lead to more efficient or more secure constructions? What can we achieve if we use pairings in different types of digital signatures?

Secondly, we focus on the use of pairings in the development of identity-based cryptosystems. Pairings were used in cryptography to solve the open problem of practical identity-based encryption for the first time in 2001. Pairings are essential to the development of most of the identity-based cryptosystems. However, identity-based cryptosystems suffer from a major drawback: there exists a private key generator (PKG) who is responsible to generate the secret key for all users in the cryptosystem. As a result, the PKG must be fully trusted. What will happen if the PKG is malicious? This is known as the *key escrow* problem. What is the best possible defence against the malicious PKG? What can be done in identity-based encryption and identity-based signatures respectively to deal with the key escrow problem?

Thirdly, pairings can be used to construct new cryptographic primitives and cryptosystems. Pairings are used to instantiate new general primitives in public key cryptography, like accumulator [162] and non-interactive zero-knowledge proof system [106]. Is it possible to use pairings to instantiate other general primitives? On the other hand, pairings are also used to construct new cryptographic primitives, like simulatable verifiable random function [58], attribute-based encryption [183] and predicate encryption [127]. In particular, attribute-based encryption and predicate encryption can be viewed as the generalised version of identity-based encryption. The identity of the user becomes the attribute or the predicate function. These generalisations have a wider range of applications in the real world, such as the fine-grained access control. Can we use pairings to further generalise the identity-based encryption?

1.2 Thesis Organisation

The thesis is divided into seven distinct parts.

- Chapter 2 contains fundamental materials in algebra, number theory, complexity theory and information theory, which will be used in the rest of the thesis.
- Chapter 3 reviews the literature on works related to our thesis.
- Chapter 4 contains two new constructions of digital signatures using pairings, namely undeniable signatures and sanitisable signatures. The undeniable signature scheme is modified from the result published in ICICS 2007 [210][1]. The sanitisable signature scheme appeared in CANS 2008 [211]. Finally, we propose a new signature primitive called *concinnous signatures*, which is designed to facilitate the fair exchange of digital signatures without trusted third party.
- Chapter 5 mainly focuses on the key escrow problem in identity-based encryption and identity-based signatures. In addition, we also present a new construction of accountable-authority identity-based encryption. All of these constructions use pairings. The escrow-free identity-based signature scheme was published in EuroPKI 2009 [213] and its extension was published in [214].
- In Chapter 6, we propose new cryptographic protocols and applications using pairings. They include:
  - new instantiations and applications of lossy trapdoor function;
  - a new cryptographic primitive called *two-tier trapdoor function*, which is a trapdoor function with a special two-tier structure;
  - a new encryption protocol called *two-tier encryption*, which generalises a number of encryption schemes including identity-based encryption.
  - a new cryptographic framework and instantiation of publish/subscribe system, which has a number of practical applications in the Internet. It is published in CANS 2010 [212].
- Chapter 7 concludes our contributions and discusses the open problems in pairing-based cryptography.

[1]There are a few reasons for the modification. Firstly, one of our basic building block from Kurosawa and Heng [133] was proved insecure by Ogata *et al.* [165]. Secondly, Phong *et al.* [176] pointed out a flaw of our scheme in [210]. Thirdly, we are able to construct a more efficient construction from the original scheme, which can avoid the flaw of the scheme in [210].

Chapter 2

Definitions

Our goal in this Chapter is to provide the necessary background and foundations of cryptography that will be used in the subsequent chapters. We first present an introduction to the topics of algebra, number theory, complexity theory and information theory. We also review some intractability assumptions that will be used in the subsequent Chapters.

2.1 Algebra and Number Theory

Algebra and Number Theory are the mathematical foundation of modern cryptography. Numerous cryptographic algorithms are designed around results from them. They are also the cornerstone of (provable) security of cryptographic schemes.

2.1.1 Groups

First recall the definition of a group (a cyclic group in particular) and some other related notions.

Definition 1 (Group)**.** *A group $(G, \circ)$ is a set G together with an operation $\circ$ that satisfies:*

1. *Closure:* $\forall a, b \in G : a \circ b \in G$.
2. *Associativity:* $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$.
3. *Identity:* $\exists$ *unique element* $e \in G : \forall a \in G : a \circ e = e \circ a = a$.
4. *Inverse:* $\forall a \in G : \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$.

For simplicity, denote $a^i = \underbrace{a \circ a \circ \cdots \circ a}_{i}$. A group is *Abelian* if for all $a, b \in G$, $a \circ b = b \circ a$. For example, $(\mathbb{Z}, +)$ is an Abelian group.

We denote e as the *identity element* in G, such that $a \circ e = e \circ a = a$ for all $a \in G$.

Definition 2 (Group Order). *The order of the element $a \in G$ is the least positive integer i satisfying $a^i = e$, and it is denoted by $ord(a)$. If such a number does not exist, then the order of a is defined to be ∞.*

Definition 3 (Cyclic Group). *A group G is cyclic if there exists an element $a \in G$ such that for any $b \in G$, there exists an integer $i \geq 0$ such that $b = a^i$. Element a is called the generator of G. G is called the cyclic group generated by a. Order of the group G is the order of generator a.*

Definition 4 (Subgroup). *Let $(G, \circ)$ be a group. We say that $(H, \circ)$ is a subgroup of G if $H \subseteq G$ and $(H, \circ)$ is a group.*

Definition 5 (Ring). *A ring $(R, +, \cdot)$ is a set R together with operations $+$ and $\cdot$ that satisfies:*

1. *$(R, +)$ is Abelian group, with 0 as identity.*
2. *$(R, \cdot)$ satisfies closure, associativity with identity 1, $1 \neq 0$.*
3. *Commutative: $\forall a, b \in G : a \cdot b = b \cdot a$.*
4. *Distribution: $\forall a, b, c \in G : a \cdot (b + c) = a \cdot b + a \cdot c$.*

Definition 6 (Field). *If the non-zero elements of a ring forms a group under multiplication, then the ring is called a field.*

Notice that $\mathbb{Z}_p$ is a finite field of prime order. We usually denote it as $\mathbb{F}_p$.

2.1.2 Elliptic Curves

Elliptic curves are defined over finite fields. For examples, in a prime field $\mathbb{F}_p$ with $p > 3$, we have a curve:

$$E : y^2 = x^3 + ax + b \qquad \mod p$$

where a and b are constants satisfying $4a^3 + 27b^2 \neq 0 \mod p$. Then we have points $P = (x, y)$ on the curve, together with $\mathcal{O} = (x, \infty)$ the point at infinity.

We denote E be an Abelian group under the operation "+" defined as follows.

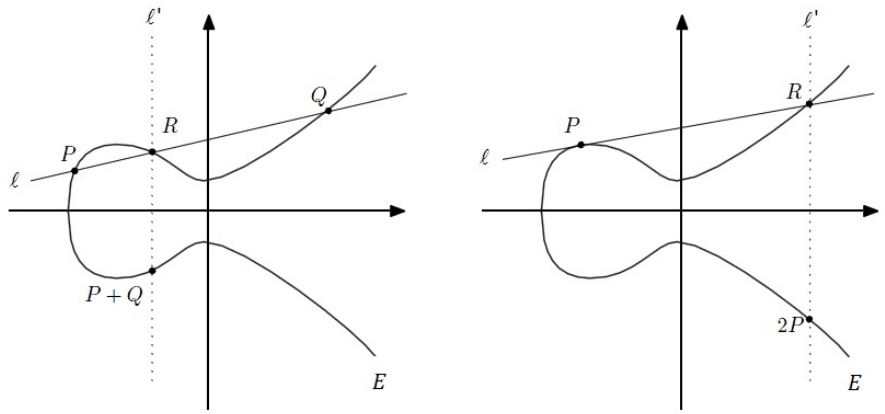


Figure 2.1: Elliptic curve operations over the real number field $\mathbb{R}$: addition of two points $P + Q$ (left) and doubling of a point P (right).

Definition 7 (Elliptic Curve Group Operation)**.** *Let $P, Q \in E$, ℓ be the line containing P and Q (tangent line to E if $P = Q$), and R, the third point of intersection of ℓ with E. Let ℓ' be the line connecting R and $\mathcal{O}$. Then $P+Q$ is the point such that ℓ' intersects E at R, $\mathcal{O}$ and $P + Q$.*

We write $P + Q = -R$. Then we have the point multiplication for $k \in \mathbb{Z}$ defined as:

$$[k]P = \begin{cases} \underbrace{P + P + \ldots + P}_{k} & \text{for } k > 0, \\ \mathcal{O} & \text{for } k = 0, \\ [-k](-P) & \text{for } k < 0. \end{cases}$$

The addition and doubling of a point on an elliptic curve over the real number field $\mathbb{R}$ is illustrated in Figure 2.1.2.

2.1.3 Pairings

We now briefly describe the notion of cyclic groups equipped with pairings, or *bilinear groups* for short. Following common cryptographic conventions, we employ the multiplicative notation for the group operations and write 1 for the identity element.

Let $\mathbb{G}$, $\hat{\mathbb{G}}$ and $\mathbb{G}_T$ be cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $\hat{g}$ be a generator of $\hat{\mathbb{G}}$. The function $\hat{e} : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is a bilinear map (pairings) if it

satisfies the following conditions.

- *Bilinearity*: For all $a, b \in \mathbb{Z}$, $\hat{e}(g^a, \hat{g}^b) = \hat{e}(g, \hat{g})^{ab}$.
- *Non-degeneracy*: $\hat{e}(g, \hat{g}) \neq 1$.
- *Computability*: It is efficient to compute $\hat{e}(u, v)$ for all $u \in \mathbb{G}, v \in \hat{\mathbb{G}}$.

The groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ are called the *bilinear groups*, and $\mathbb{G}_T$ is called the target group.

Types of Pairings

Galbraith *et al.* [87] proposed a classification of pairings. It depends on the efficiency of computing the isomorphism $\phi : \hat{\mathbb{G}} \to \mathbb{G}$ and its inverse ϕ^{-1}.

1. Type 1: The isomorphism ϕ and its inverse ϕ^{-1} are both efficiently computable. We can simply rewrite the pairing as a *symmetric pairing*: $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The symmetric pairings are commonly used in research papers. They have the advantage of simplifying the notation. However, they are generally less compact and less efficient.
2. Type 2: The isomorphism ϕ is efficiently computable, but not ϕ^{-1}.
3. Type 3: The isomorphism ϕ and its inverse ϕ^{-1} are not efficiently computable.

Efficient Constructions

The first efficient algorithm for computing certain pairings (and a class of functions on elliptic curves) was proposed by Miller in 1984, and eventually published in [153]. These pairings are known as the Weil and the Tate pairings. Tate pairing can be computed faster than Weil pairing. Recently, Eta pairing [11] and Ate pairing [113], and their generalisation [112] have also been proposed.

Pairing-friendly Elliptic Curves

Elliptic curves with small embedding degree and large prime-order subgroup are key ingredients for implementing pairing-based cryptosystems. We review these pairing-friendly elliptic curves in this section.

Let E be an elliptic curve over some finite field $\mathbb{F}_q$. We construct $\mathbb{G}$ and $\hat{\mathbb{G}}$ as two linearly independent subgroups of order p in the groups of points on E with coordinates

in $\mathbb{F}_q$ or in an extension field $\mathbb{F}_{q^k}$. The *embedding degree* k is the smallest extension degree that causes $\mathbb{F}_{q^k}$ to contain a multiplicative subgroup of order p. The target group $\mathbb{G}_T$ will be the multiplicative subgroup of order p in the extension field $\mathbb{F}_{q^k}$. Since $\mathbb{F}_{q^k}$ is a field, it follows that $\mathbb{G}_T$ will consist of the q-th roots of unity in $\mathbb{F}_{q^k}$. In general, since $\mathbb{G} \prec E(\mathbb{F}_q)$, $\hat{\mathbb{G}} \prec E(\mathbb{F}_{q^k})$ and $\mathbb{G}_T \prec \mathbb{F}_{q^k}$, the elements of $\mathbb{G}$ will have much shorter representation than those of $\hat{\mathbb{G}}$ and $\mathbb{G}_T$.

We want to match the size of parameters in pairings with the standard sizes of keys for symmetric encryption, for example the Advanced Encryption Standard (AES). When we say the 80-bit security level, we refer to the symmetric encryption with 80-bit keys. There are two factors that we have to consider when we choose the size for the various group elements on different types of elliptic curves.

1. At security level σ, the curve order must have a large prime factor $p > 2^{2\sigma-1}$, such that we can construct cyclic groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order p that is large enough to defeat generic discrete-logarithm attacks. The best known discrete logarithm algorithm on elliptic curves is the parallelised Pollard rho algorithm, which has running time $O(\sqrt{p})$. For example, if $\sigma = 80$, we need $p \geq 2^{160}$.

2. The curve must be constructed over a finite field $\mathbb{F}_q$ that is large enough to defeat the best known discrete-logarithm attacks in the extension field $\mathbb{F}_{q^k}$. The best algorithm for discrete logarithm computation in finite fields is the index calculus attack which has running time subexponential in the field size. At security levels $\sigma = 80$ and $\sigma = 128$, we respectively need $q^k \approx 2^{1024}$ and $q^k \approx 2^{3072}$. For the embedding degree $k = 6$, we have $q \approx 2^{171}$ and $q \approx 2^{512}$ respectively.

Freeman *et al.* [83] gave a classification of the known methods for constructing pairing-friendly elliptic curves. They are classified based on the method to construct the curves.

- Individual curves: given integers q and p, there is an elliptic curve E over $\mathbb{F}_q$ with a subgroup of order p and embedding degree k with respect to p. Individual curves include:
  - Supersingular curves [193, Theorem V.3.1].
  - Cocks-Pinch curves [68].
  - Dupont-Enge-Morain curves [75].
- Families of curves: given polynomials $q(x)$ and $p(x)$, if $q(x_0)$ is a prime power for some value of x_0, there is an elliptic curve E over $\mathbb{F}_{q(x_0)}$ with a subgroup of

order $p(x_0)$ and embedding degree k with respect to $p(x_0)$. Family of curves are further classified into:

- Sparse families: they give most of the known constructions of curves of prime order, but are currently limited to embedding degrees $k \leq 10$. Sparse families include:
  - Miyaji-Nakabayashi-Takano (MNT) families [155].
  - Galbraith-McKee-Valença families [86].
  - Freeman families [82].
- Complete families: constructions exist for arbitrary k but usually lead to slower arithmetic on the elliptic curve. Complete families include:
  - Cyclotomic families [12, 48].
  - Sporadic families [13, 125].
  - Scott-Barreto families [187].

Curves and Performance

We review the rough estimation on the performance issue of pairings in [44]. We consider three combinations of curve types and security parameters: Supersingular (SS) curves of embedding degree 2 over large prime fields at security level 80, and (non-supersingular) MNT families of embedding degree 6 at security level 80 and 128.

- SS curves can be used to construct type 1 pairings. They are easy to sample and simple to implement. However, they are inefficient at large security level. SS curves with embedding degree 2 were firstly used in the Boneh-Franklin identity-based encryption system [34].
- MNT families can be used to construct type 2 or type 3 pairings. Here we consider the MNT families with the embedding degree 6. These families were first proposed by Miyaji *et al.* [155].

Table 2.1 shows the representation size for the various group elements for the SS curves and the MNT families with different security levels. These numbers are derived from the intrinsic security requirements of the curves, and do not take into account of the security losses that might arise in actual cryptographic schemes or the underlying intractability assumptions.

Table 2.1: Comparison of representation sizes

	Representation sizes (bits)		
	SS @ 80-bit security	MNT @ 80-bit security	MNT @ 128-bit security
$\mathbb{Z}_p$	160	160	256
$\mathbb{G}$	512	171	512
$\hat{\mathbb{G}}$	512	1026	3072
$\mathbb{G}_T$	1024	1026	3072

Table 2.2: Comparison of estimated calculation times on the same elliptic curves at the same security levels as in Table 2.1. The time unit is defined as the cost of a general exponentiation on a random 171-bit elliptic curve for a random 160-bit exponent. Timings are indicative only.

	Relative timings (arbitrary unit)		
	SS @ 80-bit	MNT @ 80-bit	MNT @ 128-bit
In $\mathbb{G}$:			
Fix-base expon.	2	0.2	3
General expon.	10	1	15
In $\hat{\mathbb{G}}$:			
Fix-base expon.	2	8	100
General expon.	10	40	500
Hashing	10	40	500
In $\mathbb{G}_T$:			
Fix-base expon.	2	8	100
General expon.	10	10	500
Single pairing	100	100	1500
Ratio of pairings	120	120	1800

Table 2.2 gives rough estimations of the relative costs of various algebraic operations for the SS curves and the MNT families with different security levels. The numbers are set on the same arbitrary scale, and are used as a first approximation only. In practice, actual running times will depend on many factors, such as the choice of pairings, CPU types, space/time trade-offs, etc.

Pairings of Composite Order

Several recently proposed protocols require pairings having small embedding degree k with respect to a composite number n' that is presumed to be infeasible to factor, such as an RSA modulus. Boneh *et al.* [36] firstly proposed the use of bilinear groups whose order is a product of two primes. Let $\mathbb{G}', \hat{\mathbb{G}}', \mathbb{G}'_T$ be multiplicative groups of order $n' = p'q'$, where p' and q' are prime. We can define a pairing $\hat{e}' : \mathbb{G}' \times \hat{\mathbb{G}}' \to \mathbb{G}'_T$ similar to the pairing of prime order.

For pairing-based cryptosystems using elliptic curves of composite order to be secure, three problems must be infeasible:

- the discrete logarithm on the elliptic curve $E(\mathbb{F}_q)$,
- the discrete logarithm in the finite field $\mathbb{F}_{q^k}$, and
- the factorisation of the curve order n'.

The first two problems also appear to the pairings of prime order. However, for the pairings of composite order, the size of the elliptic curve group will be determined by the security level desired for the factoring problem. Factoring a large composite number n' takes roughly the same amount of time as solving the discrete logarithm in a finite field of size around n', since both algorithms use the Number Field Sieve. Therefore, we have to choose $n' \approx q^k$.

In [83], Freeman, Scott and Teske reviewed that pairings with composite orders can be constructed from supersingular curves using the method in [36] or from ordinary pairing-friendly elliptic curves using the Cocks-Pinch method. They concluded that supersingular curves with $k = 2$ are optimal for protocols requiring composite-order subgroups, as these curves can take advantage of the computational speedups.

In 2008, Katz *et al.* [127] proposed the use of bilinear groups whose order is a product of three primes. These bilinear groups can be used to construct more complicated cryptosystems. Recently, Freeman [84] proposed a framework to convert pairing-based cryptosystems from composite order groups to prime order groups.

2.2 Complexity Theory

2.2.1 Order Notation

The following is useful when describing the asymptotic behaviors of functions.

Definition 8 (Order Notation [148])**.** *For any function $f(n), g(n)$, we say that:*

- $f(n) = O(g(n))$ *if there exists a positive constant c and a positive integer n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.*
- $f(n) = \Omega(g(n))$ *if there exists a positive constant c and a positive integer n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.*

- $f(n) = \Theta(g(n))$ *if there exists positive constant* c_1 *and* c_2*, and a positive integer* n_0 *such that* $c_1 g(n) \leq f(n) \leq c_2 g(n)$ *for all* $n \geq n_0$*.*
- $f(n) = o(g(n))$ *if for any positive constant* $c > 0$ *there exists a constant* $n_0 > 0$ *such that* $0 \leq f(n) < cg(n)$ *for all* $n \geq n_0$*.*

Definition 9 (Negligibility [97])**.** *We call a function* $\mu : \mathbb{N} \to \mathbb{R}$ *negligible if for every positive polynomial* $p(\cdot)$ *there exists an* N *such that for all* $n > N$*,* $\mu(n) < 1/p(n)$*. A function is non-negligible if it is not negligible.*

Sometimes we say a probability is *overwhelming* to mean that it is negligibly less than 1.

2.2.2 Algorithms and Protocols

We model algorithms using Turing machines.

Definition 10 (Turing machine [194])**.** *A Turing machine is a 7-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *where*

1. Q *is a finite set called* states*,*
2. Σ *is the input* alphabet *not containing the special blank symbol* $\sqcup$*,*
3. Γ *is the* tape alphabet*, where* $\sqcup \in \Gamma$ *and* $\Sigma \subseteq \Gamma$*,*
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ *is the* transition function*,*
5. $q_0 \in Q$ *is the* start state*, and*
6. $q_{accept} \subseteq Q$ *is the* accept state*.*
7. $q_{reject} \subseteq Q$ *is the* reject state*, where* $q_{accept} \neq q_{reject}$*.*

A deterministic Turing machine is a Turing machine having an infinite read-write tape and the state transitions are completely determined by the input. In a probabilistic Turing machine, the state transitions are determined by the input and the output of coin tosses.

Definition 11 (Algorithm [194])**.** *A deterministic (resp. probabilistic) algorithm is a deterministic (resp. probabilistic) Turing machine.*

Given x, the output $A(x)$ of a probabilistic algorithm A is a random variable induced by the coin tosses. Let $A(x) = y$ denote the event "A outputs y on input x". By $Pr[A(x) = y]$, we mean the probability of this event. By $A(\cdot)$ we denote that the algorithm A has one input. By $A(\cdot, \ldots, \cdot)$ we denote that A has several inputs. $y \leftarrow A(x)$ denotes that y is obtained by running algorithm A on input x. In case A is deterministic, then this y is unique. If A is probabilistic (in which case we sometimes write $y \xleftarrow{R} A(x)$, then y is a random variable. If S is a set, then $y \leftarrow S$ (or sometimes $y \xleftarrow{R} S$) denotes that y was chosen from S uniformly at random.

Definition 12 (Efficient Algorithm [148])**.** *An efficient algorithm or a polynomial time algorithm is an algorithm whose worst-case running time function is of the form $O(n^k)$, where n is the input size and k is a constant.*

We use the shorthand notation "PPT" for "probabilistic polynomial-time" when describing an algorithm. Next, we define what a two-party protocol is.

Definition 13 (Two-Party Protocol [72])**.** *A two-party protocol is a pair of interactive probabilistic Turing machines $(\mathcal{P}, \mathcal{V})$. An execution (or run) of the protocol $(\mathcal{P}, \mathcal{V})$ on input x (for $\mathcal{P}$) and y (for $\mathcal{V}$) is an alternating sequence of $\mathcal{P}$-rounds and $\mathcal{V}$-rounds, each producing a message to be delivered to the other party (except for the last $\mathcal{V}$-round). The sequence of such message is called the transcript of this run of the protocol.*

If, for all x and y, the length of such sequence, as well as the expected running time of $\mathcal{P}$ and $\mathcal{V}$, are polynomial in the length of x and y, then ($\mathcal{P}$, $\mathcal{V}$) is an *efficient* two-party protocol. By "$(\mathcal{P}(x) \leftrightarrow \mathcal{V}(y))$", we denote the probability space that assigns to a sequence of strings π the probability that a run of the ($\mathcal{P}$, $\mathcal{V}$) protocol, on input x and y, will produce π as transcript.

2.2.3 Relations and Languages

If A is the set of all strings that a machine M accepts, we say that A is the *language* of the machine M and write L(M) = A.

A **verifier** for a language A is an algorithm V, where

$$A = \{w | V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}.$$

Definition 14 (Polynomial Reducible [145])**.** *We say that a language L is polynomially reducible to another language L_0 if there exists a deterministic polynomial-time-bounded Turing machine M which will convert each instance $I \in L$ into an instance $I_0 \in L_0$, such that $I \in L$ if and only if $I_0 \in L$.*

Definition 15 (Class P, NP, NPC [194]). **P** *is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine.* **NP** *is the class of languages that have polynomial time verifiers. A language B is* **NP***-complete if B is in* **NP***, and every A in* **NP** *is polynomially reducible to B. The class of all* **NP***-complete problems is denoted by* **NPC***.*

2.2.4 Intractability

Public key cryptography is based on the intractability of solving some problems. These problems have no known polynomial-time solution. In this section, we introduce the intractability assumption used in this thesis. For simplicity, denote "XYZ" as the name of the problems introduced in this section. We say that the (ϵ, t)-XYZ assumption holds if no t-time algorithm has the probability ϵ in solving the XYZ problem. For the "decisional XYZ" problems, we say that the (ϵ, t)-decisional XYZ assumption holds if no t-time algorithm has the probability ϵ plus a half in solving the decisional XYZ problem.

The two most fundamental problems are the discrete logarithm problem and the factorisation problem.

Discrete Logarithm (DL) Problem. The DL problem is that, given g, $y \in \mathbb{G}$, to output $x = \log_g y$.

Factorisation Problem. The Factorisation problem is that, given a product of two primes N, to output a non-trivial factor of N.

The best-known algorithm for factoring a number n is the general number field sieve, which works in sub-exponential time: $O(e^{(\log n)^{1/3}(\log\log n)^{2/3}})$. The Pollard's rho algorithm for solving the DL problem for a group of order n works in time $O(\sqrt{n})$[1].

Two of the most fundamental public key cryptosystems are the Diffie-Hellman protocol [71] and the RSA system [180]. Their security are based on the computational Diffie-Hellman assumption and the RSA assumption respectively.

Computational Diffie-Hellman (CDH) Problem. The CDH problem is that, given $g, g^a, g^b \in \mathbb{G}$ for unknown $a, b \in \mathbb{Z}_p^*$, to output g^{ab}.

RSA Problem. The RSA problem is that, given a product of two primes N, a positive integer e and an element a chosen uniformly at random from $\mathbb{Z}_N^*$, to output $x \in \mathbb{Z}_N^*$ such that $x^e \equiv a \mod N$.

[1]There are known efficient algorithms for solving the factorisation problem and the DL problem on quantum computers, such as the Shor's algorithm [191].

Maurer and Wolf [146] demonstrated that under an unproven but plausible assumption, the CDH problem and the DL problem are polynomial-time equivalent in $\mathbb{G}$. They also proved that the CDH problem and the DL problem are equivalent in a uniform sense for groups whose orders belong to certain classes.

Aggarwal and Maurer [2] introduced a class of algorithms called *generic ring algorithm*. They showed that the factorisation assumption is equivalent to the assumption that there exists no probabilistic polynomial-time generic ring algorithm solving the RSA problem.

The decisional version of the CDH problem is called the decisional Diffie-Hellman (DDH) problem. However, the DDH problem is easy in symmetric pairings. In asymmetric pairings, there are variants of the DDH assumption [31].

Decisional Diffie-Hellman (DDH) Problem. The DDH problem is that, given $g, g^a, g^b, g^c \in \mathbb{G}$ for unknown $a, b, c \in \mathbb{Z}_p^*$, to decide if $ab = c$.

External Diffie-Hellman (XDH) Problem. The XDH problem is equivalent to the DDH problem in $\mathbb{G}_1$ for type 2 or type 3 pairings.

Symmetric External Diffie-Hellman (SXDH) Problem. The SXDH problem is equivalent to the DDH problem in $\mathbb{G}_1$ and the DDH problem in $\mathbb{G}_2$ for type 3 pairings.

We then introduce some intractability problems in the cyclic groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, equipped with a pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We only define the problems in type 1 pairings here. The problems in type 2 or type 3 pairings can be defined similarly.

Bilinear Diffie-Hellman (BDH) Problem [34]. The BDH problem is that, given $g, g^a, g^b, g^c \in \mathbb{G}$ for unknown $a, b, c \in \mathbb{Z}_p^*$, to output $\hat{e}(g, g)^{abc}$.

Decisional Bilinear Diffie-Hellman (DBDH) Problem [34]. The DBDH problem is that, given $g, g^a, g^b, g^c \in \mathbb{G}$ and $T \in \mathbb{G}_T$ for unknown $a, b, c \in \mathbb{Z}_p^*$, to output 1 if $T = \hat{e}(g, g)^{abc}$, and to output 0, otherwise.

Decision Linear Problem [31]. The Decision Linear problem is that, given u, u^a, v, v^b, h, $h^c \in \mathbb{G}$ for unknown $a, b, c \in \mathbb{Z}_p^*$, to output 1 if $c = a + b$, and to output 0, otherwise.

Diffie-Hellman Inversion (DHI) Problem [154]. The DHI problem is that, given $g, g^x \in \mathbb{G}$ for unknown $x \in \mathbb{Z}_p^*$, to output $g^{1/x}$.

Decisional Bilinear Diffie-Hellman Inversion (BDHI) Problem [154]. The Decisional BDHI problem states: given $g, g^x \in \mathbb{G}$, and $T \in \mathbb{G}_T$ for unknown $x \in \mathbb{Z}_p^*$, to output 1 if $T = \hat{e}(g, g)^{1/x}$, and to output 0, otherwise.

q-Strong Diffie-Hellman (SDH) Problem [29]. The q-SDH is that, given g, g^{α}, ..., $g^{\alpha^q} \in \mathbb{G}$ for unknown $\alpha \in \mathbb{Z}_p^*$, to output a pair $(g^{\frac{1}{\alpha+c}}, c)$ where $c \in \mathbb{Z}_p^*$.

Decisional weak Bilinear Diffie-Hellman Inversion (wBDHI) Problem [30]. The ℓ-decisional wBDHI problem is that, given g, h, $g^{\alpha}, \ldots, g^{\alpha^{\ell}} \in \mathbb{G}$ and $T \in \mathbb{G}_T$ for unknown $\alpha \in \mathbb{Z}_p^*$, to output 1 if $T = \hat{e}(g,h)^{1/\alpha}$ and to output 0 otherwise.

Decisional q-Bilinear Diffie-Hellman Exponent (BDHE) Problem [30]. The decisional q-BDHE problem is that, given $(g,\ g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, g^s) \in \mathbb{G}$ and $T \in \mathbb{G}_T$ for unknown $a, s \in \mathbb{Z}_p$, to output 1 if $T = \hat{e}(g,g)^{a^{q+1}s}$ and to output 0, otherwise.

We also consider a hard problem in the bilinear groups with a composite order. Boneh *et al.* [36] proposed the use of bilinear groups $\mathbb{G}$ of order $N = pq$, where p and q are independent uniform random primes. Denote g as the generator of $\mathbb{G}$. Let $\mathbb{G}_q \subset \mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q. The subgroup decision problem is as follows:

Subgroup Decision Problem [36]. The subgroup decision problem is that, given $(N = pq, \mathbb{G})$ and an element w selected at random either from $\mathbb{G}$ or from $\mathbb{G}_q$, to output 1 if $w \in \mathbb{G}_q$ and to output 0, otherwise.

2.3 Information Theory

Information theory is a quantification of information. Information theory is founded in 1948 by Claude Shannon [190]. Shannon established that the number of bits needed to represent the result of an uncertain event is given by its *entropy*.

2.3.1 Entropy

The entropy, H, of a discrete random variable X is a measure of the amount of uncertainty associated with the value of X. If $\mathbb{X}$ is the set of all messages x that X could be, and $p(x)$ is the probability of X given x, then the entropy of X is defined:

$$H(X) = -\sum_{x \in \mathbb{X}} p(x) \log p(x).$$

2.3.2 Extracting Randomness

We review the concepts related to probability distributions and extracting uniform bits from weak random sources. The statistical distance between two random variables X

and Y having the same domain $\mathcal{D}$ is $\Delta(X,Y) = \frac{1}{2}\sum_{v\in\mathcal{D}} |\Pr[X=v] - \Pr[Y=v]|$. We say that two variables are ϵ-close if their statistical distance is at most ϵ.

The min-entropy of a random variable X is

$$H_\infty(X) = -\lg(\max_x \Pr[X=x]).$$

Consider another variable Y which is correlated to X. The average min-entropy [74] is

$$\tilde{H}_\infty(X|Y) := -\lg\left(\mathop{E}_{y\leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right]\right).$$

We will use the following lemmas on average min-entropy proved in [74]:

Lemma 1. *If Y has 2^r possible values and Z is any random variable, then*

$$\tilde{H}_\infty(X|(Y,Z)) \geq \tilde{H}_\infty(X|Z) - r.$$

Lemma 2. *Let X, Y be random variables such that $X \in \{0,1\}^n$ and $\tilde{H}_\infty(X|Y) \geq k$. Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$. Then for $h \leftarrow H$, we have $\Delta((Y,\ h,\ h(X)),\ (Y,\ h,\ U_\ell)) \leq \epsilon$, as long as $\ell \leq k - 2\lg(1/\epsilon)$.*

Chapter 3

Backgrounds

In this Chapter, we survey the literature on works related to our thesis. They serve as an elaborated tutorial on various security notions, models and current state-of-art constructions. We hope that after reading this Chapter, the readers can better understand the topics that will appear in later chapters, and at the same time better evaluate the contribution of this thesis.

3.1 Cryptographic Primitives

Cryptographic primitives are well-established, low-level cryptographic algorithms that are frequently used to build cryptographic protocols. Researchers use cryptographic primitives as the basic building blocks of cryptographic protocols. We introduce several cryptographic primitives which will be used in later Chapters.

3.1.1 Trapdoor Functions

We first review the definition of a collection of trapdoor functions (TDFs). For generality, let $n = poly(\lambda)$ denote the input length of the TDFs in terms of the security parameter. A collection of TDFs is given by a tuple of polynomial-time algorithms (S, F, F^{-1}) with the following properties:

1. *Easy to sample, compute, and invert with trapdoor*: $S(1^\lambda)$ outputs (s, t) where s is a function index and t is its trapdoor, $F(s, \cdot)$ computes an injective function $f_s(\cdot)$ over the domain $\{0,1\}^n$, and $F^{-1}(t, \cdot)$ computes $f_s^{-1}(\cdot)$.

2. *Hard to invert without trapdoor*: for any PPT inverter $\mathcal{I}$, the probability that $\mathcal{I}(1^\lambda, s, f_s(x))$ outputs x is at most negligible, where the probability is taken over the choice of $(s, t) \leftarrow S(1^\lambda)$, $x \leftarrow \{0,1\}^n$, and $\mathcal{I}$'s randomness.

Lossy Trapdoor Functions

We briefly review the definition of a collection of lossy trapdoor functions in [174]. We retain the same notation for n, λ as above. Denote $k \leq n$ as the lossiness of the collection. A collection of (n, k)-lossy TDFs is given by a tuple of polynomial-time algorithms $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ having the following properties: For notational convenience, define the algorithms $S_{\text{inj}}(\cdot) := S_{\text{ltdf}}(\cdot, 1)$ and $S_{\text{loss}}(\cdot) := S_{\text{ltdf}}(\cdot, 0)$.

1. *Easy to sample an injective function with trapdoor*: $S_{\text{inj}}(1^\lambda)$ outputs (s, t) where s is a function index and t is its trapdoor, $F_{\text{ltdf}}(s, \cdot)$ computes an injective function $f_s(\cdot)$ over the domain $\{0,1\}^n$, and $F_{\text{ltdf}}^{-1}(t, \cdot)$ computes $f_s^{-1}(\cdot)$. If $f_s^{-1}(y)$ does not exist, then the behavior of $F_{\text{ltdf}}^{-1}(t, y)$ is unspecified.

2. *Easy to sample a lossy function*: $S_{\text{loss}}(1^\lambda)$ outputs $(s, \perp)$ where s is a function index, and $F_{\text{ltdf}}(s, \cdot)$ computes a (deterministic) function $f_s(\cdot)$ over the domain $\{0,1\}^n$ whose image has size at most 2^{n-k}.

3. *Hard to distinguish injective from lossy*: the first outputs of $S_{\text{inj}}(1^\lambda)$ and $S_{\text{loss}}(1^\lambda)$ are computationally indistinguishable.

All-But-One Trapdoor Functions

We briefly review the definition of a collection of all-but-one (ABO) trapdoor functions in [174]. In an ABO collection, each function has several branches. Let $\mathcal{B} = \{B_\lambda\}_{\lambda \in \mathbb{N}}$ be a collection of sets whose elements represent the branches. Then a collection of (n, k)-ABO TDFs with branch collection $\mathcal{B}$ is given by a tuple of polynomial-time algorithms $(S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ having the following properties:

1. *Easy to sample a trapdoor function with given lossy branch*: for any $b^* \in B_\lambda$, $S_{\text{abo}}(1^\lambda, b^*)$ outputs (s, t), where s is a function index and t is its trapdoor. For any $b \in B_\lambda$ distinct from b^*, $G_{\text{abo}}(s, b, \cdot)$ computes an injective function $g_{s,b}(\cdot)$ over the domain $\{0,1\}^n$, and $G_{\text{abo}}^{-1}(t, b, \cdot)$ computes $g_{s,b}^{-1}(\cdot)$. As above, the behavior of $G_{\text{abo}}^{-1}(t, b, y)$ is unspecified if $g_{s,b}^{-1}(y)$ does not exist. Additionally, $G_{\text{abo}}(s, b^*, \cdot)$ computes a function $g_{s,b^*}(\cdot)$ over the domain $\{0,1\}^n$ whose image has size at most 2^{n-k}.

2. *The lossy branch is hidden*: for any $b_0^*, b_1^* \in B_\lambda$, the first output s_0 of $S_{\text{abo}}(1^\lambda, b_0^*)$ and the first output s_1 of $S_{\text{abo}}(1^\lambda, b_1^*)$ are computationally indistinguishable.

3.1.2 Hash Functions

A hash function is an efficiently computable function mapping binary strings of arbitrary finite length to binary strings of a fixed length. We consider the hash function family

$$H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y},$$

where $\mathcal{K}$ and $\mathcal{Y}$ are finite nonempty sets, and $\mathcal{M}$ and $\mathcal{Y}$ are sets of strings. Often we will write the first argument to H as a subscript, so that $H_K(M) = H(K, M)$ for all $M \in \mathcal{M}$. We write $M \xleftarrow{\$} S$ for the experiment of choosing a random element from the distribution S and calling it M.

We review the definition from Rogaway and Shrimpton [181] for seven notions of hash-function security. The definitions fall under the general categories of *preimage-resistance*, *second-preimage resistance*, and *collision-resistance*.

Preimage-resistance

Preimage-resistance means that for all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output.

Definition 16 (Types of preimage resistance)**.** *Let $H = \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash-function family and let m be a number such that $\{0,1\}^m \leftarrow \mathcal{M}$. Let A be an adversary. Then define:*

$$\mathbf{Adv}_H^{\mathrm{Pre}[m]}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^m; Y \leftarrow H_K(M); M' \xleftarrow{\$} A(K,Y) : H_K(M') = Y],$$
$$\mathbf{Adv}_H^{\mathrm{ePre}}(A) = \max_{Y \in \mathcal{Y}} \left\{ \Pr[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} A(K) : H_K(M) = Y] \right\},$$
$$\mathbf{Adv}_H^{\mathrm{aPre}[m]}(A) = \max_{K \in \mathcal{K}} \left\{ \Pr[M \xleftarrow{\$} \{0,1\}^m; Y \leftarrow H_K(M); M' \xleftarrow{\$} A(K) : H_K(M') = Y] \right\}.$$

The preimage resistance (Pre) is the usual way to define when a hash-function family is a one-way function.

Second-preimage resistance

Second-preimage resistance means that it is computationally infeasible to find any second input which has the same output as any specified input.

Definition 17 (Types of second-preimage resistance)**.** *Let $H = \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash-function family and let m be a number such that $\{0,1\}^m \leftarrow \mathcal{M}$. Let A be an*

adversary. Then define:

$$\mathbf{Adv}_H^{\mathrm{Sec}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^m; M' \xleftarrow{\$} A(K,M) : (M \neq M') \wedge H_K(M) = H_K(M')],$$
$$\mathbf{Adv}_H^{\mathrm{eSec}[m]}(A) = \max_{M \in \{0,1\}^m} \left\{ \Pr[K \xleftarrow{\$} \mathcal{K}; M' \xleftarrow{\$} A(K) : (M \neq M') \wedge H_K(M) = H_K(M')] \right\},$$
$$\mathbf{Adv}_H^{\mathrm{aSec}[m]}(A) = \max_{K \in \mathcal{K}} \left\{ \Pr[M \xleftarrow{\$} \{0,1\}^m; M' \xleftarrow{\$} A(M) : (M \neq M') \wedge H_K(M) = H_K(M')] \right\}.$$

The second-preimage resistance (Sec) is also called the *weak collision-resistance* [150]. The second definition, everywhere second-preimage resistance (eSec), is also called a *universal one-way hash function family* (UOWHF) and it was first defined by Naor and Yung [160].

Collision resistance

Collision resistance means that it is computationally infeasible to find any two distinct inputs which hash to the same output.

Definition 18 (Collision resistance)**.** *Let $H = \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash-function family and let A be an adversary. Then define:*

$$\mathbf{Adv}_H^{\mathrm{Coll}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}; (M, M') \xleftarrow{\$} A(K) : (M \neq M') \wedge H_K(M) = H_K(M')].$$

The collision resistance (Coll) defined above is also called the *strong collision-resistance* [150].

3.1.3 Random Oracle Model

The Random Oracle Model (ROM) is a paradigm that acts as a bridge between cryptographic theory and cryptographic practice. The idea of ROM was firstly formulated by Bellare and Rogaway [21].

Random oracle is a powerful and imaginary function that on any input, the distribution of the output hashed value is uniform in the function's output space. It has three properties: deterministic, efficient and uniform output. In the ROM, we assume hash functions are random functions and are publicly accessible by all parties. Random oracle, H, is an object to instantiate all hash functions in the model and reply all queries from the parties. A polynomial time algorithm cannot distinguish the query replied from a real world or the random oracle simulated by a function.

The properties of determinism and uniform output mean that the output of a random oracle has an entropy greater than that of its input. However by Shannon's entropy

theory, a deterministic function can never amplify its entropy. Therefore random oracle does not exist in the real world.

A hash function in the real world has its output values following some probability distribution which may not be discernible by a polynomially bounded distinguisher. Thus a real-world hash function only emulated behavior of a random oracle behavior to a precision where the difference is hopefully a negligible quantity. Despite of its impractical assumptions, the paradigm is useful to yield efficient solutions to the security of cryptographic protocols. It is better than no proof shown.

Many signature or encryption schemes used rewinding of hashings and (or) observing hashing input and output in their reductionist security proofs, like the Schnorr signature [186]. However the result of Barak *et al.* [9, 10] and Goldwasser and Kalai [98] proved the insecurity of the random oracle model as it is used in the Fiat-Shamir paradigm [78]. Several papers proved that some popular cryptosystems previously proved secure in the random oracle were actually provably insecure when the random oracle was instantiated by any real-world hashing functions [52, 16]. As a result, recently there are many new signature schemes which try to prove their security without random oracles.

3.1.4 Pseudorandom Functions

A pseudorandom function (PRF) is a family of functions with the property that the input-output behavior of a random instance of the family is *computationally indistinguishable* from that of a random function.

A function family is a map $F : \mathcal{K} \times D \to R$, where $\mathcal{K}$ is the set of keys of F, D is the domain of F and R is the range of F. For any key $K \in \mathcal{K}$ we define the map $F_K : D \to R$ by $F_K(x) = F(K, x)$. We define two games as follows:

- Game Real_F picks a random instance F_K of family F and then runs an adversary A with oracle F_K. The adversary A interacts with its oracle, querying it with input x and getting back answers $F_K(x)$, and eventually outputs a "guess" bit. The game returns the same bit.

- In game Rand_R, the adversary A interacts with its oracle, querying it with input x and getting back answers randomly chosen from R, and eventually outputs a "guess" bit. The game returns the same bit.

The adversary A outputs a "guess" bit "1" when A thinks that it is in the game Real_F; or outputs a "guess" bit "0" when A thinks that it is in the game Rand_R.

The *prf-advantage* of A is defined as

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) = |\Pr[\mathsf{Real}_F^A = 1] - \Pr[\mathsf{Rand}_R^A = 1]|.$$

Here the probability is taken over all the possible choices of F_K, the number chosen from R, and the internal coin tosses of A. A pseudorandom function is a family of functions such that the following two conditions holds:

- Efficient Evaluation: There exists a deterministic polynomial-time algorithm that on input K and $x \in D$ returns $F_K(x)$.
- Pseudorandomness: There is no probabilistic polynomial-time algorithm A that has non-negligible prf-advantage.

3.2 Public Key Cryptography

Public key cryptography is one of the key area in cryptography and it is widely used around the world. Each user has a pair of public key and private key (secret key), which can be generated by himself. Unlike symmetric key cryptography, it does not require a secure initial exchange of secret keys to both sender and receiver. We introduce the definitions of public key encryption and signatures.

3.2.1 Public Key Encryption

A public key encryption scheme is a tuple $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ such that:

- $(ke, kd) \leftarrow \mathcal{G}(1^\lambda)$: On input a security parameter 1^λ, the key generation algorithm outputs an encryption key ke and a decryption key kd. It also defines a message space $\mathcal{M}$.
- $c \leftarrow \mathcal{E}(m, ke)$: On input a message $m \in \mathcal{M}$ and an encryption key ke, the encryption algorithm $\mathcal{E}$ outputs a ciphertext c.
- $m/\perp \leftarrow \mathcal{D}(c, kd)$: On input a ciphertext c and a decryption key kd, the decryption algorithm $\mathcal{D}$ outputs a message m or $\perp$ for invalid ciphertext.

Correctness: We require that:

$$\mathcal{D}(\mathcal{E}(m, ke), kd) = m.$$

The security goal of an encryption is defined as the indistinguishability of ciphertexts under adaptive chosen ciphertext attack (IND-CCA2). It is defined as the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. $\mathcal{C}$ runs $(ke, kd) \leftarrow \mathcal{G}(1^\lambda)$ and gives the encryption key ke to $\mathcal{A}$.
2. $\mathcal{A}$ can make decryption queries to a decryption oracle $\mathcal{DO}$: On input a ciphertext c, the oracle outputs a message $m/\perp \leftarrow \mathcal{D}(c, kd)$.
3. $\mathcal{A}$ randomly picks two messages $m_0, m_1 \in \mathcal{M}$ and gives them to $\mathcal{C}$. $\mathcal{C}$ flips a coin b, computes $c^* \leftarrow \mathcal{E}(m_b, ke)$ and returns c^* to $\mathcal{A}$.
4. $\mathcal{A}$ can make decryption queries to the decryption oracle $\mathcal{DO}$ for any ciphertext, except c^*.
5. $\mathcal{A}$ outputs a bit b'.

$\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ $Adv_\mathcal{A}$ is $\mathcal{A}$'s probability of winning minus half.

Definition 19 (IND-CCA2)**.** *An encryption scheme is indistinguishable against adaptive chosen ciphertext attack if $Adv_\mathcal{A}$ is negligible.*

There exist some other weaker security models in which the attacker is given less power. If step 4 is removed from the above game definition, then the security becomes "indistinguishable against chosen ciphertext attack (IND-CCA1)". If both step 2 and 4 are removed from the above game definition (i.e. no decryption oracle queries), then the security becomes "indistinguishable against chosen plaintext attack (IND-CPA)".

3.2.2 Digital Signatures

A digital signature scheme is a tuple $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$ such that:

- $(sk, vk) \leftarrow \mathbf{Gen}(1^\lambda)$: On input a security parameter 1^λ, the key generation algorithm outputs a secret key sk and a public key vk. It also defines a message space $\mathcal{M}$.
- $s \leftarrow \mathbf{Sign}(m, sk)$: On input a message $m \in \mathcal{M}$ and a secret key sk, the signing algorithm $\mathcal{S}$ outputs a signature s.
- $\top/\perp \leftarrow \mathbf{Verify}(s, m, vk)$: On input a signature s, a message m and a public key vk, the verification algorithm **Verify** outputs $\top$ for valid or $\perp$ for invalid.

Correctness: We require that:

$$\mathbf{Verify}(s, m, vk) = \begin{cases} \top \text{ with probability 1} & \text{if } s \leftarrow \mathbf{Sign}(m, sk); \\ \bot \text{ with an overwhelming probability} & \text{otherwise.} \end{cases}$$

The existential unforgeability against adaptive chosen message attack of a signature scheme is defined as the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. $\mathcal{C}$ gives a public key vk to $\mathcal{A}$, where $(sk, vk) \leftarrow \mathbf{Gen}(1^\lambda)$.
2. $\mathcal{A}$ can make signing queries to a signing oracle $\mathcal{SO}$: On input a message $m \in \mathcal{M}$, the oracle outputs a signature $s = \mathbf{Sign}(m, sk)$.
3. $\mathcal{A}$ outputs a signature s^* for a message $m^* \in \mathcal{M}$.

$\mathcal{A}$ wins the game if $\mathbf{Verify}(s^*, m^*, vk) = \top$ and s^* is not the output from $\mathcal{SO}$. The advantage of $\mathcal{A}$, $Adv_{\mathcal{A}}$, is $\mathcal{A}$'s probability of winning.

Definition 20 (UF-CMA)**.** *A signature scheme is existential unforgeable against adaptive chosen message attack if $Adv_{\mathcal{A}}$ is negligible.*

One-Time Signatures

One-time signatures are digital signature schemes that enable a signer to sign a single message with a given set of public and private key pair. One-time signatures require the signer to generate different key pairs for each message to be signed, in contrast with "regular" digital signatures like RSA, whose key pairs can be used for an unlimited number of times.

The best known one-time signature scheme was presented by Merkle in [149], and is an improvement of Lamport's [137] signature scheme. The security of the scheme is implied by the irreversibility of the one-way function and the collision-resistance of the hash function used.

Reset Lemma

A three-move *canonical* protocol is defined as follows: The prover's first message is called its *commitment*. The verifier selects a *challenge* uniformly at random from a set ChSet_v associated to its input v, and, upon receiving a *response* Rsp from the prover, applies a deterministic decision predicate DEC_v(Cmt,Ch,Rsp)) to compute a boolean decision. The verifier is said to be represented by the pair (ChSet, DEC) which, given the verifier input v, defines the challenge set and decision predicate.

A prover is identified with a function Q that given an incoming message M_{in} (this is ϵ when the prover is initiating the protocol) and its current state St, returns an outgoing message M_{out} and an updated state. The initial state of the prover is (q, R), where q is an input for the prover and R is a random tape.

Many identification and signature schemes can be viewed as a canonical protocol, such as the Schnorr signatures and the Guillou-Quisquater identification. Bellare and Palacio [19] proved the Reset Lemma, which upper bounds the probability that a (cheating) prover Q can convince the verifier to accept as a function of the probability that a certain experiment based on resetting the prover yields two accepting conversation transcripts.

Lemma 3 (Reset Lemma[19])**.** *Let Q be a prover in a canonical protocol with a verifier represented by (*ChSet, DEC*), and let q, v be inputs for the prover and verifier, respectively. Let $\mathsf{acc}(q, v)$ be the probability that the verifier accepts in its interaction with Q, namely the probability that the following experiment returns 1:*

Choose random tape R for Q; $St \leftarrow (q, R)$; $(\text{CMT}, St) \leftarrow Q(\epsilon; St)$
$\text{CH} \leftarrow^R \text{ChSet}_v$; $(\text{RSP}, St) \leftarrow Q(\text{CH}; St)$; $d \leftarrow \text{DEC}_v(\text{CMT},\text{CH},\text{RSP})$
Return d

Let $\mathsf{res}(q, v)$ be the probability that the following reset *experiment returns 1:*

Choose random tape R for Q; $St \leftarrow (q, R)$; $(\text{CMT}, St) \leftarrow Q(\epsilon; St)$
$\text{CH}_1 \leftarrow^R \text{ChSet}_v$; $(\text{RSP}_1, St_1) \leftarrow Q(\text{CH}; St)$; $d_1 \leftarrow \text{DEC}_v(\text{CMT},\text{CH}_1, \text{RSP}_1)$
$\text{CH}_2 \leftarrow^R \text{ChSet}_v$; $(\text{RSP}_2, St_2) \leftarrow Q(\text{CH}; St)$; $d_2 \leftarrow \text{DEC}_v(\text{CMT},\text{CH}_2, \text{RSP}_2)$
If ($d_1 = 1$ AND $d_2 = 1$ AND $\text{CH}_1 \neq \text{CH}_2$) then return 1 else return 0

Then

$$\mathsf{acc}(q, v) \leq \frac{1}{|\text{ChSet}_v|} + \sqrt{\mathsf{res}(q, v)}.$$

The Reset Lemma provides a good upper bound, has a simple proof, and is general enough to be applicable in the proof of numerous signature schemes.

The Waters Signature Scheme

Waters [206] presented a secure signature scheme based on CDH problem without random oracles. The scheme is summarised as follows:

1. **Gen**. Randomly choose $\alpha \in \mathbb{Z}_p$ and let $g_1 = g^\alpha$. Additionally, choose two random values $g_2, u' \in \mathbb{G}$ and a random n-length vector $\mathsf{U} = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. The public key is $vk = (g_1, g_2, u', \mathsf{U})$ and the secret key is g_2^α. The message space $\mathcal{M}$ is $\{0,1\}^n$.

2. **Sign**. To generate a signature on message $M = (\mu_1, \ldots, \mu_n) \in \{0,1\}^n$, pick $s \in_R \mathbb{Z}_p^*$ and output the signature as $\sigma = (g_2^\alpha \cdot (u' \prod_{j=1}^n u_j^{\mu_j})^s,\ g^s)$ with his secret key g_2^α.

3. **Verify**. Given a signature $\sigma = (\sigma_1, \sigma_2)$ on message $M = (\mu_1, \ldots, \mu_n) \in \{0,1\}^n$, it outputs 1 if $\hat{e}(g, \sigma_1) = \hat{e}(g_1,\ g_2) \cdot \hat{e}(u' \prod_{i=1}^n u_i^{\mu_i},\ \sigma_2)$. Otherwise, it outputs 0.

Signatures of Knowledge

In a traditional signature scheme, a signature σ on a message m is issued under a public key PK, and can be interpreted as follows: "The owner of the public key PK and its corresponding secret key has signed message m." Chase and Lysyanskaya [57] proposed the notion *signatures of knowledge*. They allow one to issue signatures on behalf of any NP statement, that can be interpreted as follows: "A person in possession of a witness w to the statement that $x \in L$ has signed message m."

Chase and Lysyanskaya [57] defined the algorithms in a signature of knowledge schemes as follows: Let {Mess} be a set of message spaces, and for any language $L \in$ NP, let M_L denoted a polynomial time Turing machine which accepts input (x, w) iff w is a witness showing that $x \in L$. Let **Gen** be an algorithm that outputs public parameters $p \in \{0,1\}^\lambda$ for some parameter λ. Let **Sign**(p, M_L, x, w, m) be an algorithm that takes as input some public parameters p, a Turing machine M_L for a language L in NP, a value $x \in L$, a valid witness w for x, and $m \in Mess$, a message to be signed. **Sign** outputs a signature of knowledge for instance $x \in L$ on the message m. Let **Verify**(p, M_L, x, m, σ) be an algorithm that takes as input the values p, M_L, x, the message m, and a purported signature σ, and either outputs $\top$ for accept or $\perp$ for reject. In their game-based definition, they define the SimExt-security as follows.

Definition 21 (SimExt-security [57])**.** *Let L be the language defined by a polynomial-time Turing machine M_L as explained above, such that all witnesses for $x \in L$ are of known polynomial length $p(|x|)$. Then* (**Gen**, **Sign**, **Verify**) *constitute a SimExt-secure signature of knowledge of a witness for L, for message space {Mess} if the following properties hold:*

- Correctness. *There exists a negligible function ν such that for all $x \in L$, valid witnesses w for x (i.e. witnesses w such that $M_L(x,w) = 1$), and $m \in \{Mess_k\}$ such that*

$$\Pr[p \leftarrow \mathbf{Gen}(1^\lambda); \sigma \leftarrow \mathbf{Sign}(p, M_L, x, w, m) : \mathbf{Verify}(p, M_L, x, m, \sigma) = \top] = 1 - \nu(k).$$

- Simulatability. *There exists a polynomial time simulator consisting of algorithms Sim_{gen} and Sim_{sign} such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function ν such that for all polynomials f, for all k, for all auxiliary input $s \in \{0,1\}^{f(k)}$,*

$$\left| \begin{array}{c} \Pr[(p,\tau) \leftarrow Sim_{gen}(1^\lambda); b \leftarrow \mathcal{A}^{Sim(p,\tau,\cdot,\cdot,\cdot)}(s,p) : b = 1] \\ -\Pr[p \leftarrow \mathbf{Gen}(1^\lambda); b \leftarrow \mathcal{A}^{\mathbf{Sign}(p,\cdot,\cdot,\cdot)}(s,p) : b = 1] \end{array} \right| = \nu(\lambda),$$

where the oracle Sim receives the values (M_L, x, w, m) as inputs, checks that the witness w given to it was correct and returns $\sigma \leftarrow Sim_{sign}(p, \tau, M_L, x, m)$. τ is the additional trapdoor value that the simulator needs in order to simulate the signatures without knowing a witness.

- Extraction. *In addition to (Sim_{gen}, Sim_{sign}), there exists an extractor algorithm* Extract *such that for all probabilistic polynomial time adversaries $\mathcal{A}$ there exists a negligible function ν such that for all polynomials f, for all k, for all auxiliary input $s \in \{0,1\}^{f(k)}$,*

$$\Pr\left[\begin{array}{c} (p,\tau) \leftarrow Sim_{gen}(1^\lambda); (M_L, x, m, \sigma) \leftarrow \mathcal{A}^{Sim(p,\tau,\cdot,\cdot,\cdot)}(s,p); \\ w \leftarrow \mathsf{Extract}(p, \tau, M_L, x, m, \sigma) : \\ M_L(x,w) \vee (M_L, x, m) \in Q^+ \vee \neg\mathbf{Verify}(p, M_L, x, m, \sigma) \end{array}\right] = 1 - \nu(\lambda),$$

where Q^+ denotes the query tape which lists all the previous successful queries $\mathcal{A}$ has sent to the oracle Sim, i.e. all those queries (M_L, x, m) which were sent with some valid witness w.

3.3 Identity-based Cryptography

Identity-based cryptography was introduced by Shamir [189] in 1984. It allows the user to use his identity (any arbitrary string such as email address) as his public key. Usually a trusted third party called "Private Key Generator" (PKG) computes the private key from the identity. Comparing with using a digital certificate in the

traditional public key cryptography, the identity-based public key can identify the user immediately. Besides, the problem of distribution of public keys is avoided in identity-based cryptography. Therefore, identity-based cryptography is an attractive alternative to the traditional public key cryptography.

3.3.1 Identity-based Signatures

Since the introduction of identity-based cryptography by Shamir [189], there are some identity-based signature schemes developed. The identity-based signatures and identity-based identifications are summarised by Bellare et al. [17]. They can be divided according to the underlying intractability assumptions.

- RSA: These include Shamir's first IBS scheme [189], the Guillou-Quisquater scheme [107], the Girault scheme [95], the Okamoto scheme [167], etc.
- Factorisation: the Fiat-Shamir IBS scheme [78], the Ohta-Okamoto scheme [166], the Fischlin-Fischlin scheme [80], etc.
- Discrete Logarithm: the Beth scheme [23], the Bellare et al. scheme [17], etc.
- Pairing: the Sakai et al. scheme [185], the Cha-Cheon scheme [55], the Hess scheme [111], etc.

Galindo *et al.* [89] further extended the discussion to IBS with various additional properties, which has more practical applications.

3.3.2 Identity-based Encryption

The first several proposals for IBE are not satisfactory [70, 201, 203, 147]. Some require that users do not collude. Some require tamper resistant hardware. Other require the PKG to spend a lot of time on private key generation. Identity-based encryption becomes practical only since 2001, when Boneh and Franklin [34] used a new mathematical tools called "Pairings". Since then, there are many identity-based cryptosystems built using pairings.

Several IBE schemes [53, 27, 110] are secure without random oracles under a weaker "selective-ID" model [53]. Later, Boneh and Boyen [28] and Waters [206] proposed IBE schemes which are provably secure without random oracles under the stronger "adaptive-ID" model of [34]. The main disadvantage of their scheme is that the size of the system parameters is large. Recently, Waters [208] and Lewko and Waters [139] proposed fully secure IBE schemes with short system parameters.

3.4 Cryptographic Protocols

In the real world, cryptographic applications are more complicated than just a simple encryption or signature scheme. As a result, we need different cryptographic protocols to satisfy different security requirements. In this section, we will introduce three protocols which will be used in later Chapters. Firstly, *oblivious transfer* is a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is received. It provides confidentiality to the receiver's selections. Secondly, *secret sharing* is a protocol to distribute a secret to a group of recipients, and each recipient can obtain a share of the secret. Finally, *fair exchange* is a protocol by which each participant holds a secret and all participants exchange their secrets in a fair manner.

3.4.1 Oblivious Transfer

In a k-out-of-n oblivious transfer [47], a sender has n messages, and a receiver can choose to receive k messages out of the n messages. There are two security requirements for oblivious transfer:

1. the sender learns nothing about the receiver's selections, and
2. the receiver only learns about the k requested messages.

Various efficient constructions are known based on decisional BDH (in random oracles) [104], DDH, quadratic residuosity [173] and some other specific assumptions [51, 105].

3.4.2 Secret Sharing

Secret sharing was invented by both Shamir [188] and Blakley [25] independently in 1979. In a k-out-of-n secret sharing scheme, a dealer wants to give a secret to n players. The dealer gives each player a share in a way that any group of k (for threshold) or more players can together reconstruct the secret but no group of fewer than k players can.

Shamir's Secret Sharing

We briefly review the Shamir's Secret Sharing [188]. Suppose the dealer wants to share a secret s in a finite field F with a group of n players. The dealer chooses at random $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ in F, and let $a_0 = s$. The dealer builds the polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}.$$

Each player i will receive a pair $(i, f(i))$, where $i = 1, \ldots, n$. Given any subset of k of these pairs, the players can find the coefficients of the polynomial using interpolation and the secret is the constant term a_0.

Linear Secret Sharing Schemes

We adapt the definition of Linear Secret Sharing Schemes (LSSS) in [14]. We first review the definition of access structure in [14].

Definition 22 (Access Structure [14]). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An* access structure *(resp. monotone access structure) is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets.

We restrict our attention to monotone access structure. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

Definition 23 (LSSS [14]). *A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called* linear *over $\mathbb{Z}_p$ if:*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix M called the share-generating matrix for Π. The matrix M has ℓ rows and n columns. For $i = 1, \ldots, \ell$, the i-th row of M we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.*

Beimel [14] showed that every LSSS enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Futhermore, Beimel [14] showed that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

3.4.3 Fair Exchange

Fair exchange protocol can be used to exchange encrypted messages or digital signatures between participating parties. Table 3.1 provides the fair exchange system considered

Table 3.1: Input and output values of fair exchange system considered in [170].

	P	Q
input	i_P, d_Q, Q	i_Q, d_P, P
output	i_Q with $desc(i_Q) = d_Q$ or "aborted"	i_P with $desc(i_P) = d_P$ or "aborted"

in [170]. The two parties P and Q want to exchange their items i_P and i_Q, respectively. It is assumed that there exists a function *desc* which maps any exchangeable item to some description which may be used to compare against some expected description value (d_P and d_Q).

Asokan [5] suggested that a fair exchange protocol should have the following properties, using the notations in Table 3.1:

- Effectiveness: If P and Q both behave correctly and do not abandon the exchange, then when the protocol has completed, P has i_Q such that $desc(i_Q) = d_Q$ and Q has i_P such that $desc(i_P) = d_P$.

- Fairness: It is further divided into:

  - *Strong Fairness*: When the protocol has completed, either both P and Q has i_Q and i_P respectively, such that $desc(i_Q) = d_Q$ and $desc(i_P) = d_P$, or both P and Q has gained no information about i_Q and i_P.

  - *Weak Fairness*[1]: When the protocol has completed, the following conditions must happen:

    * either P has i_Q such that $desc(i_Q) = d_Q$, or Q has gained no information about i_P, or P can prove to an arbiter that Q has received i_P such that $desc(i_P) = d_P$, without further intervention from P;
    * either Q has i_P such that $desc(i_P) = d_P$, or P has gained no information about i_Q, or Q can prove to an arbiter that P has received i_Q such that $desc(i_Q) = d_Q$, without further intervention from Q.

    The same condition holds for Q.

- Timeliness: P can be sure that the protocol will be completed at a certain point in time. At completion, the state of the exchange as of this point is either final or changes to the state will not degrade the level of fairness reached so far.

[1]Note that most fair exchange protocols with trusted third party only achieve weak fairness, since the trusted third party acts as the arbiter in the weak fairness definition.

Fair Exchange with Trusted Third Party

Most fair exchange protocols make use of a trusted (to various degrees) third party. Some protocols require the use of an *online* trusted third party, who actively involves in every exchange transaction. Some protocols, called "*optimistic fair exchange*"(OFE) [6], require the use of an *offline* trusted third party, who involves only when there is a dispute between the participants. There are many OFE protocols proposed since then. In 2007, Dodis, Lee and Yum [73] considered OFE in a multi-user setting. In 2008, Huang *et al.* [116] considered OFE in the chosen-key model. Huang *et al.* [115] also proposed the notion of *ambiguous OFE*, which can be viewed as a combination of OFE and concurrent signatures[2] with respect to the signer ambiguity property.

Fair Exchange without Trusted Third Party

There are several attempts to construct fair exchange protocols without trusted third party. Some researchers try to prove that fair exchange protocols without trusted third party is possible or impossible, under certain assumptions. We review some results as follows:

Gradual Release of Secrets *Gradual release of secrets* is a method for fair exchange without trusted third party. The exchange protocol consists of many rounds of communication. In each round, each participant releases a small portion of its secret. Each portion must be verifiable as a valid component of the secret. If a participant detects any misbehavbiour, it stops running the exchange protocol immediately. Assuming that all participants have roughly equal computational power, and the small portion gives no more information about the secret itself, the strong fairness property is guaranteed in a probabilistic sense. Several protocols of this type have been proposed, including [26, 39]. However, the main disadvantage of this method is that it is not efficient.

Relations with Distributed Consensus Distributed consensus (DC) is an essential building block for fault-tolerant distributed computing. Pagnia and Gärtner [170] show that when one process is prone to crash failure, DC between two processes is reducible to fair exchange without trusted third party (FE), which means that FE is harder than DC. This, along with the impossibility of DC in such settings [79], establish the unsolvability of FE.

[2]Concurrent signatures will be defined at the latter part of this section.

Orzan and Dashti [169] showed that the reduction of DC to FE in [170] is bound to two processes, and does not hold in general. They proved that in asynchronous systems where participating processes are prone to crash failures, while a majority of the processes are correct, FE is *incomparable* to DC.

Concurrent Signatures Chen, Kudla and Paterson [62] introduced the concept of *concurrent signatures* in 2004. Concurrent signatures allow two (or more) parties to exchange digital signatures in a fair manner. The initial signer generates a *keystone* and all participants sign on the corresponding *keystone fix*. All signatures can be verified only if the keystone is released. Otherwise, all signatures remain ambiguous.

In 2004, Susilo, Mu and Zhang [199] pointed out a problem for the ambiguity of concurrent signatures. They proposed a strengthened version called the *perfect concurrent signatures*. The requirements of (perfect) concurrent signatures include:

- Unforgeability: any efficient adversary with neither of the participants' secret keys cannot forge a valid concurrent signature with non-negligible probability.
- Ambiguity: given a concurrent signature without the keystone, any adversary cannot distinguish who of the two signers issued this signature.
- Fairness: it requires that:
  - a concurrent signature scheme should be fair for the initial signer P, i.e., only P can reveal the keystone; and
  - a concurrent signature scheme should be fair for the matching signer Q, i.e., once the keystone is released, both signatures are bound to their signers concurrently.

In 2006, Wang, Bao and Zhou [205] pointed out an attack against the perfect concurrent signatures by Susilo *et al.* [199] and by Chow and Susilo [66]. Wang *et al.* improved the security model of fairness to capture this attack. Some variants of concurrent signatures have also been proposed, including asymmetric concurrent signatures [161], identity-based concurrent signatures [66] and multi-party concurrent signatures [202].

The major difference between concurrent signatures and the fair exchange of digital signatures is that, in concurrent signatures the initial signer is able to control the release of keystone. In other words, the initial signer has the extra power to control whether the exchange protocol is complete or terminates. This power may bring unfairness to

the other participants which is outside the security model of concurrent signatures. In practical applications like contract signing, this extra power to the initial signer is undesirable.

Chapter 4

Digital Signatures

A digital signature is used to provide authenticity of a message. Once the signer generates a digital signature of a message, everyone can verify the signature by himself and believe that the message is not altered. In real world applications, the above statement may not be desirable in some cases. In this Chapter, we first study two interesting variants of standard signatures: undeniable signatures and sanitisable signatures. They differ from the above fundamental principles of digital signatures. Nevertheless, they have their own important practical applications.

In undeniable signatures, the verifier can only verify the signature with the interaction with the signer. Even if the verifier eventually believes that the signature is valid, he cannot prove it to any third party. It differs from the property of standard signatures that "everyone can verify the signature by himself". Undeniable signatures can be used in any situation in which an individual or a company wishes to sign a document, but does not want the press to be able to prove that he signed the document.

In sanitisable signatures, the signature of a message can be converted into a new signature, which remains valid even if part of the message is sanitised. This conversion does not require the signer's secret key. It is somehow contradictory to the standard signatures that "the signature is not valid if the message is altered". Nevertheless, sanitisable signatures is useful in cases like releasing confidential files from the government. The government can release signed documents while sanitising the confidential information.

In the existing literature, most of the concrete constructions of undeniable signatures and sanitisable signatures are provably secure in the random oracle model only. Several papers proved that some popular cryptosystems previously proved secure in the random oracle were actually provably insecure when the random oracle was instantiated by any real-world hashing functions [52, 16]. Therefore, it is more desirable to construct digital signatures in the standard model. In §4.1 and §4.2, we give new constructions of these undeniable signatures and sanitisable signatures without random

oracles respectively, under standard intractability assumptions.

In the last part of this Chapter, we discuss how two parties exchange their signatures in a fair manner. It has important applications like contract signing between two companies. In the literature, there are some fair exchange protocol which involves a trusted third party. If such trusted third party is not desirable in practice, we can use *concurrent signatures* to achieve the fair exchange. However, only the initiator of the concurrent signature protocol has the advantage to choose whether to complete the protocol or not. It is not so "fair" to the other party involved in the exchange protocol from this point of view. In §4.3, we define a new type of signature called *concinnous signatures*, to solve this problem in concurrent signatures.

4.1 Undeniable Signatures without Random Oracles

Standard digital signatures allow universal verification. However in some real world scenarios, privacy is an important issue. In this situation, we may require that the verification of signatures is restricted by the signer. Then, the verification of a signature requires an interaction with the signer. A signer can deny generating a signature that he never signs, but cannot deny one that he signs. The proof by the signer cannot be transferred to convince other verifiers. This concept is known as the "Undeniable Signatures" that was proposed by Chaum and van Antwerpen [60]. Later, Boyar, Chaum, Damgård and Pedersen [42] proposed an extension called "Convertible Undeniable Signatures", that allows the possibility to transform an undeniable signature into a self-authenticating signature. This transformation can be restricted to a particular signature only, or can be applied to all signatures of a signer.

There are many different undeniable signatures with variable features and security levels. These features include convertibility [42, 69, 151, 152], designated verifier technique [122], designated confirmer technique [59, 168], identity based scheme [142], time-selective scheme [136], etc. Undeniable signatures are said to be *secure* if it is unforgeable, invisible and the confirmation and disavowal protocols are zero-knowledge. It is believed that the zero-knowledgeness is required to make undeniable signatures non-transferable. However, Kurosawa and Heng [133] suggested that zero-knowledgeness and non-transferability can be separated; and the concept of witness indistinguishability can be incorporated. They proposed another security notion called impersonation attack.

Our Contribution. We propose the *first* convertible undeniable signatures without random oracles using pairings. Most of the existing convertible undeniable signatures are proven secure in the random oracle model only [42, 151, 152, 159, 136] [1], except the RSA-based construction in 2006 [134].

Most efficient undeniable signatures are proven secure in the random oracle model only. The scheme in [90] is secure in the random oracle model currently. [2] In 2005, Laguillaumie and Vergnaud proposed the first efficient undeniable signatures without random oracles [135]. However, their anonymity relies on their *new assumption* DSDH, while their unforgeability relies on the GSDH assumption with the access of a DSDH oracle. It seems to be contradictory becasuse their anonymity relies on the assumption that the DSDH problem is hard, while their proof of unforgeability requires an oracle which can solve the DSDH problem. Our proposed variant of undeniable signature is proven unforgeable by the CDH assumption and anonymous by the decision linear assumption. Therefore by removing the protocol for convertible parts, our undeniable signature scheme is the *first* proven secure scheme *without using random oracles* and *without using a new assumption* in discrete logarithm settings.

Recent Works. An earlier version of the scheme in this section appears in [210]. In 2007, Huang *et al.* [118] proposed a pairing-based convertible undeniable signatures secure in the random oracle model. Huang *et al.* [117] also proposed a generic construction of universally-convertible undeniable signatures from a strongly unforgeable classic signature scheme, a selectively-convertible undeniable signature scheme and a collision resistant hash function. In 2008, Kurosawa and Furukawa [132] defined the universal composability security of undeniable signatures.

In 2009, Phong *et al.* [177] proposed a new RSA-based selectively-convertible undeniable signatures. They also demonstrated an attack on the invisibility of the RSA-based construction in [134]. Phong *et al.* [176] proposed a new discrete-logarithm based selectively-convertible undeniable signature. This scheme is more efficient than our scheme proposed in this section. They pointed out a flaw in the earlier version of our scheme in [210]. This problem is fixed in the proposed scheme in this section.

[1] The scheme in [69] does not prove the invisibility property. The authors only conjecture the security in section 5.1 and 5.2.

[2] Refer to section 1.1 in [134] for details.

4.1.1 Security Models of Undeniable Signatures

We now review the security notions and model of (convertible) undeniable signatures. Unforgeability and invisibility are popular security requirements for undeniable signatures. Kurosawa and Heng [133] proposed another security notion called impersonation. We will use the security model of [133], and extend it to convertible undeniable signatures. The changes for convertible undeniable signatures will be given in brackets.

Security Notions

An (convertible) undeniable signature scheme has the following algorithms:

- **Setup**(1^λ): the setup algorithm takes a unary security parameter λ as input, and outputs some public parameters param.
- **KeyGen**(param): the key generation algorithm takes the public parameters param as input, and outputs a public key pk and a secret key sk.
- **USign**$(\mathsf{param}, \mathsf{sk}, m)$: the signing algorithm takes the public parameters param, a secret key sk and a message m as inputs, and outputs an undeniable signature σ.
- **Confirmation/Disavowal.** This is an interactive protocol between a prover and a verifier. Their common inputs are the public parameters param, a public key pk, a message m and a signature σ. The prover's private input is a secret key sk. At the end of the protocol, the verifier outputs 1 if σ is a valid signature of m and outputs 0 otherwise.

  (The following algorithms are for convertible schemes only.)

- **IConvert**$(\mathsf{param}, \mathsf{sk}, m, \sigma)$: The individual conversion algorithm takes the public parameters param, a secret key sk, a message m and a signature σ as inputs, and outputs an individual receipt r which makes it possible to individually verify σ.
- **IVerify**$(\mathsf{param}, \mathsf{pk}, m, \sigma, r)$: The individual verification algorithm takes the public parameters param, a public key pk, a message m, a signature σ and an individual receipt r as inputs, and
  - outputs $\perp$ if r is an invalid individual receipt, or
  - outputs 1 if σ is a valid signature of m, or

– outputs 0 if σ is not a valid signature of m.

- **UConvert**($\mathsf{param}, \mathsf{sk}$): The universal conversion algorithm takes the public parameters param and a secret key sk as inputs, and outputs an universal receipt R which makes it possible to universally verify all signatures for pk.

- **UVerify**($\mathsf{param}, \mathsf{pk}, m, \sigma, R$): The universal verification algorithm takes the public parameters param, a public key pk, a message m, a signature σ and an universal receipt R as inputs, and

  – outputs $\perp$ if R is an invalid universal receipt, or

  – outputs 1 if σ is a valid signature of m, or

  – outputs 0 if σ is not a valid signature of m.

The convertible undeniable signature schemes with all four algorithms (**IConvert, IVerify, UConvert, UVerify**) are sometimes denoted as *universally-convertible undeniable signature*. The convertible undeniable signature schemes with only the algorithms (**IConvert, IVerify**) are sometimes denoted as *selectively-convertible undeniable signature*.

Unforgeability

Strong unforgeability against chosen message attack is defined as in the following game involving an adversary $\mathcal{A}$ and a challenger over message space $\mathcal{M}$.

1. The challenger runs the algorithm $\mathsf{param} \leftarrow \mathbf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{KeyGen}(\mathsf{param})$. The challenger gives param and pk to $\mathcal{A}$. (For convertible schemes, the challenger also gives $\mathcal{A}$ the universal receipt $R \leftarrow \mathbf{UConvert}(\mathsf{param}, \mathsf{sk})$.)

2. $\mathcal{A}$ can query the following oracles adaptively:

   - Signing oracle: $\mathcal{A}$ requests a signature on any message $m \in \mathcal{M}$ and the challenger responds with $\sigma \leftarrow \mathbf{USign}(\mathsf{param}, \mathsf{sk}, m)$.
   - Confirmation/disavowal oracle: $\mathcal{A}$ queries the oracle with input message-signature pair (m, σ). If it is a valid pair, the challenger returns a bit $\mu = 1$ and proceeds with the execution of the **Confirm** protocol with $\mathcal{A}$. Otherwise, the challenger returns a bit $\mu = 0$ and proceeds with the execution of the **Deny** protocol with $\mathcal{A}$.
     (For convertible scheme, this oracle is not necessary as the universal receipt is given.)

3. Finally $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).

$\mathcal{A}$ wins the game if σ^* is a valid signature for m^* and the pair (m^*, σ^*) is not the output from the signing oracle.

Definition 24. *An (convertible) undeniable signature scheme is (ϵ, t, q_c, q_s)-strongly unforgeable against chosen message attack if there is no t time adversary winning the above game with probability greater than ϵ, where q_c and q_s are the number of queries to the confirmation/disavowal oracle and the signing oracle respectively.*

Invisibility

Invisibility against chosen message attack is defined as in the following game involving an adversary $\mathcal{A}$ and a challenger over message space $\mathcal{M}$.

1. The challenger runs the algorithm $\mathsf{param} \leftarrow \mathbf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{KeyGen}(\mathsf{param})$. The challenger gives param and pk to $\mathcal{A}$.

2. $\mathcal{A}$ can query the following oracles adaptively:

   - Signing oracle and Confirmation/disavowal oracle: they are the same as that in the unforgeability game.
   - (For convertible schemes only.) Receipt generating oracle: $\mathcal{A}$ queries the oracle with input message-signature pair (m, σ), and the challenger returns an individual receipt r.

3. $\mathcal{A}$ outputs a message m^*. The challenger choose a random bit b^*. If $b^* = 1$, then $\sigma^* \leftarrow \mathbf{USign}(\mathsf{param}, \mathsf{sk}, m^*)$. Otherwise σ^* is chosen uniformly at random from the signature space of the scheme.

4. $\mathcal{A}$ can adaptively query the signing oracle and confirmation/disavowal oracle, where no signing query (and receipt generating query) for m^* and no confirmation/disavowal query for (m^*, σ^*) is allowed.

5. Finally $\mathcal{A}$ outputs a guessing bit b'

$\mathcal{A}$ wins the game if $b^* = b'$ and there is no confirmation/disavowal query (and receipt generating query) for (m^*, σ^*). $\mathcal{A}$'s advantage is $Adv(\mathcal{A}) = |\Pr[b' = b^*] - \frac{1}{2}|$.

Definition 25. *An (convertible) undeniable signature scheme is (ϵ, t, q_c, q_r, q_s)-invisible if there is no t time adversary winning the above game with advantage greater than ϵ, where q_c, (q_r) and q_s are the number of queries to the confirmation/disavowal oracle, (the receipt generating oracle) and the signing oracle respectively.*

Impersonation

Impersonation against chosen message attack is defined as in the following game involving an adversary $\mathcal{A}$ and a challenger over message space $\mathcal{M}$.

1. The challenger runs the algorithm $\mathsf{param} \leftarrow \mathbf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathbf{KeyGen}(\mathsf{param})$. The challenger gives param and pk to $\mathcal{A}$.

2. $\mathcal{A}$ can query the Signing oracle and the Confirmation/disavowal oracle, which are the same as the one in the unforgeability game.

3. Finally $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*) and a bit b^*. If $b^* = 1$, $\mathcal{A}$ executes the confirmation protocol with the challenger. Otherwise, $\mathcal{A}$ executes the disavowal protocol with the challenger.

$\mathcal{A}$ wins the game if the challenger is convinced that σ^* is a valid signature for m^* if $b^* = 1$, or is an invalid signature for m^* if $b^* = 0$.

Definition 26. *An (convertible) undeniable signature scheme is* (ϵ, t, q_c, q_s)*-secure against impersonation if there is no* t *time adversary winning the above game with probability at least* ϵ*, where* q_c *and* q_s *are the number of queries to the confirmation/disavowal oracle and the signing oracle respectively.*

Remark. For convertible schemes, if an adversary can forge an individual or universal receipt, he can always convince a verifier in the interactive protocol, by directly giving the receipt to him. Therefore the model of impersonation attack already includes the security notion regarding receipts in convertible schemes.

4.1.2 Convertible Undeniable Signature Scheme

An earlier version of our scheme in [210] used the Waters signatures [206] (refer to §3.2.2 for details) and the 3-move witness indistinguishable protocol by Kurosawa and Heng [133]. However, there are two problems of the scheme in in [210]:

1. Ogata *et al.* [165] later proved that any 3-move confirmation/disavowal protocols are not secure against active attacks. As a result, the 3-move protocol by Kurosawa and Heng is insecure.

2. Phong *et al.* [176] showed that the proof of invisibility is not correct in [210]. The attack is based on the fact that the underlying signature scheme in [210] is not strongly unforgeable.

Therefore, we propose the use of the following techniques to solve these problems:

1. We use a standard 4-move proof of knowledge of discrete logarithm, or the non-interactive zero-knowledge proof system for bilinear groups by Groth and Sahai [106], to replace the protocol by Kurosawa and Heng in [210].

2. We use the generic construction of strongly unforgeable signatures in [22] in order to satisfy the stronger model of invisibility in [176].

Finally, we also use the proof technique in [108] to achieve a tight security reduction.

Scheme Construction

In this section, we present our convertible undeniable signature scheme. The scheme consists of the following algorithms.

- **Setup**(1^λ). Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p. Select generators $g, g_2 \in \mathbb{G}$. Generator $u' \in \mathbb{G}$ is selected in random, and a random n-length vector $\mathsf{U} = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. Select an integer ℓ as a system parameters. Let $H : \{0,1\}^n \to \mathbb{Z}_\ell^*$ be a collision resistant hash function. Let $\mathsf{SIG}_{OT} = (\mathsf{Kg}_{OT}, \mathsf{Sign}_{OT}, \mathsf{Verify}_{OT})$ be a secure one time signature scheme and the length of the verification key vk_{OT} is n-bits. The system parameters param are

$$(g, g_2, u', \mathsf{U}, H).$$

- **KeyGen**(param). Randomly select $\alpha, \beta', \beta_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq \ell$. Set $g_1 = g^\alpha$, $v' = g^{\beta'}$ and $v_i = g^{\beta_i}$. The public keys pk are $(g_1, v', v_1, \ldots, v_\ell)$. The secret keys sk are $(\alpha, \beta', \beta_1, \ldots, \beta_\ell)$.

- **USign**$(\mathsf{param}, \mathsf{sk}, m)$. To sign a message m, the signer runs $(sk_{OT}, vk_{OT}) \leftarrow \mathsf{Kg}_{OT}(1^\lambda)$. Denote $vk_{OT} = (vk_1, \ldots, vk_n) \in \{0,1\}^n$, and denote $\bar{vk} = H(vk_{OT})$. The signer picks $r \in_R \mathbb{Z}_p^*$ and computes the signature

$$S_1 = g_2^\alpha (u' \prod_{i=1}^{n} u_i^{vk_i})^r, \quad S_2 = (v' \prod_{i=1}^{\ell} v_i^{\bar{vk}^i})^r, \quad S_3 = \mathsf{Sign}_{OT}(sk_{OT}, m||S_1||S_2).$$

  The output signature σ is (S_1, S_2, S_3, vk_{OT}).

- **Confirmation/Disavowal.** On input a signature $\sigma = (S_1, S_2, S_3, vk_{OT})$, the signer computes:

$$\begin{aligned} L &= \hat{e}(g, g_2), \\ M &= \hat{e}(g_1, g_2), \\ N &= \hat{e}(v' \prod_{i=1}^{\ell} v_i^{\bar{vk}^i}, g_2), \\ O &= \hat{e}(v' \prod_{i=1}^{\ell} v_i^{\bar{vk}^i}, S_1) / \hat{e}(S_2, u' \prod_{i=1}^{n} u_i^{vk_i}). \end{aligned} \tag{4.1}$$

  Note that $\alpha = \log_L M = \log_N O$. The zero-knowledge proof of knowledge can be implemented using known 4-move proof of knowledge of discrete logarithm, or the non-interactive zero-knowledge proof system for bilinear groups by Groth and Sahai [106].

- **IConvert**$(\mathsf{param}, \mathsf{sk}, m, \sigma)$. Upon input the signature $\sigma = (S_1, S_2, S_3, vk_{OT})$ on the message m, the signer computes $\bar{vk} = H(vk_{OT})$ and

$$S_2' = S_2^{1/(\beta' + \sum_{i=1}^{\ell} \beta_i \bar{vk}^i)}.$$

  The signer outputs the individual receipt $r = S_2'$ for message m.

- **IVerify**$(\mathsf{param}, \mathsf{pk}, m, \sigma, r)$. Upon input the signature $\sigma = (S_1, S_2, S_3, vk_{OT})$ for the message m and the individual receipt $r = S_2'$, compute $\bar{vk} = H(vk_{OT})$ and check if:

$$\hat{e}(g, S_2) \stackrel{?}{=} \hat{e}(S_2', v' \prod_{i=1}^{\ell} v_i^{\bar{vk}^i}).$$

  If they are not equal, output $\perp$. Otherwise, denote $vk_{OT} = (vk_1, \ldots, vk_n)$ and compare if:

$$\begin{aligned} \hat{e}(g, S_1) &\stackrel{?}{=} \hat{e}(g_1, g_2) \cdot \hat{e}(S_2', u' \prod_{i=1}^{n} u_i^{vk_i}), \\ 1 &\stackrel{?}{=} \mathsf{Verify}_{OT}(vk_{OT}, S_3, m||S_1||S_2). \end{aligned}$$

  Output 1 if the all the above hold. Otherwise output 0.

- **UConvert**$(\mathsf{param}, \mathsf{sk})$. The signer publishes his universal receipt $R = (\beta', \beta_1, \ldots, \beta_\ell)$.

- **UVerify**$(\mathsf{param}, \mathsf{pk}, m, \sigma, R)$. Upon input the signature $\sigma = (S_1, S_2, S_3, vk_{OT})$ on the message m and the universal receipt $R = (\beta', \beta_1, \ldots, \beta_\ell)$, check if:

$$v' \stackrel{?}{=} g^{\beta'}, \qquad v_i \stackrel{?}{=} g^{\beta_i} \quad \text{for } 1 \leq i \leq \ell.$$

If they are not equal, output $\perp$. Otherwise compute $\bar{vk} = H(vk_{OT})$ and denote $vk_{OT} = (vk_1, \ldots, vk_n)$. Compare if:

$$\hat{e}(g, S_1) \stackrel{?}{=} \hat{e}(g_1, g_2) \cdot \hat{e}(S_2^{1/(\beta' + \sum_{i=1}^{\ell} \beta_i \bar{vk}^i)}, u' \prod_{i=1}^{n} u_i^{vk_i}),$$

$$1 \stackrel{?}{=} \mathsf{Verify}_{OT}(vk_{OT}, S_3, m||S_1||S_2).$$

Output 1 if all of the above hold. Otherwise output 0.

Security Result

Theorem 1. *The proposed convertible undeniable signature scheme is (ϵ, t, q_s)-strongly unforgeable if the (ϵ', t')-CDH assumption holds in $\mathbb{G}$, where*

$$\epsilon' \geq \frac{\epsilon}{2n+1}, \quad t' = t + O\Big(q_s(\rho + \omega)\Big),$$

and ρ, ω are the time for an exponentiation in $\mathbb{G}$ and for running Kg_{OT} and Sign_{OT} respectively.

Proof. Assume there is a (ϵ, t, q_s)-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the CDH problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a CDH problem instance (g, g^a, g^b). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. $\mathcal{B}$ runs $\mathsf{Kg}_{OT}(1^\lambda)$ for $2q_s$ times and obtains the pairs $(\mathsf{sk_t}, \mathsf{vk_t})$ for $1 \leq t \leq 2q_s$. $\mathcal{B}$ randomly selects the following integers:

- $x'_0 \in_R [0, 2n]$; $x'_1 \in_R [0, 2n]$; $y' \in_R \mathbb{Z}_p$, where $x'_0 \neq x'_1$.
- $x_i \in_R \{1, 2\}$, for $i = 1, \ldots, n$.
- $y_i \in_R \mathbb{Z}_p$, for $i = 1, \ldots, n$.

We further define the following functions for binary strings $\mathsf{vk_t} = (vk_{t,1}, \ldots, vk_{t,n})$ as follows:

$$F_0(\mathsf{vk_t}) = x'_0 + \sum_{i=1}^{n} x_i vk_{t,i}, \quad F_1(\mathsf{vk_t}) = x'_1 + \sum_{i=1}^{n} x_i vk_{t,i}, \quad J(\mathsf{vk_t}) = y' + \sum_{i=1}^{n} y_i vk_{t,i}.$$

For $j = 0, 1$, if there are at least q_s number of $\mathsf{vk_t}$ such that $F_j(\mathsf{vk_t}) = 0$ for $\mathsf{vk_t} \in \{\mathsf{vk_1}, \ldots, \mathsf{vk_{2q_s}}\}$, then there must be at least q_s number of $\mathsf{vk_t}$ satisfying $F_{1-j}(\mathsf{vk_t}) \neq 0$. Without loss of generality, assume $F_0(\mathsf{vk_t}) \neq 0$ holds for $\mathsf{t} = 1, \ldots, q_s$. We denote the function $F = F_0$ for simplicity.

$\mathcal{B}$ randomly picks $\beta', \beta_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq \ell$ and sets $v' = g^{\beta'}$ and $v_i = g^{\beta_i}$. $\mathcal{B}$ constructs a set of public parameters as follows:

$$g, \quad g_2 = g^b, \quad u' = g_2^{x'_0} g^{y'}, \quad u_i = g_2^{x_i} g^{y_i} \quad \text{for} \quad 1 \leq i \leq n.$$

The signer's public key is $(g_1 = g^a, v', v_1, \ldots, v_\ell)$.

Denote $\bar{vk}_t = H(\mathsf{vk_t})$ and $G(\mathsf{vk_t}) = \beta' + \sum_{i=1}^{\ell} \beta_i \bar{vk}_t^i$. Note that we have the following equation:

$$u' \prod_{i=1}^{n} u_i^{vk_{t,i}} = g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})}, \quad v' \prod_{i=1}^{\ell} v_i^{\bar{vk}_t^i} = g^{G(\mathsf{vk_t})}.$$

All the public parameters and the universal receipt $(\beta', \beta_1, \ldots, \beta_\ell)$ are passed to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:

(*Signing oracle.*) Upon receiving the t-th signing oracle query for a message m, $\mathcal{B}$ retrieves the key pairs $(\mathsf{sk_t}, \mathsf{vk_t})$. $\mathcal{B}$ randomly chooses $r \in_R \mathbb{Z}_p$ and computes

$$S_1 = g_1^{-\frac{J(\mathsf{vk_t})}{F(\mathsf{vk_t})}} \left(g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})}\right)^r, \quad S_2 = (g_1^{-\frac{1}{F(\mathsf{vk_t})}} g^{r_i})^{G(\mathsf{vk_t})}, \quad S_3 = \mathsf{Sign}_{OT}(\mathsf{sk_t}, m||S_1||S_2).$$

By letting $\tilde{r} = r - \frac{a}{F(\mathsf{vk_t})}$, it can be verified that $(S_1, S_2, S_3, \mathsf{vk_t})$ is a signature, shown as follows:

$$\begin{aligned}
S_1 &= g_1^{-\frac{J(\mathsf{vk_t})}{F(\mathsf{vk_t})}} (g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})})^r \\
&= g^{-\frac{aJ(\mathsf{vk_t})}{F(\mathsf{vk_t})}} (g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})})^{\frac{a}{F(\mathsf{vk_t})}} (g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})})^{-\frac{a}{F(\mathsf{vk_t})}} (g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})})^r \\
&= g^{-\frac{aJ(\mathsf{vk_t})}{F(\mathsf{vk_t})}} g_2^a g^{\frac{aJ(\mathsf{vk_t})}{F(\mathsf{vk_t})}} (g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})})^{\tilde{r}} \\
&= g_2^a (u' \prod_{j=1}^{n} u_j^{vk_{t,j}})^{\tilde{r}},
\end{aligned}$$

$$S_2 = (g_1^{-\frac{1}{F(\mathsf{vk_t})}} g^r)^{G(\mathsf{vk_t})} = (g^{r - \frac{a}{F(\mathsf{vk_t})}})^{G(\mathsf{vk_t})} = g^{G(\mathsf{vk_t})\tilde{r}} = (v' \prod_{i=1}^{\ell} v_i^{\bar{vk}_t^i})^{\tilde{r}}.$$

$\mathcal{B}$ outputs the signature $(S_1, S_2, S_3, \mathsf{vk_t})$. To the adversary, all signatures given by $\mathcal{B}$ are indistinguishable from the signatures generated by the signer. Notice that $F(\mathsf{vk_t}) \neq 0 \bmod p$ by the construction in the Setup phase.

Output. Finally $\mathcal{A}$ outputs a signature $\sigma^* = (S_1^*, S_2^*, S_3^*, vk_{OT}^*)$ for message $\mathfrak{m}^*$. Denote $vk_{OT}^* = \{vk_1^*, \ldots, vk_n^*\}$. $\mathcal{B}$ checks if $F(vk_{OT}^*) = 0 \bmod p$. If not, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes $\bar{vk}^* = H(vk_{OT}^*)$ and outputs

$$\frac{S_1^*}{{S_{2,1}^*}^{J(vk_{OT}^*)/G(vk_{OT}^*)}} = \frac{g_2^a \Big(u' \prod_{i=1}^n u_i^{vk_i^*}\Big)^r}{\Big(v' \prod_{i=1}^{\ell} v_i^{\bar{vk}_1^{*i}}\Big)^{rJ(vk_{OT}^*)/G(vk_{OT}^*)}} = \frac{g_2^a \Big(g^{J(vk_{OT}^*)}\Big)^r}{g^{rJ(vk_{OT}^*)}} = g^{ab},$$

which is the solution to the CDH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require that in the challenge phase, $F(vk_{OT}^*) = 0 \bmod p$. We consider the following cases:

- If $vk_{OT}^* \in \{\mathsf{vk_1}, \ldots, \mathsf{vk_{q_s}}\}$, and σ^* is not the output from the signing oracle query, then $\mathcal{B}$ obtains a forgery of the one time signature S_3^* with the message $m^*||S_1^*||S_2^*$.
- If $vk_{OT}^* \notin \{\mathsf{vk_1}, \ldots, \mathsf{vk_{q_s}}\}$, observe that $\sum_{i=1}^n x_i vk_{t,i} \in [0, 2n]$, where $x_i \in \{1, 2\}$ and $vk_{t,i} \in \{0, 1\}$. Since x_0' is chosen uniformly at random from $[0, 2n]$. Therefore

$$\Pr[F(vk_{OT}^*) = 0 \bmod p] = \frac{1}{2n+1}.$$

If the one time signature is secure, the probability of $\mathcal{B}$ not aborting is

$$\Pr[\mathsf{not\ abort}] \geq \frac{1}{2n+1}.$$

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{G}$ element and one Sign_{OT} in the signing stage. There are $2q_s$ of Kg_{OT} in the setup stage. The time complexity of $\mathcal{B}$ is

$$t + O\Big(q_s(\rho + \omega)\Big).$$

□

Theorem 2. *The scheme is $(\epsilon, t, q_c, q_r, q_s)$-invisible if the (ϵ', t')-decision linear assumption holds in $\mathbb{G}$, where*

$$\epsilon' \geq \frac{\epsilon}{2n+1}, \quad t' = t + O\Big((q_s + q_r)\rho + q_c\tau + q_s\omega\Big),$$

where ρ, τ, ω are the time for an exponentiation in $\mathbb{G}$, for an exponentiation in $\mathbb{G}_T$ and for running Kg_{OT} and Sign_{OT} respectively, under the assumption that $\ell > q_s$.

Proof. Assume there is a $(\epsilon, t, q_c, q_r, q_s)$-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the decisional linear problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a decisional linear problem instance (u, v, h, u^a, v^b, h^c). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. $\mathcal{B}$ runs $\mathsf{Kg}_{OT}(1^\lambda)$ for $2q_s + 2$ times and obtains the pairs $(\mathsf{sk}_\mathsf{t}, \mathsf{vk}_\mathsf{t})$ for $1 \leq t \leq 2q_s + 2$. $\mathcal{B}$ randomly selects the following integers:

- $x'_0 \in_R \mathbb{Z}_p$; $x'_1 \in_R \mathbb{Z}_p$; $y' \in_R [0, 2n]$, where $x'_0 \neq x'_1$.
- $x_i \in_R \mathbb{Z}_p$, for $i = 1, \ldots, n$.
- $y_i \in_R \{1, 2\}$, for $i = 1, \ldots, n$.

We further define the following functions for binary strings $\mathsf{vk}_\mathsf{t} = (vk_{t,1}, \ldots, vk_{t,n})$ as follows:

$$F_0(\mathsf{vk}_\mathsf{t}) = x'_0 + \sum_{i=1}^{n} x_i vk_{t,i}, \quad F_1(\mathsf{vk}_\mathsf{t}) = x'_1 + \sum_{i=1}^{n} x_i vk_{t,i}, \quad J(\mathsf{vk}_\mathsf{t}) = y' + \sum_{i=1}^{n} y_i vk_{t,i}.$$

For $j = 0, 1$, if there are at least $q_s + 1$ number of vk_t such that $F_j(\mathsf{vk}_\mathsf{t}) = 0$ for $\mathsf{vk}_\mathsf{t} \in \{\mathsf{vk}_1, \ldots, \mathsf{vk}_{2\mathsf{q_s}+2}\}$, then there must be at least $q_s + 1$ number of vk_t satisfying $F_{1-j}(\mathsf{vk}_\mathsf{t}) \neq 0$. Without loss of generality, assume $F_0(\mathsf{vk}_\mathsf{t}) \neq 0$ holds for $\mathsf{t} = 1, \ldots, q_s+1$. We denote the function $F = F_0$ for simplicity.

Assume that $\ell > q_s$. Denote the set $\bar{\mathcal{S}}$ as the set of numbers $\bar{vk_t} = H(\mathsf{vk}_\mathsf{t})$, for $\mathsf{t} = 1, \ldots, q_s$. Also denote the set $\mathcal{S} = \mathbb{Z}_\ell \setminus \bar{\mathcal{S}}$. We further define the following functions for any integer $\mathsf{vk}_\mathsf{t} \in \mathbb{Z}_\ell$

$$G(\mathsf{vk}_\mathsf{t}) = \prod_{i \in \mathcal{S}} (\bar{vk_t} - i) = \sum_{i=0}^{\ell - q_s} \gamma_i \bar{vk_t}^i \quad \text{and} \quad K(\mathsf{vk}_\mathsf{t}) = \prod_{i \in \bar{\mathcal{S}}} (\bar{vk_t} - i) = \sum_{i=0}^{q_s} \alpha_i \bar{vk_t}^i,$$

for some $\gamma_i, \alpha_i \in \mathbb{Z}_p$. For consistency, define $\gamma_{\ell - q_s + 1} = \ldots = \gamma_\ell = \alpha_{q_s+1} = \ldots = \alpha_\ell = 0$.

$\mathcal{B}$ constructs a set of public parameters as follows:

$$g = u, \quad g_2 = h, \quad u' = g_2^{x'} g^{y'}, \quad u_i = g_2^{x_i} g^{y_i} \ \text{ for } \ 1 \leq i \leq n.$$

The signer's public key is:

$$g_1 = u^a, \quad v' = v^{\alpha_0} g^{\gamma_0}, \quad v_i = v^{\alpha_i} g^{\gamma_i} \ \text{ for } \ 1 \leq i \leq \ell.$$

Note that we have the following equation:

$$u' \prod_{i=1}^{n} u_i^{vk_{t,i}} = g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})}, \quad v' \prod_{i=1}^{\ell} v_i^{\bar{vk_t}^i} = g^{G(\mathsf{vk_t})} v^{K(\mathsf{vk_t})},$$

where $\bar{vk_t} = H(\mathsf{vk_t})$. All public parameters are passed to $\mathcal{A}$. $\mathcal{B}$ also maintains an empty list $\mathcal{L}$.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:

(*Signing oracle.*) Upon receiving the t-th signing oracle query for a message m, $\mathcal{B}$ retrieves the key pairs $(\mathsf{sk_t}, \mathsf{vk_t})$. Note that by the construction in setup, we have $F(\mathsf{vk_t}) \neq 0 \bmod p$ and $K(\mathsf{vk_t}) = 0 \bmod p$. $\mathcal{B}$ randomly chooses $r \in_R \mathbb{Z}_p$ and computes

$$S_1 = g_1^{-\frac{J(\mathsf{vk_t})}{F(\mathsf{vk_t})}} \left(g_2^{F(\mathsf{vk_t})} g^{J(\mathsf{vk_t})}\right)^{r_i}, \quad S_2 = (g_1^{-\frac{1}{F(\mathsf{vk_t})}} g^{r_i})^{G(\mathsf{vk_t})}, \quad S_3 = \mathsf{Sign}_{OT}(\mathsf{sk_t}, m||S_1||S_2).$$

Same as the above proof, the signature $\sigma = (S_1, S_2, S_3, \mathsf{vk_t})$ is valid. $\mathcal{B}$ puts (m, σ) into the list $\mathcal{L}$ and then outputs the signature σ. To the adversary, all signatures given by $\mathcal{B}$ are indistinguishable from the signatures generated by the signer.

(*Confirmation/Disavowal oracle.*) Upon receiving a signature $\sigma = (S_1, S_2, S_3, \mathsf{vk_t})$ for message m, $\mathcal{B}$ checks whether (m, σ) is in $\mathcal{L}$. If so, $\mathcal{B}$ outputs Valid and runs the confirmation protocol with $\mathcal{A}$, to show that (L, M, N, O) in equation (4.1) are DH tuples. Notice that since $\mathcal{B}$ knows discrete logarithm of N with base L ($= 1/G(\mathsf{vk_t})$), it can simulate the interactive proof perfectly. Note that $G(\mathsf{vk_t}) \neq 0$ if $(m, \sigma) \in \mathcal{L}$.

If the signature is not in $\mathcal{L}$, $\mathcal{B}$ outputs $\mathsf{Invalid}$ and runs the disavowal protocol with $\mathcal{A}$. By theorem 1, the signature is strongly unforgeable if the CDH assumption holds. $\mathcal{B}$ runs the oracle incorrectly only if $\mathcal{A}$ can forge a signature. However if one can solve the CDH problem, he can also solve the decision linear problem.

(*Receipt generating oracle.*) Upon receive a signature $\sigma = (S_1, S_2, S_3, \mathsf{vk_t})$ for message m, $\mathcal{B}$ checks whether (m, σ) is in $\mathcal{L}$. If so, $\mathcal{B}$ outputs $S_2' = S_{2,1}^{1/G(\mathsf{vk_t})}$, which is a valid individual receipt for the signature. Otherwise, $\mathcal{B}$ returns $\perp$ which indicates that σ is not a valid signature.

Challenge. $\mathcal{A}$ gives m^* to $\mathcal{B}$ as the challenge message. $\mathcal{B}$ retrieves the key pairs $(\mathsf{sk_{q_s+1}}, \mathsf{vk_{q_s+1}})$. Denote $\mathsf{vk_{q_s+1}} = \{vk_1^*, \ldots, vk_n^*\}$ and $\bar{vk^*} = H(\mathsf{vk_{q_s+1}})$. Note by the construction in setup, we have $F(\mathsf{vk_{q_s+1}}) \neq 0 \bmod p$. We can also see that if $G(\mathsf{vk_{q_s+1}}) \neq 0 \bmod p$, then $\mathsf{vk_{q_s+1}} \in \bar{\mathcal{S}}$. It implies that $H(\mathsf{vk_{q_s+1}}) = H(\mathsf{vk_t})$ for some $t \in [1, \ldots, q_s]$. If the hash function H is collision resistant, then $G(\mathsf{vk_{q_s+1}}) = 0 \bmod p$.

If $J(\mathsf{vk}_{\mathsf{q_s}+1}) \neq 0 \bmod p$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ computes:

$$S_1^* = h^c, \quad S_2^* = v^{bK(\mathsf{vk}_{\mathsf{q_s}+1})/F(\mathsf{vk}_{\mathsf{q_s}+1})}, \quad S_3^* = \mathsf{Sign}_{OT}(\mathsf{sk}_{\mathsf{q_s}+1}, m^* || S_1^* || S_2^*).$$

and returns $(S_1^*, S_2^*, S_3^*, \mathsf{vk}_{\mathsf{q_s}+1})$ to $\mathcal{A}$.

Output. Finally $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ returns b' as the solution to the decision linear problem. Notice that if $c = a + b$, then:

$$S_1^* = g_2^{a+b} = g_2^a (g_2^{F(\mathsf{vk}_{\mathsf{q_s}+1})})^{b/F(\mathsf{vk}_{\mathsf{q_s}+1})} = g_2^a (u' \prod_{i=1}^{n} u_i^{m_i^*})^{b/F(\mathsf{vk}_{\mathsf{q_s}+1})},$$

$$S_2^* = v^{bK(\mathsf{vk}_{\mathsf{q_s}+1})/F(\mathsf{vk}_{\mathsf{q_s}+1})} = (v' \prod_{i=1}^{\ell} v_i^{\bar{vk}^{*i}})^{b/F(\mathsf{vk}_{\mathsf{q_s}+1})}.$$

Probability Analysis. For the simulation to complete without aborting, we require that in the challenge phase, $J(\mathsf{vk}_{\mathsf{q_s}+1}) = 0 \bmod p$. Observe that $\sum_{i=1}^{n} y_i vk_{t,i} \in [0, 2n]$, where $y_i \in \{1, 2\}$ and $vk_{t,i} \in \{0, 1\}$. Since y' is chosen uniformly at random from $[0, 2n]$. Therefore

$$\Pr[J(\mathsf{vk}_{\mathsf{q_s}+1}) = 0 \bmod p] = \frac{1}{2n+1}.$$

The probability of $\mathcal{B}$ not aborting is

$$\Pr[\mathsf{not\ abort}] \geq \frac{1}{2n+1}.$$

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{G}$ element and one Sign_{OT} in the signing stage. There are $O(1)$ exponentiations of $\mathbb{G}_T$ element in the confirmation/disavowal stage. There are $O(1)$ exponentiations of $\mathbb{G}$ element in the receipt generating stage. There are $2q_s + 2$ of Kg_{OT} in the setup stage. The time complexity of $\mathcal{B}$ is

$$t + O\Big((q_s + q_r)\rho + q_c\tau + q_s\omega\Big).$$

□

Theorem 3. *The scheme is (ϵ, t, q_c, q_s)-secure against impersonation if the (ϵ', t')-discrete logarithm assumption holds in $\mathbb{G}$, where*

$$\epsilon' \geq \frac{1}{2}(\epsilon - \frac{1}{p})^2, \quad t' = t + O\Big(q_s\rho + q_c\tau + q_s\omega\Big),$$

where ρ, τ, ω are the time for an exponentiation in $\mathbb{G}$, for an exponentiation in $\mathbb{G}_T$ and for running Kg_{OT} and Sign_{OT} respectively.

Proof. Assume there is a (ϵ, t, q_c, q_s)-adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the discrete logarithm problem with probability at least ϵ' and in time at most t'. $\mathcal{B}$ is given a discrete logarithm problem instance (g, g^a). The remaining proof is very similar to the proof of theorem 1, so we sketch the proof here.

With 1/2 probability, $\mathcal{B}$ sets $g_1 = g^a$ and hence the user secret key is a. The oracle simulation is the same as the proof in theorem 1, except that $\mathcal{B}$ now knows $b = \log_g g_2$. At the end of the game, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*) and a bit b^*. For either $b^* = 0/1$, $\mathcal{B}$ can extract a with probability 1/2, using the extractor of the proof of knowledge protocol.

With 1/2 probability, $\mathcal{B}$ sets $v' = g^a$ and hence $\mathcal{B}$ knows the secret key α. $\mathcal{B}$ can simulate the oracles perfectly with α. At the end of the game, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*) and a bit b^*. For either $b^* = 0/1$, $\mathcal{B}$ can extract $a + \sum_{i=1}^{\ell} \beta_i v \bar{k}^{*^i}$ with probability 1/2, using the extractor of the proof of knowledge protocol. Hence $\mathcal{B}$ can find a.

Probability Analysis. For the simulation to complete without aborting, we require that $\mathcal{B}$ correctly extract a at the end of the game. By Reset Lemma [19] (refer to §3.2.2 for details), it happens with probability at least $\frac{1}{2}(\epsilon - \frac{1}{p})^2$. We have

$$\epsilon' \geq \frac{1}{2}(\epsilon - \frac{1}{p})^2.$$

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{G}$ element and one Sign_{OT} in the signing stage. There are $O(1)$ exponentiations of $\mathbb{G}_T$ element and $O(1)$ modular addition in $\mathbb{Z}_p$ in the confirmation/disavowal stage. There are $2q_s + 2$ of Kg_{OT} in the setup stage. The time complexity of $\mathcal{B}$ is

$$t + O\Big(q_s\rho + q_c\tau + q_s\omega\Big).$$

□

Comparison

We improve the earlier version of our scheme in [210] in several ways. Firstly, our current scheme provides strong unforgeability while the earlier version provides existential unforgeability. Secondly, our current scheme fixes a flaw in the proof of invisibility

Table 4.1: Comparison of undeniable signatures in the standard model. Unf. stands for unforgeability, Inv. stands for invisibility, Imp. stands for impersonation. "exp" is the time for exponentiation in $\mathbb{G}$, "mul" is the time for multiplcation in $\mathbb{G}$, $|\sigma_{OT}|$ is the signature size of a one time signature, $|vk_{OT}|$ is the size of a one time verification key, and t_{OTS} is the time for computing a one-time signature.

Scheme	Unf.	Inv.	Imp.	$\lvert\sigma\rvert$	**USign** Time
[135]	q-GSDH	q-DSDH	?	1 $\mathbb{G}$, 1 $\mathbb{Z}_p$	1 exp + 1 mul
[176] SCUS_1	q-SDH	DLIN	?	3 $\mathbb{G}$, 1 $\mathbb{Z}_p$	4 exp + 1 mul
[176] SCUS_2	q-SDH	DLIN	?	3 $\mathbb{G}$, 1 $\mathbb{Z}_p$	4 exp + 1 mul
Our scheme	CDH	DLIN	DL	2 $\mathbb{G}$, 1 $\lvert\sigma_{OT}\rvert$, 1 $\lvert vk_{OT}\rvert$	$n+\ell$ exp + $n+\ell+1$ mul + t_{OTS}

[176]. Finally, our current scheme significantly reduces the reduction loss in the security proof. The earlier version of our scheme [210] has sub-exponential reduction loss. Our current scheme has $O(n)$ reduction loss only.

We propose the first convertible undeniable signatures without random oracles using pairings. Comparing with the part of undeniable signatures, our scheme is better than the existing undeniable signatures without random oracles [135] by using more standard assumption in the security proofs.

In 2009, Phong *et al.* [176] proposed another convertible undeniable signatures without random oracles using pairings. Their SCUS_1 and SCUS_2 are more efficient than our current scheme, since it has less public keys and less multiplication in the USign algorithm. However, our current scheme uses the CDH assumption for unforgeability, while they use the non-static q-SDH assumption.

We also note that the other undeniable signature schemes does not consider the impersonation security model proposed in [133].

4.2 Sanitisable Signatures without Random Oracles

A digital signature prohibits any alteration of the original message once it is signed. It protects the signer against the message forgery. Nevertheless, it also prevents the message from being processed further legitimately as well, which sometimes is actually desirable.

A typical example of sanitisable signature includes the case when the government wants to release some *partial* information in an officially signed document. In this particular case, a government officer may want to delete some sensitive information

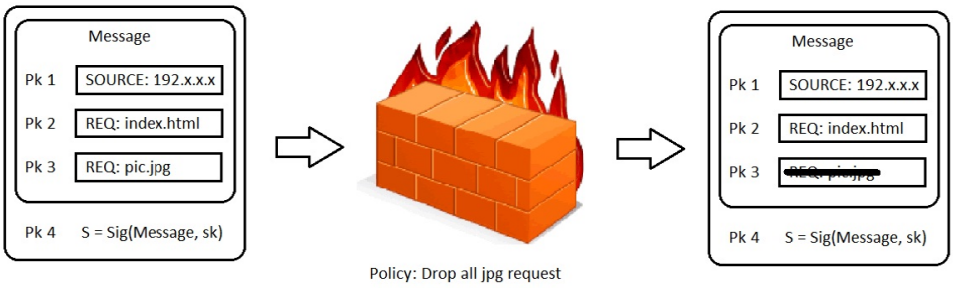


Figure 4.1: How sanitisable signatures work in an firewall scenario.

such as personal information or national secrets. In order to avoid the process of having the message to be signed again (since the original signer may not be available at that time), a sanitisable signature can be used to sign the document at the first place; and the sensitive information can be sanitised *prior to* the release of the signature.

In a networking scenario where an application level firewall is employed, the firewall can examine the packets at the application level. A packet reaches the firewall and is passed to an application-specific proxy, which inspects the validity of the packet. For example, if a Web request (HTTP) comes in, the data payload containing the HTTP request will be passed to an HTTP-proxy process. When the data payload does not satisfy the condition setup in the application proxy, the packet will be dropped. The problem arises when the overall packets are actually authenticated by the sender. If the complete packets are delivered to the receiver, then the receiver can verify the authenticity of the packets by verifying the signature attached. Nonetheless, if part of the packets have been dropped, then these packets can no longer be authenticated unless the sender signs the "new" packets again. The application-proxy cannot sign on behalf of the signer since the application-proxy does not hold the sender's secret key. In this scenario, we require the "sanitised" packets to be authenticated, and therefore the application-proxy should be able to somehow obtain the correct signature on the sanitised packets. This is where sanitisable signature can come into play. Consider a message containing three packets, as shown in Figure 4.2. The fourth packet is a sanitizable signature on the message. The firewall policy is to "drop all jpg request". As a result, the firewall drops the third packet. Finally, the receiver gets the filtered message and the signature. He can still verify the signature due to the property of sanitizable signatures.

The major goal of sanitisable signature is to protect the confidentiality of part of

the document while ensuring the integrity of the document. This is called the "digital document sanitising problem" in [158]. Similar solutions have been proposed earlier in [197] as "content extraction signature"; and in [123] as "redactable signature". Ateniese et al. [7] introduced the "sanitisable signature" which can change the signed document instead of hiding the signed document. Following these works, several authors [157, 200, 131, 156, 121, 56] proposed various sanitisable signature schemes with different properties.

One of the major differences between the existing schemes is due to the information used to replace a sanitised message. The majority of these works uses a special character, ϕ, to represent a sanitised message. In contrast to this approach, Miyazaki et al. [156] directly removed the sanitised message and the verifier does not even notice that the original document has been sanitised. Several other works [7, 131, 121] replaced the sanitised message to construct a new message. In order to prevent forgery by the adversary, the sanitiser needs to use his secret key in the sanitising process. Chang et al. [56] proposed a scheme which hides the number (length) of sanitised messages. Another distinct feature among these works is how to restrict the sanitisation for part of the document. Some schemes can sanitise any part of the document. Some schemes can prohibit some part of the documents from being sanitised, and this decision can be made after the document is signed, performed by either the signer or anyone else. Furthermore, the designation of sanitiser is another difference between these schemes. Some schemes select the designated sanitiser a priori when signing. On the contrary, anyone can sanitise a message in many other schemes. Transparency is also considered as a new property in some schemes. If a verifier knows which part of the document is sanitised, then the scheme has no transparency. If he does not know whether the message is sanitised, then the scheme has weak transparency. If he also does not know whether the message can be sanitised, then the scheme has strong transparency.

Our Contribution. In this section, our contribution is twofold. Firstly, we formalise the security model for sanitisable signatures to capture different properties of sanitisable signatures in the literature. We provide a generic conversion between some of the properties of sanitisable signatures. Secondly, we also provide a new concrete construction which is proven secure in the standard model, without resolving the security to underlying signature schemes. It is the *first* concrete construction of sanitisable signature scheme that is provably secure under a standard assumption without random oracle. We provided a security analysis based on our proposed model.

4.2.1 Security Models of Sanitisable Signatures

In this section we review the security notions and models of sanitisable signatures. We extend the model introduced in [200].

Notation

In this section, we describe some terms for sanitisable signatures.

Document, Message and Flag. We denote a *document* M as a list of *messages* $m_1||m_2||\ldots||m_n$, where the length n is the number of messages. We denote that $||$ is the concatenation. We use a *flag* ϕ as the sanitised message. For two messages M^1 and M^2 having the same length, we say that M^1 is a *subdocument* of M^2 if $m_i^1 = m_i^2$ for all i where m_i^1 are not sanitised.

State. For a message M, let st_{M} be the states of m_i. A state can be either:

- sanitised,
- disclosed and sanitising is allowed, or
- disclosed and sanitising is prohibited.

st_{M} is constructed as $(st^S_{\mathsf{M}}, st^A_{\mathsf{M}}, st^P_{\mathsf{M}})$, where st^S_{M} is a set of indices of sanitised messages; st^A_{M} is a set of indices of the messages that are "disclosed and sanitising is allowed"; st^P_{M} is a set of indices of the messages that are "disclosed and sanitising is prohibited".

Syntax

Sanitisable signatures consist of four algorithms:

- **KeyGen**(1^λ). On input the security parameter 1^λ, it outputs a public key and a private key $(\mathsf{pk}, \mathsf{sk})$ and the system parameters param.
- **Sign**$(\mathsf{param}, \mathsf{M}, \mathsf{sk}, st_{\mathsf{M}})$. On input the system parameters param, a document M, a secret key sk and a state st_{M} of the document, it outputs a signature σ.
- **Sanitise**$(\mathsf{param}, \sigma, \mathsf{M}, st_{\mathsf{M}}, st_{\mathsf{M}'})$. On input the system parameters param, a signature σ, (a document M,) a state st_{M} of M, and a new state of the sanitised document $st_{\mathsf{M}'}$, it outputs a sanitised document M' and a new signature σ'. It may output $\perp$ if there exists $i \in st^S_{M'}$ which is also in st^P_M.

- **Verify**$(\mathsf{param}, \sigma, \mathsf{M}, st_{\mathsf{M}}, \mathsf{pk})$. On input the system parameters param, a signature σ, a document M, a state st_{M} of M, and a public key pk, it outputs $\top$ for valid signature and $\perp$ otherwise.

Security Model

Correctness. We require that **Verify**$(\mathsf{param}, \sigma', \mathsf{M}', st_{\mathsf{M}'}, \mathsf{pk}) = \top$ if:

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{param}) \leftarrow$ **KeyGen**(1^λ),
- $\sigma \leftarrow$ **Sign**$(\mathsf{param}, \mathsf{M}, \mathsf{sk}, st_{\mathsf{M}})$,
- $(\mathsf{M}', st_{M'}, \sigma') \leftarrow$ **Sanitise**$(\mathsf{param}, \sigma, st_{\mathsf{M}}, st_{\mathsf{M}'}, \mathsf{pk})$.

Unforgeability. We have the following game for unforgeability.

1. The challenger runs $(\mathsf{pk}, \mathsf{sk}, \mathsf{param}) \leftarrow$ **KeyGen**(1^λ) and gives param and pk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to query the signing oracle q_s times adaptively. During the j-th query, on input a document $\mathsf{M}^j = m_1^j || m_2^j || \ldots, m_n^j$ and the state st_{M^j}, the oracle returns the signature $\sigma \leftarrow$ **Sign**$(\mathsf{param}, \mathsf{M}^j, \mathsf{sk}, st_{\mathsf{M}^j})$.
3. Finally $\mathcal{A}$ outputs a document M^*, a signature σ^* and a state st_{M^*}.

$\mathcal{A}$ wins if **Verify**$(\mathsf{param}, \sigma^*, \mathsf{M}^*, st_{\mathsf{M}^*}, \mathsf{pk}) = \top$ and one of the following holds:

1. M^* is not a subdocument of any M^j for $1 \leq j \leq q_s$.
2. M^* is a subdocument of some M^j for $1 \leq j \leq q_s$ and some m_i^* are sanitised, where $i \in st^P_{\mathsf{M}^j}$.
3. M^* is a subdocument of some M^j for $1 \leq j \leq q_s$ and there exists some i such that $i \in st^S_{\mathsf{M}^j} \wedge i \notin st^S_{\mathsf{M}^*}$.

Definition 27. *A sanitisable signature scheme is (ϵ, t, q_s)-unforgeable if there is no t time adversary winning the above game with probability at least ϵ with at most q_s queries to the signing oracle.*

Indistinguishability. We have the following game for indistinguishability.

1. The challenger runs $(\mathsf{pk}, \mathsf{sk}, \mathsf{param}) \leftarrow$ **KeyGen**(1^λ) and gives param and pk to the adversary $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the signing oracle q_s times adaptively. The oracle is the same as in the game for unforgeability.

3. $\mathcal{A}$ gives the challenger two documents M^0, M^1 and a state st_{M^*}. It is required that:

   - $m_i^0 = m_i^1$ for all $i \notin st^S_{\mathsf{M}^*}$ and
   - $m_i^0 \neq m_i^1$ for some $i \in st^S_{\mathsf{M}^*}$.

   The challenger first checks if the documents satisfy the requirements. Then the challenger randomly chooses a bit b^* and sends the signature $\sigma^{b^*} \leftarrow \mathbf{Sign}(\mathsf{param}, \mathsf{M}^{b^*}, \mathsf{sk}, st_{\mathsf{M}^*})$.

4. Finally $\mathcal{A}$ outputs a bit b'.

$\mathcal{A}$ wins the game if $b^* = b'$. The advantage of $\mathcal{A}$ is $|\Pr[b^* = b'] - 1/2|$.

Definition 28. *A sanitisable signature scheme is (ϵ, t, q_s)-indistinguishable if there is no t time adversary winning the above game with advantage at least ϵ with at most q_s queries to the signing oracle.*

Various Properties and their Implications in Security

We will discuss various properties of sanitisable signature schemes. We extend the discussion from [200] by adding more properties from various schemes. We then explain the impacts on the security model.

State Controllability. We consider three types of state controllability:

1. The sanitiser can sanitise any message he wants and the signer cannot restrict it. In the security model, there is no state st^P_{M}.

2. The signer can assign the states st^P_{M} or st^A_{M} to the non-sanitised message. However the states cannot be changed from st^A_{M} to st^P_{M} without the signer's secret key after the signature is generated. This property imposes a restriction on the **Sanitise** protocol. To reflect this in the security model, we add an extra condition for the adversary to win the unforgeability game: M^* is a subdocument of some M^j for $1 \leq j \leq q_s$ and there exists some i such that $i \in st^A_{\mathsf{M}^j} \wedge i \notin st^P_{\mathsf{M}^*}$.

3. The signer can assign the states st^P_{M} or st^A_{M} to the non-sanitised message. The states can be changed from st^A_{M} to st^P_{M} without the signer's secret key. The current model is for this type. It was proposed by [157] to prevent the *additional sanitising attack*.

Sanitised Message. We consider four different types of *sanitised message*:

1. The sanitisation of the message causes the shortening of the message. In the security model, the sanitised message ϕ is equal to a null string. The definition of a *subdocument* is changed as follows: M^1 is a subdocument of M^2 if it can be obtained from M^2 by removing some non-empty messages in it. The security model includes an extra *invisibility* game [156], which can be included in our indistinguishability game by setting the challenge document (to form the challenge signature) as a subdocument of both M^0 and M^1.

2. Each sanitised message is represented by a special character ϕ. Everyone can notice where the document is sanitised. The current model is for this type.

3. Each sanitised message is represented by a special character ϕ and consecutive ϕs can be combined into one. The length of the sanitised message is hidden. The definition of a *subdocument* is changed as follows: M^1 is a subdocument of M^2 if it can be obtained from M^2 by removing some non-empty messages in it and replacing it by a single ϕ. The security model in [56], which can be included in our indistinguishability game by setting the challenge document as a subdocument of both M^0 and M^1.

4. Each sanitised message can be changed to any message chosen by the sanitiser. In the security model, the sanitised message ϕ is equal to a new message m'_i[3].

Designated Sanitiser. We consider two types of designation of sanitiser.

1. The signer cannot choose who are the designated sanitisers when he signs the document. The current model is for this type. In some protocols, **Sign** also outputs a secret information SI[4] to the sanitiser. **Sanitise** will then have an additional input SI. In the security model, the signing oracle should also output SI[5].

[3]Possible extension includes *enforcing the same modification of different messages*, and *limiting the number of modifications* [131].

[4]SI is firstly formalised in [200], but the idea is implicit in early papers.

[5]If several SI can combine together to form an aggregate SI, the scheme has the *binding subdocuments* function [156].

2. The signer has to designate the specific sanitisers. In the protocol, **KeyGen** will also generate the keys for sanitisers. **Sign** should also take the sanitisers' public keys as the input. **Sanitise** will have the sanitisers' secret keys as an additional input[6]. For the unforgeability and indistinguishability game, the adversary is given the public and private keys of the sanitisers. It prevents the attack from dishonest sanitisers like the *Deletion-of-Last-Sanitiser Attack* in [121]. At the challenge phase of the indistinguishability game, the adversary also gives the public keys of the sanitisers to the challenger. The adversary may have the secret keys of the challenge sanitisers[7].

Transparency. We consider three types of transparency.

1. No transparency. The verifier knows which part of the document is sanitised. The current model is for this type. The *sanitised message* must be either type 2 or type 3.

2. Weak transparency. The verifier does not know if the message is sanitised. The verifier only knows if the state is st_{M}^{P} or not. In the model the states st_{M}^{S} and st_{M}^{A} are combined into one state when it is sent to the verifier. The *sanitised message* must be either type 1 or type 4.

3. Strong transparency. The verifier does not know if the message can be sanitised. In the model the state information is not sent to the verifier. The *sanitised message* must be either type 1 or type 4.

Generic Conversion

As there are different properties of sanitisable signatures needed in different scenarios, we propose some generic conversions between different properties.

State Controllability. Type 2 and type 3 can be converted to type 1 by forbidding state st_{M}^{P} in the scheme.

Sanitised Message. Type 4 can be converted to type 2 by using a special character ϕ. Type 3 can be converted to type 2 by using special characters ϕ_1 and ϕ_2 alternatively.

[6]It is optional for **Verify** to take the sanitisers' public keys as the input, e.g. in [120, 121]. It will then have the property *sanitiser identification*.

[7][121] has extensions called *dishonest sanitation identification* and *dishonest sanitiser identification*. However there is no formal model proposed in [121].

Designated Sanitiser. Type 1 can be converted to type 2 by verifiablely encrypting the secret information to the designated sanitiser. Type 2 can be converted to type 1 by publishing a private and public key pairs and always designating to that public key.

4.2.2 Sanitisable Signature Scheme

We present our sanitisable signature scheme. It is motivated by Waters signatures [206] (refer to §3.2.2 for details). Our scheme consists of the following algorithms.

Key Generation. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p. Given a pairing: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Select g, $h_1, \ldots, h_n \in \mathbb{G}_1$ and $g_2, u', u_1, \ldots, u_n \in \mathbb{G}_2$. The system parameters is $\mathsf{param} = (g, g_2, u', u_1, \ldots, u_n, h_1, \ldots, h_n)$. The signer randomly picks a secret key sk as $\alpha \in \mathbb{Z}_p^*$ and his public key pk is computed as $g_1 = g^\alpha$.

Sign. To sign a n-bit document $M \in \{0,1\}^n$ with a state st_M of M, we first encode the document as $m_1 m_2 \ldots m_n \in \{-1, 1\}^n$, where for $i = 1, \ldots, n$:

- $m_i = 1$ if the i-th bit of M is 1,
- $m_i = -1$ if the i-th bit of M is 0.

The signer randomly picks $r \in \mathbb{Z}_p^*$ and returns (σ_1, σ_2), where:

$$\sigma_1 = g_2^\alpha (u' \prod_{i=1}^{n} u_i^{m_i})^r, \quad \sigma_2 = g^r.$$

Sanitise. Upon input a signature (σ_1, σ_2), a document M, an old state st_M and a new state $st_{M'}$, the sanitiser does the followings:

1. Encode M as $m_1 m_2 \ldots m_n$.
2. Check for all $i \in st_{M'}^S$ is also in st_M^A. If not, return $\perp$ and exit.
3. Check if $\hat{e}(g, \sigma_1) = \hat{e}(g_1, g_2) \cdot \hat{e}(\sigma_2, u' \prod_{i=1}^n u_i^{m_i})$. If not, return $\perp$ and exit.
4. For all $i \in st_{M'}^S$, pick a random $t_i, s_i \in \mathbb{Z}_p^*$, $s \in \mathbb{Z}_p^*$ and then compute:

$$\sigma_1' = \sigma_1 \cdot (\prod_{j \notin st_{M'}^S} u_j^{m_j})^s \cdot (\prod_{i \in st_{M'}^S} u_i^{m_i t_i}), \quad \sigma_2' = \sigma_2 \cdot g^s,$$
$$A_i = (\sigma_2 g^{t_i})^{m_i} h_i^{s_i}, \quad B_i = u_i^{s_i} \qquad \text{for all } i \in st_{M'}^S.$$

5. For $i \in st^S_{M'}$, change the i-th bit of M as ϕ to form a document M'. The sanitised signature is $(\sigma'_1, \sigma'_2, \{A_i, B_i | i \in st^S_{M'}\})$.

Verify. Upon receiving a signature $(\sigma'_1, \sigma'_2, \{A_i, B_i | i \in st^S_M\})$ and a document M, encode M as $m_1 m_2 \dots m_n \in \{\phi, -1, 1\}^n$, check if:

$$\hat{e}(g, \sigma'_1) \cdot \prod_{i|m_i=\phi} \hat{e}(h_i, B_i) = e(g_1, g_2) \cdot \hat{e}(\sigma'_2, u' \prod_{j|m_j \neq \phi} u_j^{m_j}) \cdot \prod_{i|m_i=\phi} \hat{e}(A_i, u_i).$$

Return $\top$ if the above holds. Otherwise return $\perp$.

Security Result

We prove the security of our scheme under the model of state type 1, message type 2, sanitiser type 1 and transparency type 1, as defined in section 4.2.1.

The correctness of our scheme is as follows.

$$\begin{aligned}
&\hat{e}(g, \sigma'_1) \cdot \prod_{i|m_i=\phi} \hat{e}(h_i, B_i) \\
=&\hat{e}(g, g_2^{\alpha}(u' \prod_{i=1}^{n} u_i^{m_i})^r \cdot (\prod_{j \notin st^S_M} u_j^{m_j})^s \cdot (\prod_{i \in st^S_M} u_i^{m_i t_i})) \cdot \prod_{i|m_i=\phi} \hat{e}(h_i, u_i^{s_i}) \\
=&\hat{e}(g_1, g_2) \cdot \hat{e}(g, (u' \prod_{j|m_j \neq \phi} u_j^{m_j})^{r+s} \cdot \prod_{i|m_i=\phi} u_i^{m_i(r+t_i)}) \cdot \prod_{i|m_i=\phi} \hat{e}(h_i, u_i^{s_i}) \\
=&\hat{e}(g_1, g_2) \cdot \hat{e}(\sigma'_2, u' \prod_{j|m_j \neq \phi} u_j^{m_j}) \cdot \prod_{i|m_i=\phi} (\hat{e}(g, u_i^{m_i(r+t_i)}) \cdot \hat{e}(h_i, u_i^{s_i})) \\
=&\hat{e}(g_1, g_2) \cdot \hat{e}(\sigma'_2, u' \prod_{j|m_j \neq \phi} u_j^{m_j}) \cdot \prod_{i|m_i=\phi} \hat{e}(A_i, u_i).
\end{aligned}$$

Theorem 4. *Our sanitisable signature scheme is $(\epsilon,\ t,\ q_s)$-unforgeable if the (ϵ', t')-CDH assumption holds, where:*

$$\epsilon' \geq \frac{\sqrt{\pi}\epsilon}{4q_s(n+1)\sqrt{2n}2^n}, \quad t' = t + O(q_s(\rho + \tau)),$$

and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively.

Proof. Assume there is a (ϵ, t, q_s)-adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the CDH problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a problem instance as follows: Given a group $\mathbb{G}$, a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$. It is asked to output another element $g^{ab} \in \mathbb{G}$. In order to use $\mathcal{A}$

to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the signing oracle for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. Let $l = 2q_s$. Also assume that $l(n+1) < p$, for the given values of q_s and n. $\mathcal{B}$ guesses the number of sanitised messages in the challenge message as N. $\mathcal{B}$ picks a set of N random integers from $[1, n]$ and we denote this set as S. $\mathcal{B}$ sets $\hat{x}_j = 0$ for all $j \in S$. $\mathcal{B}$ randomly selects the following integers:

- $x' \in_R [-l, l]$; $y' \in_R \mathbb{Z}_p$.
- $\hat{x}_i \in_R [0, l]$, for $i \in [1, n] \setminus S$.
- $\hat{y}_i, \hat{z}_i \in_R \mathbb{Z}_p$, for $i = 1, \ldots, n$.

We further define the following functions for a document $\mathfrak{m}$. Encode $\mathfrak{m}$ as (m_1, m_2, …, m_n), where

- $m_i = 1$ if the i-th bit of $\mathfrak{m}$ is 1,
- $m_i = -1$ if the i-th bit of $\mathfrak{m}$ is 0,
- $m_i = 0$ if the i-th bit of $\mathfrak{m}$ is ϕ.

$$F(\mathfrak{m}) = x' + \sum_{i=1}^{n} \hat{x}_i m_i \quad \text{and} \quad J(\mathfrak{m}) = y' + \sum_{i=1}^{n} \hat{y}_i m_i.$$

$\mathcal{B}$ constructs a set of public parameters as follows:

$$g_2 = g^b, \quad u' = g_2^{x'} g^{y'}, \quad u_i = g_2^{\hat{x}_i} g^{\hat{y}_i}, \quad h_i = g^{\hat{z}_i} \text{ for } 1 \leq i \leq n.$$

We have the following equation:

$$u' \prod_{i=1}^{n} u_i^{m_i} = g_2^{F(\mathfrak{m})} g^{J(\mathfrak{m})}.$$

All the above public parameters and public key $g_1 = g^a$ are passed to $\mathcal{A}$.

Oracle Simulation. $\mathcal{B}$ simulates the signing oracle as follows. Upon receiving a j-th query for a document $\mathfrak{m}_j$, although $\mathcal{B}$ does not know the secret key, it can still construct the signature by assuming $F(\mathfrak{m}_j) \neq 0 \bmod p$. It randomly chooses $r_j \in_R \mathbb{Z}_p$ and computes the signature as

$$\sigma_{1,j} = g_1^{-\frac{J(\mathfrak{m}_j)}{F(\mathfrak{m}_j)}} \left(g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)}\right)^{r_j}, \quad \sigma_{2,j} = g_1^{-\frac{1}{F(\mathfrak{m}_j)}} g^{r_j}.$$

By letting $\tilde{r}_j = r_j - \frac{a}{F(\mathfrak{m}_j)}$, it can be verified that $(\sigma_{1,j}, \sigma_{2,j})$ is a valid signature, shown as follows:

$$\begin{aligned}
\sigma_{1,j} &= g_1^{-\frac{J(\mathfrak{m}_j)}{F(\mathfrak{m}_j)}} (g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)})^{r_j} \\
&= g^{-\frac{aJ(\mathfrak{m}_j)}{F(\mathfrak{m}_j)}} (g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)})^{\frac{a}{F(\mathfrak{m}_j)}} (g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)})^{-\frac{a}{F(\mathfrak{m}_j)}} (g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)})^{r_j} \\
&= g_2^a (g_2^{F(\mathfrak{m}_j)} g^{J(\mathfrak{m}_j)})^{\tilde{r}_j}, \\
\sigma_{2,j} &= g_1^{-\frac{1}{F(\mathfrak{m}_j)}} g^{r_j} = g^{r_j - \frac{a}{F(\mathfrak{m}_j)}} = g^{\tilde{r}_j}.
\end{aligned}$$

To the adversary, all signatures given by $\mathcal{B}$ are indistinguishable from the signatures generated by the true challenger.

If $F(\mathfrak{m}_j) = 0 \bmod p$, since the above computation cannot be performed (division by 0), the simulator aborts. To make it simple, the simulator will abort if $F(\mathfrak{m}_j) = 0 \bmod l$. The equivalence can be observed as follows. From the assumption $l(n+1) < p$, it implies $-l(n+1) \leq x' + \sum_{i=1}^{n} \hat{x}_i m_i \leq l(n+1)$ ($\because -l \leq x' \leq l, 0 \leq \hat{x}_i \leq l$). We have $-p < F(\mathfrak{m}_j) < p$, which implies if $F(\mathfrak{m}_j) = 0 \bmod p$ then $F(\mathfrak{m}_j) = 0 \bmod l$. Hence, $F(\mathfrak{m}_j) \neq 0 \bmod l$ implies $F(\mathfrak{m}_j) \neq 0 \bmod p$. Thus the former condition will be sufficient to ensure that a signature can be computed without aborting.

Output Calculation. If $\mathcal{B}$ does not abort, $\mathcal{A}$ will return a document $\mathfrak{m}^*$ with a forged signature σ^*. Encode $\mathfrak{m}^*$ as $m_1^* \ldots m_n^*$. If σ^* is not a sanitised signature and $\mathcal{A}$ wins the game, it means $\mathcal{A}$ can forge the Waters' signature [206] and hence $\mathcal{B}$ can solve the CDH problem.

If σ^* is a sanitised signature with $\sigma^* = (\sigma_1^*, \sigma_2^*, \{A_i^*, B_i^* | i \in st_{\mathfrak{m}^*}^S\})$. $\mathcal{B}$ aborts if the number of sanitised messages is not equal to N or the position of the sanitised message is not equal to the set S. $\mathcal{B}$ also aborts if $x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* \neq 0 \bmod p$.

By the verification equation, we can rewrite:

$$\sigma_1^* \cdot \prod_{i|m_i^*=\phi} B_i^{\hat{z}_i} = g_2^a \cdot {\sigma_2'}^{y' + \sum_{j|m_j^* \neq \phi} \hat{y}_j m_j^*} \cdot \prod_{i|m_i^*=\phi} A_i^{\hat{y}_i}$$

Therefore $\mathcal{B}$ can compute and output

$$Z = \sigma_1^* \cdot \prod_{i|m_i^*=\phi} B_i^{\hat{z}_i} \cdot {\sigma_2'}^{-y' - \sum_{j|m_j^* \neq \phi} \hat{y}_j m_j^*} \cdot \prod_{i|m_i^*=\phi} A_i^{-\hat{y}_i} = g_2^a = g^{ab},$$

as the answer to the CDH problem.

Probability Analysis. For the simulation to complete without aborting, we define the

events A_i, A_1^*, A_2^* such that the following conditions fulfilled:

$$\begin{aligned}
&A_i : F(\mathfrak{m}_j) \neq 0 \bmod l \qquad \text{where } j = 1, \ldots, q_s,\\
&A_1^* : x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod p,\\
&A_2^* : \text{the number of sanitised messages in } \mathfrak{m}^* = N,\\
&A_3^* : \text{the position of the sanitised message in } \mathfrak{m}^* = S.
\end{aligned}$$

The probability of $\mathcal{B}$ not aborting is

$$\Pr[\mathsf{not\ abort}] \geq \Pr\left[\Big(\bigwedge_{i=1}^{q_s} A_i\Big) \wedge A_1^* \wedge A_2^* \wedge A_3^*\right].$$

For the output calculation, as $0 \leq \hat{x}_i \leq l$, we have $-p < x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* < p$. Therefore $x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod p$ implies $x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod l$.

$$\begin{aligned}
\Pr[A_1^*] &= \Pr[x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod p \ \wedge x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod l]\\
&= \Pr[x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod l]\\
&\quad \cdot \Pr[x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod p \mid x' + \sum_{j|m_j^* \neq \phi} \hat{x}_j m_j^* = 0 \bmod l]\\
&\geq \frac{1}{l(n+1)}.
\end{aligned}$$

Notice that $\hat{x}_i$ is hidden in u_i by the random element $\hat{y}_i$. The adversary cannot make $\mathcal{B}$ abort by a chance better than by randomly choosing $st_{\mathfrak{m}^*}^S$.

The number of sanitised messages in $\mathfrak{m}^*$ can be any integer from 0 to $n-1$. If $\mathcal{B}$ guess the number N correctly, then the probability of choosing S correctly is $1/C_N^n$. We review the upper bound of the binomial coefficient in [196] as follows. Let m, N, p be positive integers, with $m > p \geq 1$ and $N \geq 1$. Then

$$\binom{mN}{pN} < \frac{1}{\sqrt{2\pi}} N^{-\frac{1}{2}} \frac{m^{mN+\frac{1}{2}}}{(m-p)^{(m-p)N+\frac{1}{2}} p^{pN+\frac{1}{2}}}.$$

By putting $m = 2$, $p = 1$, $n = 2N$ (we assume n is even), we have

$$\binom{n}{\frac{n}{2}} < \frac{1}{\sqrt{2\pi n}} 2^{n+1}.$$

Therefore the probability of choosing S correctly is at least $\sqrt{2\pi n} 2^{-n-1}$. However, in practice it is not common to sanitise half of the document in a signature (Otherwise, the

sanitiser may change the document such that it turns to a completely different meaning!). We may assume that the system only supports sanitising 10% of the document. Then the probability bound is improved to $\sqrt{18\pi n}9^{\frac{9n}{10}}10^{-n-1}$.

Notice that the event A_i is independent of the event A_1^*, A_2^* and A_3^*. Therefore

$$\begin{aligned}\Pr[\textsf{not abort}] &\geq \Pr[A_1^* \wedge A_2^* \wedge A_3^*] \Pr[\bigwedge_{i=1}^{q_s} A_i | A_1^* \wedge A_2^* \wedge A_3^*] \\ &\geq \Pr[A_1^*] \Pr[A_2^* \wedge A_3^*](1 - \sum_{i=1}^{q_s} \Pr[\neg A_i | A_1^* \wedge A_2^*]) \\ &\geq \frac{1}{l(n+1)} \Pr[A_3^* | A_2^*] \Pr[A_2^*](1 - \frac{q_s}{l}) \\ &= \frac{1}{l(n+1)} \frac{\sqrt{2\pi n}}{2^{n+1}} \frac{1}{n}(1 - \frac{q_s}{l}).\end{aligned}$$

Putting $l = 2q_s$, we have

$$\Pr[\textsf{not abort}] \geq \frac{\sqrt{\pi}}{4q_s(n+1)\sqrt{2n}2^n}.$$

If we assume that the system only supports sanitising 10% of the document, the probability $\Pr[\textsf{not abort}]$ is improved to $\frac{3\sqrt{\pi}9^{\frac{9n}{10}}}{20q_s(n+1)\sqrt{2n}10^n}$. If we take $n = 160$ as in Waters' signature, then the term $\frac{9^{\frac{9n}{10}}}{10^n} \approx 2^{75}$ (the typical value of q_s is 2^{30}).

Time Analysis. In the proof, $\mathcal{A}$ has to compute $O(1)$ multiplication and $O(1)$ exponentiation for every signing oracle query. □

Theorem 5. *Our sanitisable signature scheme is indistinguishable.*

Proof. We rewrite the challenge signature by letting $S = r + s$ and $T_i = r + t_i$, where r is the randomness used in Sign and s, t_i, s_i are the randomness used in Sanitise:

$$\sigma_1 = g_2^{\alpha} \cdot (u' \prod_{j|j \notin st^S_{\mathsf{M}^{b*}}} u_j^{m_j})^S \cdot \prod_{i|i \in st^S_{\mathsf{M}^{b*}}} u_i^{m_i T_i}, \quad \sigma_2 = g^S, \quad A_i = g^{m_i T_i} h_i^{s_i}, \quad B_i = u_i^{s_i}.$$

Note that the sanitzed messages $\{m_i | i \in st^S_{\mathsf{M}^{b*}}\}$ are always hidden by the random numbers T_i and T_i are uniformly distributed in $\mathbb{Z}_p$. On the other hand, the set $\{m_j | j \notin st^S_{\mathsf{M}^{0*}}\}$ is the same as the set $\{m_j | j \notin st^S_{\mathsf{M}^{1*}}\}$. Therefore our scheme is indistinguishable in an information theoretic sense. □

Comparison

The comparison of different sanitisable signature schemes are summarised in Table 4.2. Different types of state controllability, sanitised message, designated sanitiser and

Table 4.2: Comparison of sanitisable signature schemes (refer to section 4.2.1 for the meaning of the number). Trans stands for transparency.

Scheme	State	Message	Sanitiser	Trans.	Security	Model
[197]	2	2	1	1	RSA	ROM
[123]	1	2	1	1	underlying signature	standard
[158]	1	2	1	1	underlying signature	standard
[7]	2	4	2	2	underlying signature and chameleon hash	standard
[157]	3	2	1	1	underlying signature and commitment	standard
[200]	3	2	1	1	co-GDH	ROM
[131]	2	4	2	3	-	-
[156]	3	1	1	2	CDH	ROM
[121]	1	4	1	1	co-GDH	ROM
[56]	1*	3	1*	1	strong RSA + ?	standard
[109]	3	2	1	1	underlying signature, commitment and pseudorandom generator	standard
this section	1	2	1	1	CDH	standard

transparency are explained in Section 4.2.1. The security of some schemes rely on the security of the underlying signature, hash function or commitment scheme used. There is no security theorem for [131]. The scheme of Chang et al. [56] must be used with other sanitisable signature scheme (with unordered documents). Therefore its properties and security may change depending on the underlying scheme used.

For efficiency, our signature has one $\mathbb{G}_1$ and one $\mathbb{G}_2$ elements when it is not sanitised. This is efficient even when compared to standard signature scheme, when we take 170-bit for pairings. Each time when it is sanitised, it will generates extra 1 $\mathbb{G}_1$ element and 1 $\mathbb{G}_2$ element. The unforgeability suffers from an exponential reduction loss. The loss is reduced if we restrict the percentage of the document that can be sanitised. However our scheme is the first scheme proven secure in the standard model, without resolving to the underlying signature.

Finally, we give a direct comparison on the efficiency of our scheme with the scheme of Johnson *et al.* [123] and Miyazaki *et al.* [158]. It is because our scheme has the same property with these two schemes. We instantiate these two schemes using the Waters signature [206]. As a result, the security of these two schemes are also reduced to the CDH assumption. The size of the system parameters, public keys and private keys are similar. We compare the size of the signature before and after sanitising in Table 4.3.

Table 4.3: Comparison of the efficiency of three sanitisable signature schemes. Suppose the original document has length n, which consists of t segments, and the hash of the each segment has length h. Assume s segments is sanitised and each segment has length at most ℓ. In our scheme, the length of each segment is always equal to 1.

Scheme	$\lvert\sigma\rvert$ before sanitise	$\lvert\sigma\rvert$ after sanitise
[123]	$1\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + th$	$1\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + th + O(sth \lg(n\ell))$
[158]	$t\ \mathbb{G}_1 + t\ \mathbb{G}_2$	$t\ \mathbb{G}_1 + t\ \mathbb{G}_2 + sh$
this section	$1\ \mathbb{G}_1 + 1\ \mathbb{G}_2$	$(1+s)\ \mathbb{G}_1 + (1+s)\ \mathbb{G}_2$

We can see that our signature is the shortest before sanitisation. After sanitisation, the scheme of Johnson *et al.* [123] has variable signature size, which depends on the change of the tree structure of the sanitised document.

4.3 Concinnous Signatures

In daily commercial operations, it usually involves the exchange of one item for another. In the ancient world, the bartering system allows *goods or services* directly exchanged for other *goods or services*. In the modern world, the customers *pay* for the *goods or services* they purchase. No matter when, people want to ensure that the exchange is *fair*. It means that at end of the exchange, either each player receives the item it expects, or neither player receives any additional information about the other's item. The fair exchange protocols in cryptography are reviewed in §3.4.3.

In electronic commerce, fair exchange is not easy to achieve since we do not physically exchange the item nor physically check if the item is what we expected. Consider the case that Alice wants to buy a song from an online music store. How can the store obtain the electronic payment and how can Alice download the song at the same time? If Alice pays for the song first, what happens if the downloaded file is corrupted, or the store simply refuses Alice to download the song? A simple solution to the fair exchange problem is the introduction of a *trusted third party*. For example, the trusted third party can send the song to Alice if she cannot get the song after payment. Pagnia and Gärtner [170] showed that, under certain assumptions, fair exchange is impossible without a trusted third party in the system model proposed by Asokan [5]. As a result, there is a line of research called "*optimistic fair exchange*", where the trusted third party is only involved if there is any dispute.

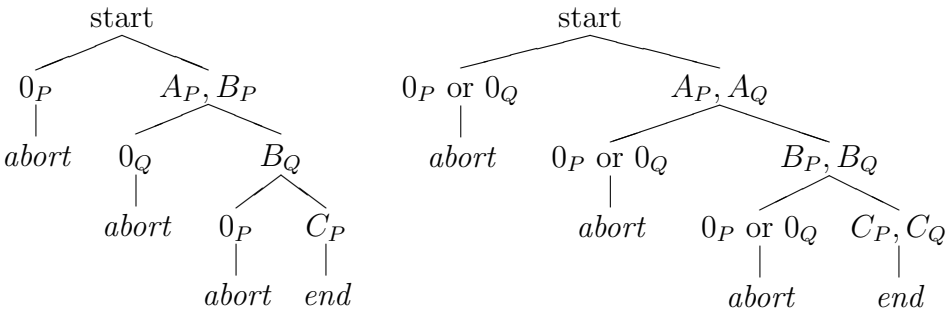
Fair exchange of digital signatures has important applications in electronic commerce. For examples, two companies want to digitally sign a contract, and the signed terms of agreement are exchanged. There are a number of fair exchange of digital

signatures schemes in the literature [81, 6, 209]. They rely on an online or offline trusted third party. However, a trusted third party may not be desirable in electronic commerce. In the example of contract signing, the contract may contain confidential information that they do not want to disclose to any third party when they fail to make the deal. In this case, *concurrent signatures* [62] can provide a certain degree of fairness without any trusted third party. The initiating company A sends a *keystone fix* and a *partial signature* to the company B. After checking their validity, the company B returns a partial signature to the company A. Finally, the company A releases a *keystone*, which allow both partial signatures to be verified by everyone at the same time. Without the keystone, no third party can verify the validity of the partial signatures. *Concurrent signatures* provide fairness in the sense that either both signatures can be verified or both signatures cannot be verified by third party. However, this protocol may not be fair from the company B's point of view, since only the company A has the power to control whether releasing the keystone or not, and the time of releasing the keystone. Therefore, it is important to consider fairness in terms of the control of keystone. Details about concurrent signatures can be found in §3.4.3.

Fair Exchange of Signatures without Trusted Third Party. Pagnia and Gärtner [170] proved the impossibility of fair exchange without a trusted third party. We first describe why concurrent signatures and our proposed improvement allow fair exchange of signatures without trusted third party. Table 3.1 provides the fair exchange system considered in [170]. The two parties P and Q want to exchange their items i_P and i_Q, respectively. Pagnia and Gärtner [170] proved the impossibility of fair exchange without a trusted third party in this system. However when we consider the exchange of signatures (i.e. i_P and i_Q are signatures), this system regards the signatures i_P and i_Q as the input. The signatures must be generated before the start of the exchange protocol. Therefore the impossibility result in [170] cannot be applied to protocols like concurrent signatures, where the signature generation requires the interaction of the other party.

Fairness of Concurrent Signatures. One drawback of the concurrent signatures is that the party who initiates the exchange protocol can control whether releasing the keystone or not, and the time of releasing the keystone. The other party does not possess this power, and it implies some unfairness which is not included in the current fairness model. We review the concurrent signatures as follows. Suppose P initiates the exchange protocol with Q. We define the following protocols:

Table 4.4: Flow diagram of concurrent signatures (left) and our proposal (right). The subscript of the protocol $A, B, C, 0$ denotes the running party.



- Protocol A: Generate a keystone and send a keystone fix to the other party.
- Protocol B: Generate a partial signature and send the signature to the other party.
- Protocol C: Release the keystone.
- Protocol 0: Abort.

We give the flow diagram of concurrent signatures and our proposed improvement in Table 4.4. For concurrent signatures, the initiating party P runs the protocol A and B. After that, the other party Q can either abort or run the protocol B. Finally, the initiating party P can either abort or run the protocol C. We can see that in concurrent signatures, the initiating party P can control whether the exchange protocol aborts or successfully ends during the final step[8].

We propose an improvement of concurrent signatures such that both parties P and Q can abort at any step of the protocol. Firstly, both parties P and Q run the protocol A independently. We do not care who finish running the protocol first. After that, P and Q run the protocol B independently. Either party can decide to abort. If both parties do not abort, P and Q run the protocol C finally. The symmetric structure of our proposal ensure that both participating parties are treated equally and has the same power within the protocol. If the exchange protocol successfully ends, both parties use the same computational power.

Our Proposal. We would like to propose a fair exchange protocol of signatures where both parties can participate in the release of the final keystone. Our proposal in Table 4.4 is similar to running two concurrent signature protocols by P and Q simultaneously.

[8]We note that the fair exchange signatures in [144] also suffer from this problem

However, we have to overcome two major problems in order to make the scheme works. Firstly, we have to bind two keystones together, such that no adversary can use P's partial signature in some concurrent signature σ_A and use Q's partial signature in some concurrent signature σ_B. We overcome this problem by using a Diffie-Hellman exchange protocol for their keystones. Denote x as the private key, k as the keystone and s as the keystone fix. We use the subscript P and Q to denote the participating parties. When computing the partial signature, there is a computation of the form

$$t = H(s_P, s_Q, g^{x_Q k_P}, g^{x_P k_Q}), \tag{4.2}$$

where H is a hash function. As a result, both keystones are bound to the partial signature.

Secondly, in most existing concurrent signature schemes, the protocol B_Q uses the output of the protocol B_P (which is the partial signature of the initiating party P) as its input. It is used to ensure the ambiguity property of concurrent signatures. In our proposed improvement, both parties P and Q run the protocol B at the same time. We overcome this problem by formulating t as a deterministic function of the keystones from P and Q, and both parties can effectively compute this t as shown in equation (4.2).

We name our proposed improvement as "*Concinnous Signatures*", as our proposal is an elegant arrangement of two concurrent signatures as a whole, and it brings a fair and harmony exchange of digital signatures to both participating parties. In concinnous signatures, both parties can terminate at any step of the protocol (where in concurrent signature, only the initiating party has the advantage to terminate just before the last step). If the exchange protocol successfully ends, both parties use the same computational power. These two types of fairness are not considered in concurrent signatures.

4.3.1 Security Models of Concinnous Signatures

We introduce the notion of concinnous signatures. The security of the concinnous signatures includes unforgeability, ambiguity and fairness.

Syntax

A concinnous signature scheme consists of the following six algorithms.

Table 4.5: Concinnous signature protocol

	Alice			Bob
1	Run $(K_A, F_A) \leftarrow \mathbf{KGen}(m_A, y_A, y_B)$. Send F_A to Bob.	$\longrightarrow$	$\longleftarrow$	Run $(K_B, F_B) \leftarrow \mathbf{KGen}(m_B, y_B, y_A)$. Send F_B to Alice.
2	After receiving F_B from Bob, run $\sigma_A \leftarrow \mathbf{ASign}(m_A, (y_A, y_B), x_A, F_A, F_B)$. Send σ_A to Bob.	$\longrightarrow$	$\longleftarrow$	After receiving F_A from Alice, run $\sigma_B \leftarrow \mathbf{ASign}(m_B, (y_A, y_B), x_B, F_B, F_A)$. Send σ_B to Alice.
3	After receiving σ_B from Bob, run $\mathbf{AVerify}(\sigma_B, m_B, (y_A, y_B), F_B, F_A, x_A, K_A)$. If it outputs accept, reveal K_A.	$\longrightarrow$	$\longleftarrow$	After receiving σ_A from Alice, run $\mathbf{AVerify}(\sigma_A, m_A, (y_A, y_B), F_A, F_B, x_B, K_B)$. If it outputs accept, reveal K_B.

- **Setup**(1^λ). On input a security parameter λ, it outputs the system parameters param, including the keystone space $\mathcal{K}$, the keystone fix space $\mathcal{F}$ and the message space $\mathcal{M}$.

- **Key-Gen**(param). On input the system parameters param, it outputs (x, y), where x is the private key and y is the public key.

- **KGen**(m_A, y_A, y_B). On input a message m_A, the sender public key y_A and the recipient public key y_B, it outputs a keystone $K_A \in \mathcal{K}$ and a keystone fix $F_A \in \mathcal{F}$.

- **ASign**$(m_A, (y_A, y_B), x_A, F_A, F_B)$. On input a message $m_A \in \mathcal{M}$, the public key set (y_A, y_B), the sender secret key x_A, the sender's keystone fix F_A and the recipient's keystone fix F_B, it outputs an anonymous signature σ_A to Bob.

- **AVerify**$(\sigma_B, m_B, (y_A, y_B), F_B, F_A, x_A, K_A)$. On input an anonymous signature σ_B, a message $m_B \in \mathcal{M}$, the public key set (y_A, y_B), the recipient's keystone fix F_B, the sender's keystone fix F_A, the sender's secret key x_A and the sender's keystone K_A, it outputs accept or reject.

- **Verify**$(\sigma_A, \sigma_B, m_A, m_B, (y_A, y_B), K_A, K_B, F_A, F_B)$. On input the signatures σ_A, σ_B, the messages m_A, m_B, the public key set (y_A, y_B), the keystones K_A, K_B and the keystone fixes F_A, F_B, it outputs accept or reject.

A concinnous signature protocol between two parties Alice and Bob works as follows. Suppose after the **Setup** and **Key-Gen** algorithms, Alice's public and private keys are y_A and x_A, and Bob's public and private keys are y_B and x_B. Alice will sign on a message m_A and Bob will sign on a message m_B.

Finally, everyone with the signatures σ_A, σ_B, the messages m_A, m_B, the public key set (y_A, y_B) and the keystones K_A, K_B can run the **Verify** algorithm to check the validity of the concinnous concurrent signature.

We say that a concinnous concurrent signature scheme is *correct* if:

- every anonymous signature σ properly generated by **ASign** will be accepted by **AVerify**;
- every tuple $(\sigma_A, \sigma_B, m_A, m_B, (y_A, y_B), K_A, K_B, F_A, F_B)$ properly generated by **KGen** and **ASign** will be accepted by **Verify**.

Unforgeability

We define the existential unforgeability of a concinnous signature scheme under a chosen message attack using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Firstly, $\mathcal{A}$ is given the public parameters and a set of n public keys. $\mathcal{A}$ is allowed to get $n-1$ secret keys by the ExtractOrcale. $\mathcal{A}$ can query the ASignOracle such that $\mathcal{C}$ will interact with $\mathcal{A}$ until step 2 of Table 4.3.1. $\mathcal{A}$ can query the KRevealOracle such that $\mathcal{C}$ will interact with $\mathcal{A}$ until step 3 of Table 4.3.1. Note that step 1 of Table 4.3.1 can be computed by anyone since step 1 does not require any secret key. $\mathcal{A}$ wins the unforgeability game if he can forge a valid anonymous signature with a valid keystone.

The game is defined as follows.

1. Setup. The challenger $\mathcal{C}$ runs $\mathsf{param} \leftarrow \mathbf{Setup}(1^\lambda)$. $\mathcal{C}$ also runs $(x_i, y_i) \leftarrow \mathbf{KeyGen}(\mathsf{param})$ for $i = 1, \ldots, n$, where $n-1$ is the maximum number of ExtractOrcale query defined below. Denote $\mathcal{U} = \{y_1, \ldots, y_n\}$ as the set of public keys. $\mathcal{C}$ gives param and $\mathcal{U}$ to the adversary $\mathcal{A}$.

2. Query. The adversary $\mathcal{A}$ can make the following queries to $\mathcal{C}$:

   - ASignOracle: On input (m_A, y_A, y_B) (where $y_A, y_B \in \mathcal{U}$), the oracle firstly returns a keystone fix F_A, where $(K_A, F_A) \leftarrow \mathbf{KGen}(m_A, y_A, y_B)$. Then $\mathcal{A}$ gives another keystone fix F_B to the oracle[9]. The oracle returns the anonymous signature $\sigma_A \leftarrow \mathbf{ASign}(m_A, (y_A, y_B), x_A, F_A, F_B)$.
   - KRevealOracle: On input (m_A, y_A, y_B) (where $y_A, y_B \in \mathcal{U}$), the oracle firstly returns a keystone fix F_A, where $(K_A, F_A) \leftarrow \mathbf{KGen}(m_A, y_A, y_B)$. Then

[9]The adversary is allowed to give F_B to the oracle before it receives F_A. It reflects that F_A or F_B may be generated first in the actual protocol.

$\mathcal{A}$ gives another keystone fix F_B to the oracle[10]. The oracle returns the anonymous signature $\sigma_A \leftarrow \mathbf{ASign}(m_A, (y_A, y_B), x_A, F_A, F_B)$ to $\mathcal{A}$. Then $\mathcal{A}$ gives another anonymous signature σ_B and a message m_B to the oracle[11]. Finally the oracle outputs the keystone K_A if $\mathsf{accept} \leftarrow \mathbf{AVerify}(\sigma_B, m_B, (y_A, y_B), F_B, F_A, x_A, K_A)$.

- ExtractOracle: On input a public key $y_i \in \mathcal{U}$, the oracle returns the corresponding private key x_i.

3. Output. Finally, $\mathcal{A}$ outputs a tuple $(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, y_A^*, y_B^*, K_A^*, K_B^*, F_A^*, F_B^*)$, where $y_A^*, y_B^* \in \mathcal{U}$.

The adversary $\mathcal{A}$ wins if $\mathsf{accept} \leftarrow \mathbf{Verify}(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, (y_A^*, y_B^*), K_A^*, K_B^*)$. y_A^* has never been queried to the ExtractOracle; and (m_A^*, y_A^*, y_B^*) has never been queried to the KRevealOracle such that the oracle returns the keystone K_A^*[12]. We say that the advantage of $\mathcal{A}$ is its probability of winning the above game.

A concinnous signature scheme is $(\epsilon, t, q_s, q_r, q_e)$-unforgeable if there is no t time adversary that has advantage at least ϵ with q_s, q_r, and q_e queries to the ASignOracle, KRevealOracle and ExtractOracle, respectively.

Ambiguity

We define the perfect ambiguity of a concinnous signature scheme under a chosen message attack using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Firstly, $\mathcal{A}$ is given the public parameters and two pairs of public and private keys of signer A and B. $\mathcal{A}$ can query the ASignOracle and KRevealOracle defined in the unforgeability game. In the challenge phase, $\mathcal{C}$ tries to run the concinnous signature scheme to generate two anonymous signatures σ_A and σ_B such that either:

- σ_A and σ_B are generated by user A,
- σ_A and σ_B are generated by user B,
- σ_A and σ_B are generated by user A and user B respectively, or
- σ_A and σ_B are generated by user B and user A respectively.

[10]The adversary is allowed to give F_B to the oracle before it receives F_A.

[11]The adversary is allowed to give σ_B and m_B to the oracle before it receives σ_A.

[12]It captures the model for accountability of concurrent signatures [141], since the adversary can always use the same keystone K_B^* to query the oracles with different messages.

$\mathcal{A}$ wins if he can determine which is the case.

The ambiguity game is defined as follows.

1. Setup. The adversary $\mathcal{A}$ gives param and two pairs of public and private keys $(y_A^*, x_A^*), (y_B^*, x_B^*)$ to the challenger $\mathcal{C}$.

2. Phase 1. The adversary $\mathcal{A}$ can make queries to the ASignOracle and KRevealOracle, where the input public keys must be either y_A^* or y_B^*.

3. Challenge. At some point, the adversary $\mathcal{A}$ sends two messages m_A^*, m_B^* to $\mathcal{C}$. The challenger $\mathcal{C}$ runs $(K_A^*, F_A^*) \leftarrow \mathbf{KGen}(m_A^*, y_A^*, y_B^*)$ and $(K_B^*, F_B^*) \leftarrow \mathbf{KGen}(m_B^*, y_A^*, y_B^*)$. $\mathcal{C}$ randomly picks $b \in \{1, 2, 3, 4\}$ and:

   - If $b = 1$, $\mathcal{C}$ calculates $\sigma_A^* \leftarrow \mathbf{ASign}(m_A^*, (y_A^*, y_B^*), x_A^*, F_A^*, F_B^*)$ and $\sigma_B^* \leftarrow \mathbf{ASign}(m_B^*, (y_A^*, y_B^*), x_A^*, F_B^*, F_A^*)$.
   - If $b = 2$, $\mathcal{C}$ calculates $\sigma_A^* \leftarrow \mathbf{ASign}(m_A^*, (y_A^*, y_B^*), x_B^*, F_A^*, F_B^*)$ and $\sigma_B^* \leftarrow \mathbf{ASign}(m_B^*, (y_A^*, y_B^*), x_B^*, F_B^*, F_A^*)$.
   - If $b = 3$, $\mathcal{C}$ calculates $\sigma_A^* \leftarrow \mathbf{ASign}(m_A^*, (y_A^*, y_B^*), x_A^*, F_A^*, F_B^*)$ and $\sigma_B^* \leftarrow \mathbf{ASign}(m_B^*, (y_A^*, y_B^*), x_B^*, F_B^*, F_A^*)$.
   - If $b = 4$, $\mathcal{C}$ calculates $\sigma_A^* \leftarrow \mathbf{ASign}(m_A^*, (y_A^*, y_B^*), x_B^*, F_A^*, F_B^*)$ and $\sigma_B^* \leftarrow \mathbf{ASign}(m_B^*, (y_A^*, y_B^*), x_A^*, F_B^*, F_A^*)$.

   $\mathcal{C}$ returns $(\sigma_A^*, \sigma_B^*, F_A^*, F_B^*, K_A^*)$ to $\mathcal{A}$.

4. Phase 2. The adversary $\mathcal{A}$ can make queries to the KGenOracle, ASignOracle and KRevealOracle.

5. Output. The adversary $\mathcal{A}$ returns a value $b' \in \{1, 2, 3, 4\}$ as its guess for b.

The adversary $\mathcal{A}$ wins the above game if $b = b'$. We say that the advantage of $\mathcal{A}$ is its probability of winning the above game minus 1/4.

A concinnous signature scheme is $(\epsilon,\ t,\ q_s,\ q_r)$-ambiguous if there is no t time adversary that has advantage at least ϵ with q_s and q_r queries to the ASignOracle and KRevealOracle, respectively.

Fairness

We define the fairness of a concinnous signature scheme under a chosen message attack using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Firstly, $\mathcal{A}$ is

given the public parameters and two pairs of public and private keys of signer A and B. $\mathcal{A}$ can query the ASignOracle and KRevealOracle defined in the unforgeability game. Finally, $\mathcal{A}$, acting as user A, runs the concinnous signature scheme with $\mathcal{C}$, acting as user B. $\mathcal{A}$ wins if:

- $\mathcal{C}$ finish running step 2 of Table 4.3.1, and $\mathcal{A}$ can output a valid keystone of user B; or
- $\mathcal{C}$ finish running step 3 of Table 4.3.1, and $\mathcal{A}$ can output a valid concinnous signature such that the keystone fix of user B is the same as that of step 1 of Table 4.3.1, but the keystone fix of user A is different.

The fairness game is defined as follows.

1. Setup. The adversary $\mathcal{A}$ gives param and two pairs of public and private keys $(y_A^*, x_A^*), (y_B^*, x_B^*)$ to the challenger $\mathcal{C}$.
2. Query. The adversary $\mathcal{A}$ can make queries to the ASignOracle and KRevealOracle, where the input public keys must be either y_A^* or y_B^*.
3. Output. Finally, $\mathcal{A}$ chooses to act as either user A or user B. Then $\mathcal{A}$ informs $\mathcal{C}$ to act as the other party. Without loss of generality, assume $\mathcal{A}$ chooses to be user A. Then $\mathcal{A}$ and $\mathcal{C}$ interacts as follows (if $\mathcal{A}$ chooses to be user $\mathcal{B}$, then all subscripts A are changed to B, and vice versa).

   $\mathcal{A}$ sends a message m_B^* to $\mathcal{C}$. $\mathcal{C}$ returns F_B^* to $\mathcal{A}$, where $(K^*, F_B^*) \leftarrow \mathbf{KGen}(m_B^*, y_B^*, y_A^*)$. $\mathcal{A}$ sends a keystone fix F_A^* to $\mathcal{C}$. Then $\mathcal{C}$ returns σ_B^*, where $\sigma_B^* \leftarrow \mathbf{Asign}(m_B^*, (y_A^*, y_B^*), x_B^*, F_B^*, F_A^*)$. $\mathcal{A}$ wins by one of the following ways:

   - $\mathcal{A}$ outputs a tuple $(\sigma_A^*, m_A^*, K_A^*, K_B^*)$. $\mathcal{A}$ wins if accept $\leftarrow \mathbf{Verify}(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, (y_A^*, y_B^*), K_A^*, K_B^*, F_A^*, F_B^*)$.
   - $\mathcal{A}$ returns a tuple (σ_A^*, m_A^*) to $\mathcal{C}$, where accept $\leftarrow \mathbf{AVerify}(\sigma_A^*, m_A^*, (y_A^*, y_B^*), F_B^*, F_A^*, x_B^*, K^*)$. After that $\mathcal{C}$ returns K^*, the keystone for the keystone fix F_B^*. Finally $\mathcal{A}$ outputs a tuple $(\sigma'_A, m'_A, K'_A, F'_A)$. $\mathcal{A}$ wins if accept $\leftarrow \mathbf{Verify}(\sigma'_A, \sigma_B^*, m'_A, m_B^*, (y_A^*, y_B^*), K'_A, K^*, F'_A, F_B^*)$, and $F'_A \neq F_A^*$.

We say that the advantage of $\mathcal{A}$ is its probability of winning the above game.

A concinnous signature scheme is (ϵ, t, q_s, q_r)-fair secure if there is no t time adversary that has advantage at least ϵ with q_s and q_r queries to the ASignOracle and KRevealOracle respectively.

4.3.2 Concinnous Signature Scheme

We propose a concrete construction of our concinnous signatures and give some security proofs in this section. We first point out two major difference between the concinnous signatures and the concurrent signatures. Firstly, in concurrent signatures, only the initiating party has the power to control the release of the keystone. On the other hand, in concinnous signatures, both participating parties have the power to control the release of the keystone.

Secondly, in most perfect concurrent signature schemes, there is no clear reason of why certain computations are included in the *concurrent signature protocols*, but not in the *concurrent signature algorithms* (e.g. **ASign**, **AVerify**). For example, a signature from proof of knowledge is needed in the perfect concurrent signature protocols of [199], but it is not included in any perfect concurrent signature algorithms. There is no formal method to differentiate *concurrent signature algorithms* from *concurrent signature protocols*. In concinnous signatures, we define in a way that all computations are mentioned in the *concinnous signature algorithms*. The *concinnous signature protocols* are only used to describe the sequence of running the algorithms and how the participating parties interact with each other. It is much closer to the original definitions for concurrent signatures in [62].

As a result, our *concinnous signature algorithms* has some differences with the *concurrent signature algorithms*. For example, our **AVerify** algorithm takes two extra inputs: the sender's secret key and the sender's keystone. It prevents the man-in-the-middle attack. Our **Verify** algorithm uses a direct definition instead of reusing the **AVerify** notation as in the *concurrent signature algorithms*. It is because the **AVerify** require the use of sender's secret key, which is not known by any verifier.

Our Concinnous Signature Algorithms

We introduce our concinnous signature algorithms as follows.

- **Setup**. Let λ be a security parameter. Let p be a λ-bit prime number and q is a prime such that $q|(p-1)$. Let g be a generator of order q. Let $\mathcal{K} = \mathbb{Z}_q$ be the keystone space, $\mathcal{F} = \mathbb{Z}_q \times \mathbb{Z}_p$ be the keystone fix space and $\mathcal{M}$ be the message space. Let $H : \mathcal{M} \times \mathbb{Z}_p \to \mathbb{Z}_q$, $H_1 : \mathcal{K} \times \mathcal{M} \times \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_q$ and $H_2 : \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_q$ be some collision resistant hash functions. It outputs $\mathsf{param} = \{\mathcal{K}, \mathcal{F}, \mathcal{M}, p, q, g, H, H_1, H_2\}$.

- **Key-Gen**. On input param, it picks a random private key $x \in \mathbb{Z}_q$ and calculates the public key $y = g^x \mod p$.

- **KGen**. On input a message m_A, the sender public key y_A and the recipient public key y_B, it picks a random keystone k_A from the keystone space $\mathbb{Z}_q$. Then it calculates the keystone fix (s_A, R_A), where:

$$\begin{aligned} s_A &= H_1(k_A, m_A, y_A, y_B), \\ R_A &= g^{k_A} \mod p. \end{aligned}$$

- **ASign**. On input a message m_A, the public key set (y_A, y_B), the sender secret key x_A, the sender's keystone fix $F_A = (s_A, R_A)$ and the recipient's keystone fix $F_B = (s_B, R_B)$, it picks a random number $\alpha_A \in \mathbb{Z}_q$ and calculates:

$$\begin{aligned} t_A &= H_2(s_A, s_B, y_B^{k_A} \mod p, R_B^{x_A} \mod p), \\ c_A &= H(m_A, g^{\alpha_A} y_B^{t_A} \mod p), \\ z_A &= (\alpha_A - c_A) x_A^{-1} \mod q. \end{aligned}$$

  It outputs the signature $\sigma_A = (c_A, z_A, t_A)$.

- **AVerify**. On input an anonymous signature $\sigma_B = (c_B, t_B, z_B)$, a message $m_B \in \mathcal{M}$, the public key set (y_A, y_B), the recipient's keystone fix $F_B = (s_B, R_B)$, the sender's keystone fix $F_A = (s_A, R_A)$, the sender's secret key x_A and the sender's keystone k_A, it outputs accept if

$$\begin{aligned} c_B &= H(m_B, g^{c_B} y_A^{t_B} y_B^{z_B}), \\ t_B &= H_2(s_B, s_A, R_B^{x_A} \mod p, y_B^{k_A} \mod p). \end{aligned}$$

  Otherwise, it outputs reject.

- **Verify**. On input the signatures $\sigma_A = (c_A, t_A, z_A), \sigma_B = (c_B, t_B, z_B)$, the messages m_A, m_B, the public key set (y_A, y_B), the keystones k_A, k_B and the keystone fixes $F_A = (s_A, R_A), F_B = (s_B, R_B)$, it outputs accept if all of the following holds:

$$\begin{aligned} s_A &= H_1(k_A, m_A, y_A, y_B), \\ s_B &= H_1(k_B, m_B, y_B, y_A), \\ c_A &= H(m_A, g^{c_A} y_B^{t_A} y_A^{z_A}), \\ c_B &= H(m_B, g^{c_B} y_A^{t_B} y_B^{z_B}), \\ t_A &= H_2(s_A, s_B, y_B^{k_A} \mod p, y_A^{k_B} \mod p), \\ t_B &= H_2(s_B, s_A, y_A^{k_B} \mod p, y_B^{k_A} \mod p), \end{aligned}$$

Otherwise, it outputs reject.

Security

The correctness of the scheme is straightforward.

Theorem 6. *Our concinnous signature scheme is* $(\epsilon, t, q_s, q_r, q_e)$*-unforgeable if the* (ϵ', t')*-discrete logarithm assumption holds in the random oracle model, where*

$$\epsilon' \geq \frac{1}{q_e + 1}(\epsilon - \frac{1}{p})^2, \quad t' = t + O\Big((q_s + q_r + q_e)\rho\Big),$$

where ρ *is the time for an exponentiation in* $\mathbb{Z}_p$*.*

Proof. Assume that there is an adversary $\mathcal{A}$ who can break the unforgeability, we can construct a simulator $\mathcal{B}$ to solve the discrete logarithm (DL) problem. $\mathcal{B}$ is given the DL problem instance (g, g^α). $\mathcal{B}$ is asked to output α.

Setup. $\mathcal{B}$ honestly generates $\mathsf{param} = \{\mathcal{K}, \mathcal{F}, \mathcal{M}, p, q, g, H, H_1, H_2\}$ as in **Setup**. $\mathcal{B}$ picks a random index $j \in \{1, \ldots, n\}$ as his guess of the challenge public key y_j. Without loss of generality, assume $\mathcal{B}$ picks $j = n$. $\mathcal{B}$ picks some random $x_1, \ldots, x_{n-1} \in_R \mathbb{Z}_q$ and sets $x_n = \alpha$. $\mathcal{B}$ calculates $y_i = g^{x_i} \mod p$ for $i = 1, \ldots, n$. $\mathcal{B}$ gives param and $\mathcal{U} = \{y_1, \ldots, y_n\}$ to the adversary $\mathcal{A}$.

Oracle Simulation.

- ASignOracle. On input (m_A, y_A, y_B), $\mathcal{B}$ honestly runs **KGen** and **ASign** if $y_A = y_i \in \mathcal{U}$ for some $i = 1, \ldots, n-1$, since the corresponding secret key x_i is known.

  If $y_A = y_n$, $\mathcal{B}$ firstly returns the keystone fix $F_A = (s_A, R_A)$ where $(k_A, F_A) \leftarrow$ **KGen**(m_A, y_A, y_B). The adversary gives the keystone fix $F_B = (s_B, R_B)$ to $\mathcal{B}$. $\mathcal{B}$ searches the H_1-list for the tuple using the following algorithm:

  - Loop for $(i = 1, \ldots, |\mathsf{H}_1\text{list}|)$:
    - if the tuple $(k_i, \cdot, y_B, y_A, s_B)$ is in the H_1-list:
      - if $R_B = g^{k_i} \mod p$, $\mathcal{B}$ sets $k_B = k_i$ and exits this algorithm.
  - $\mathcal{B}$ picks a random $k_B \in \mathbb{Z}_q$ and exits this algorithm.

  $\mathcal{B}$ calculates $t_A = H_2(s_A, s_B, y_B^{k_A} \mod p, y_A^{k_B} \mod p)$. $\mathcal{B}$ picks some random $z_A, c_A \in \mathbb{Z}_q$ and calculates $T_A = g^{c_A} y_A^{z_A} y_B^{t_A} \mod p$. $\mathcal{B}$ puts the tuple (m_A, T_A, c_A) in the H-list ($\mathcal{B}$ declares failure and exits if $(m_A, T_A, \cdot)$ is already the H-list). Finally $\mathcal{B}$ returns (c_A, z_A, t_A) to $\mathcal{A}$ as the signature.

- KRevealOracle. $\mathcal{B}$ simulates as in the ASignOracle for the **KGen** and **ASign** algorithms. After that, $\mathcal{A}$ returns a signature $\sigma_B = (c_B, t_B, z_B)$ and a message m_B to $\mathcal{B}$. $\mathcal{B}$ checks if $c_B = H(m_B, g^{c_B} y_A^{t_B} y_B^{z_B})$ and $t_B = H_2(s_B, s_A, y_A^{k_B} \mod p, y_B^{k_A} \mod p)$. $\mathcal{B}$ outputs reject if they are not equal. Otherwise, $\mathcal{B}$ returns k_A.

- ExtractOracle. On input $y_i \in \mathcal{U}$, $\mathcal{B}$ returns x_i if $i = 1, \ldots, n-1$. If $y_i = y_n$, then $\mathcal{B}$ declares failure and exits.

- H-Oracle and H_1-Oracle: They are simulated as normal random oracles.

Output. Finally the adversary $\mathcal{A}$ outputs a tuple $(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, y_A^*, y_B^*, k_A^*, k_B^*, F_A^*, F_B^*)$. If $y_A^* \neq y_n$, then $\mathcal{B}$ declares failure and exits. If $(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, k_A^*, k_B^*)$ is a valid signature for y_A^*, y_B^*, then we denote $\sigma_A^* = (c_A^*, t_A^*, z_A^*)$, $F_A^* = (s_A^*, R_A^*)$ and we have:

$$
\begin{aligned}
s_A^* &= H_1(k_A^*, m_A^*, y_A^*, y_B^*), \\
s_B^* &= H_1(k_B^*, m_B^*, y_B^*, y_A^*), \\
t_A^* &= H_2(s_A^*, s_B^*, {y_B^*}^{k_A^*} \mod p, {y_A^*}^{k_B^*} \mod p), \\
c_A^* &= H(m_A^*, g^{c_A^*} {y_A^*}^{z_A^*} {y_B^*}^{t_A^*}).
\end{aligned}
$$

We have the following cases:

- If σ_A^* is the output from the ASignOracle:
  - If s_A^* is equal to the s_A in the same ASignOracle query, then it means that $\mathcal{A}$ can find the pre-image k_A^*, which breaks the one-wayness property of the hash function H_1.
  - If s_A^* is not equal to the s_A in the same ASignOracle query, then it means that $\mathcal{A}$ can find two different inputs to H_2 and obtains t_A^*, which breaks the collision-resistant property of the hash function H_2.

- If σ_A^* is the output from the KRevealOracle:
  - If s_A^* is equal to the s_A in the same KRevealOracle query, then it means that $\mathcal{A}$ can find two different inputs k_A^* and k_A and obtains s_A^*, which breaks the collision-resistant property of the hash function H_1.
  - If s_A^* is not equal to the s_A in the same KRevealOracle query, then it means that $\mathcal{A}$ can find two different inputs to H_2 and obtains t_A^*, which breaks the collision-resistant property of the hash function H_2.

- If σ_A^* is not the output from the ASignOracle and the KRevealOracle, $\mathcal{B}$ rewinds to the point where the query $(m_A^*, g^{c_A^*} {y_A^*}^{z_A^*} {y_B^*}^{t_A^*})$ is made to the H oracle. This time the H oracle returns c'_A instead and finally the adversary returns a different signature $\sigma_A^* = (c'_A, t'_A, z'_A)$. Therefore we have:

$$g^{c_A^*} {y_A^*}^{z_A^*} {y_B^*}^{t_A^*} = g^{c'_A} {y_A^*}^{z'_A} {y_B^*}^{t'_A} \mod p.$$

  It implies $c_A^* - c'_A = (z'_A - z_A^*)x_A^* + (t'_A - t_A^*)x_B^* \mod q$. We consider two cases:

  1. $z'_A \neq z_A^*$. Since $y_A^* = y_n$, $\mathcal{B}$ can obtain $\alpha = x_A^* = [(c_A^* - c'_A) + (t_A^* - t'_A)x_B^*]/(z'_A - z_A^*) \mod q$ as the solution to the DL problem.
  2. $z'_A = z_A^*$. Since $c_A^* \neq c'_A$, it implies that $t'_A \neq t_A^*$. Also we have:

$$c_A^* - c'_A = (t'_A - t_A^*)x_B^* \mod q.$$

     Notice that x_B^*, c_A^* and c'_A are randomly chosen by $\mathcal{B}$ and t_A^* is given to $\mathcal{B}$ before rewinding. After rewinding, the adversary $\mathcal{A}$ has to find a t'_A such that it satisfies the above equation and also $t'_A = H_2(s'_A, s'_B, {y_B^*}^{k'_A}, {y_A^*}^{k'_B})$. Since H_2 is also a random oracle, the probability of $\mathcal{A}$ finding such s'_A, s'_B, k'_A, k'_B is negligible. Therefore the probability of $z'_A = z_A^*$ is negligible.

Probability Analysis. For the simulation to complete without aborting, we require that $\mathcal{B}$ correctly extract α at the end of the game. By the Reset Lemma [19], it happens with probability at least $\frac{1}{n}(\epsilon - \frac{1}{p})^2$. We have

$$\epsilon' \geq \frac{1}{q_e + 1}(\epsilon - \frac{1}{p})^2.$$

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{Z}_p$ element for each ASignOracle and KRevealOracle query. There are n exponentiations in the **Setup** phase and $q_e = n - 1$. The time complexity of $\mathcal{B}$ is

$$t + O\Big((q_s + q_r + q_e)\rho\Big).$$

□

Theorem 7. *Our concinnous signature scheme is (ϵ, t, q_s, q_r)-ambiguous if the (ϵ, t')-discrete logarithm assumption holds in the random oracle model, where*

$$t' = t + O\Big((q_s + q_r + q_{H_1})\rho\Big),$$

where ρ is the time for an exponentiation in $\mathbb{Z}_p$, and q_{H_1} is the number of query to the H_1 oracle.

Proof. Assume that there is an adversary $\mathcal{A}$ who can break the ambiguity, we can construct a simulator $\mathcal{B}$ to solve the discrete logarithm (DL) problem. $\mathcal{B}$ is given the DL problem instance (g, g^{α}). $\mathcal{B}$ is asked to output α.

Setup. $\mathcal{B}$ receives $\mathsf{param} = \{\mathcal{K}, \mathcal{F}, \mathcal{M}, p, q, g, H, H_1, H_2\}$ and the key pairs (y_A^*, x_A^*) and (y_B^*, x_B^*) from the adversary $\mathcal{A}$.

Oracle Simulation. Since the simulator $\mathcal{B}$ knows the user secret keys, he can simulate all oracle queries correctly.

Challenge. At some point, $\mathcal{A}$ sends two messages m_A^*, m_B^* to $\mathcal{B}$. $\mathcal{B}$ runs $(k_A^*, (s_A^*, R_A^*)) \leftarrow \mathbf{KGen}(m_A^*,\ y_A^*, y_B^*)$. $\mathcal{B}$ randomly picks $s_B^* \in \mathbb{Z}_q$ and sets $R_B^* = g^{\alpha}$. It implicitly sets $k_B^* = \alpha$. $\mathcal{B}$ randomly picks $b \in \{1, 2, 3, 4\}$ and:

- If $b = 1$, $\mathcal{B}$ randomly picks $t_A^*, \alpha_A^* \in \mathbb{Z}_q$ and calculates $c_A^* = H(m_A^*, g^{\alpha_A^*} {y_B^*}^{t_A^*} \bmod p)$ and $z_A^* = (\alpha_A^* - c_A^*){x_A^*}^{-1} \bmod q$. Denote $\sigma_A^* = (c_A^*, z_A^*, t_A^*)$. $\mathcal{B}$ randomly picks $z_B^*, \alpha_B^* \in \mathbb{Z}_q$ and calculates $c_B^* = H(m_B^*, g^{\alpha_B^*} {y_B^*}^{z_B^*} \bmod p)$ and $t_B^* = (\alpha_B^* - c_B^*){x_A^*}^{-1} \bmod q$. Denote $\sigma_B^* = (c_B^*, t_B^*, z_B^*)$.
- If $b = 2$, $\mathcal{B}$ randomly picks $z_A^*, \alpha_A^* \in \mathbb{Z}_q$ and calculates $c_A^* = H(m_A^*, g^{\alpha_A^*} {y_A^*}^{z_A^*} \bmod p)$ and $t_A^* = (\alpha_A^* - c_A^*){x_B^*}^{-1} \bmod q$. Denote $\sigma_A^* = (c_A^*, z_A^*, t_A^*)$. $\mathcal{B}$ randomly picks $t_B^*, \alpha_B^* \in \mathbb{Z}_q$ and calculates $c_B^* = H(m_B^*, g^{\alpha_B^*} {y_A^*}^{t_B^*} \bmod p)$ and $z_B^* = (\alpha_B^* - c_B^*){x_B^*}^{-1} \bmod q$. Denote $\sigma_B^* = (c_B^*, t_B^*, z_B^*)$.
- If $b = 3$, $\mathcal{B}$ randomly picks $t_A^*, \alpha_A^* \in \mathbb{Z}_q$ and calculates $c_A^* = H(m_A^*, g^{\alpha_A^*} {y_B^*}^{t_A^*} \bmod p)$ and $z_A^* = (\alpha_A^* - c_A^*){x_A^*}^{-1} \bmod q$. Denote $\sigma_A^* = (c_A^*, z_A^*, t_A^*)$. $\mathcal{B}$ randomly picks $t_B^*, \alpha_B^* \in \mathbb{Z}_q$ and calculates $c_B^* = H(m_B^*, g^{\alpha_B^*} {y_A^*}^{t_B^*} \bmod p)$ and $z_B^* = (\alpha_B^* - c_B^*){x_B^*}^{-1} \bmod q$. Denote $\sigma_B^* = (c_B^*, t_B^*, z_B^*)$.
- If $b = 4$, $\mathcal{B}$ randomly picks $z_A^*, \alpha_A^* \in \mathbb{Z}_q$ and calculates $c_A^* = H(m_A^*, g^{\alpha_A^*} {y_A^*}^{z_A^*} \bmod p)$ and $t_A^* = (\alpha_A^* - c_A^*){x_B^*}^{-1} \bmod q$. $\mathcal{B}$ randomly picks $z_B^*, \alpha_B^* \in \mathbb{Z}_q$ and calculates $c_B^* = H(m_B^*, g^{\alpha_B^*} {y_B^*}^{z_B^*} \bmod p)$ and $t_B^* = (\alpha_B^* - c_B^*){x_A^*}^{-1} \bmod q$. Denote $\sigma_B^* = (c_B^*, t_B^*, z_B^*)$.

$\mathcal{B}$ returns $(\sigma_A^*, \sigma_B^*, (s_A^*, R_A^*), (s_B^*, R_B^*), k_A^*)$ to $\mathcal{A}$.

Oracle Simulation. The simulator $\mathcal{B}$ simulates the oracles as before, except that for each $(K_B, m_B^*, y_B^*, y_A^*)$ query to the H_1 oracle, $\mathcal{B}$ checks if $g^{K_B} = g^{\alpha}$. If so, $\mathcal{B}$ terminates

and outputs K_B as the solution to the DL problem.

Output. Finally, $\mathcal{A}$ outputs a guess $b' \in \{1, 2, 3, 4\}$. The adversary can check if

- $s_A^* = H_1(k_A^*, m_A^*, y_A^*, y_B^*)$,
- $c_A^* = H(m_A^*, g^{c_A^*} {y_A^*}^{z_A^*} {y_B^*}^{t_A^*})$,
- $c_B^* = H(m_B^*, g^{c_B^*} {y_A^*}^{t_B^*} {y_B^*}^{z_B^*})$.

without the knowledge of k_B^*. On the other hand, if $\mathcal{A}$ can distinguish whether

- $s_B^* = H_1(k_B^*, m_B^*, y_B^*, y_A^*)$,
- $t_A^* = H_2(s_A^*, s_B^*, {y_B^*}^{k_A^*} \mod p, {y_A^*}^{k_B^*} \mod p)$,
- $t_B^* = H_2(s_B^*, s_A^*, {y_A^*}^{k_B^*} \mod p, {y_B^*}^{k_A^*} \mod p)$,

without the knowledge of k_B^*, it contradicts with the condition that H_1 and H_2 are independent collision-resistant hash functions. If $\mathcal{A}$ has the knowledge of $k_B^*(= \alpha)$, and verifies the signature using the H_1 oracle (as defined in the **Verify** protocol), then $\mathcal{B}$ should have terminated in the second Oracle Simulation phase and outputs α as the solution to the DL problem. Therefore the adversary can only win with probability 1/4 if it proceeds to the Output phase.

Probability Analysis. It is straightforward.

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{Z}_p$ element for each ASignOracle, KRevealOracle and H_1 query. The time complexity of $\mathcal{B}$ is

$$t + O\Big((q_s + q_r + q_{h_1})\rho\Big).$$

□

Theorem 8. *Our concinnous signature scheme is (ϵ, t, q_s, q_r)-fair if the (ϵ', t')-discrete logarithm assumption holds in the random oracle model, where*

$$\epsilon' \geq \frac{\epsilon}{2}, \quad t' = t + O\Big((q_s + q_r)\rho\Big),$$

where ρ is the time for an exponentiation in $\mathbb{Z}_p$.

Proof. Assume that there is an adversary $\mathcal{A}$ who can break the fairness, we can construct a simulator $\mathcal{B}$ to solve the discrete logarithm (DL) problem. $\mathcal{B}$ is given the DL problem instance (g, g^{α}). $\mathcal{B}$ is asked to output α.

Setup. $\mathcal{B}$ receives $\mathsf{param} = \{\mathcal{K}, \mathcal{F}, \mathcal{M}, p, q, g, H, H_1, H_2\}$ and the key pairs (y_A^*, x_A^*) and (y_B^*, x_B^*) from the adversary $\mathcal{A}$.

Oracle Simulation. Since the simulator $\mathcal{B}$ knows the user secret keys, he can simulate all oracle queries correctly.

Output. At some point, $\mathcal{A}$ sends a message m_B^* to $\mathcal{B}$. $\mathcal{B}$ flips a random coin b and:

1. If $b = 0$, $\mathcal{B}$ runs $(k_B^*, (s_B^*, R_B^*)) \leftarrow \mathbf{KGen}(m_B^*, y_B^*, y_A^*)$.
2. If $b = 1$, $\mathcal{B}$ randomly picks $s_B^* \in \mathbb{Z}_q$ and sets $R_B^* = g^{\alpha}$. It implicitly sets the keystone as α.

$\mathcal{B}$ returns (s_B^*, R_B^*) to $\mathcal{A}$. $\mathcal{A}$ gives a keystone fix (s_A^*, R_A^*) to $\mathcal{B}$. Then $\mathcal{B}$ runs $\sigma_B^* = (c_B^*, t_B^*, z_B^*) \leftarrow \mathbf{ASign}(m_B^*, (y_A^*, y_B^*), x_B^*, (s_B^*, R_B^*), (s_A^*, R_A^*))$. $\mathcal{B}$ returns σ_B^* to $\mathcal{A}$. $\mathcal{A}$ wins by:

- $\mathcal{A}$ outputs a tuple $(\sigma_A^*, m_A^*, K_A^*, K_B^*)$. If $b = 0$, $\mathcal{B}$ declares failure and exits. If $b = 1$, $\mathcal{A}$ wins if $\mathsf{accept} \leftarrow \mathbf{Verify}(\sigma_A^*, \sigma_B^*, m_A^*, m_B^*, (y_A^*, y_B^*), K_A^*, K_B^*, (s_A^*, R_A^*), (s_B^*, R_B^*))$. It implies that

$$\begin{aligned} s_B^* &= H_1(K_B^*, m_B^*, y_B^*, y_A^*), \\ t_B^* &= H_2(s_B^*, s_A^*, {y_A^*}^{K_B^*} \mod p, {y_B^*}^{K_A^*} \mod p), \\ c_B^* &= H(m_B^*, g^{c_B^*} {y_A^*}^{t_B^*} {y_B^*}^{z_B^*}). \end{aligned}$$

  If the signature is valid, $\mathcal{B}$ outputs K_B^* as the solution of the DL problem.

- $\mathcal{A}$ returns a valid signature (σ_A^*, m_A^*) to $\mathcal{B}$. If $b = 1$, $\mathcal{B}$ declares failure and exits. If $b = 0$, then $\mathcal{B}$ returns the keystone k_B^*. Finally, $\mathcal{A}$ outputs a tuple (σ'_A, m'_A, k'_A). Denote $s'_A = H_1(k'_A, m'_A, y_A^*, y_B^*)$, $R'_A = g^{k'_A} \mod p$ and $\sigma_A^* = (c_A^*, t_A^*, z_A^*)$. $\mathcal{A}$ wins implies that
  - $(s'_A, R'_A) \neq (s_A^*, R_A^*)$,
  - $t_B^* = H_2(s_B^*, s'_A, {y_A^*}^{k_B^*} \mod p, {y_B^*}^{k'_A} \mod p)$.

Since $\mathcal{B}$ honestly generates t_B^*, it implies that $t_B^* = H_2(s_B^*, s_A^*, {y_A^*}^{k_B^*} \mod p, {R_A^*}^{k_B^*} \mod p)$. Therefore the probability that $s_A^* \neq s'_A$ is negligible. On the other hand, the condition $s_A^* = s'_A$ implies that $R_A^* \neq R'_A$ and hence $k_A^* \neq k'_A$. However, the probability that $s_A^* = s'_A$ and $k_A^* \neq k'_A$ is also negligible. Therefore $\mathcal{A}$ wins with negligible probability in this case.

Therefore, $\mathcal{A}$ wins with negligible probability.

Probability Analysis. It is straightforward.

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is determined as follows. There are $O(1)$ exponentiations of $\mathbb{Z}_p$ element for each ASignOracle and KRevealOracle query. The time complexity of $\mathcal{B}$ is

$$t + O\Big((q_s + q_r)\rho\Big).$$

□

4.3.3 Summary

In this Chapter, we demonstrate how to use pairings to construct efficient and provably secure digital signature schemes. We first study two variants of standard signatures, undeniable signatures and sanitizable signatures, and give two new constructions which is provably secure without using random oracle model. Finally, we propose the notion of concinnous signatures, which is used to provide a fair exchange of digital signatures *without* any trusted party. It solves the fairness issue related to the initiating party in the concurrent signature. We also proposed a concrete construction of concinnous signatures and prove the security in the random oracle model.

Pairing is an efficient tool to verify the validity of digital signatures in pairing-based cryptography. Pairing helps us to construct various signatures which may not be able to be constructed by the traditional hash-and-sign paradigm. One of the first pairing-based signature, the BLS signature [38], is not only simple and efficient, it also has the same structure as the identity-based secret key of the first practical identity-based encryption [34]. In fact, the role of pairing-based signatures is significant during the construction of identity-based encryption. Naor observed any identity-based encryption scheme can be converted into a signature scheme with the identity-based secret key playing the role of signatures [34]. In the next Chapter, we will show how pairings can be used in identity-based cryptography.

Chapter 5

Identity-based Cryptography

The notion of identity-based cryptography was put forth by Shamir [189]. This notion was proposed to simplify the authentication of a public key by merely using an identity string as the public key. From the verifier's or the encryptor's point of view, only the identity of the other party is required. Hence, there is no necessity to ensure the validity of the public key. Due to this nice property, a series of identity-based schemes have been proposed, including identity-based signatures [189], identity-based encryption [34], hierarchical identity-based cryptography [93] and so forth.

In this Chapter, we first review the key escrow problem in identity-based cryptography and discuss the existing solution to the key escrow problem in the literature in §5.1. We classify the existing solutions to the key escrow problem into two main categories: *preventive measure* and *blaming mechanism*. For identity-based signatures, we introduce the notion of *escrow-free identity-based signatures* in §5.2 and give a concrete construction. For identity-based encryption, we first discuss the impossibility of *ideal escrow-free identity-based encryption* in §5.3. We then consider the best possible defence against the key escrow problem. In the category of preventive measure, we introduce the notion of *fully anonymous identity-based encryption* in §5.4 and give a concrete construction. In the category of blaming mechanism, we propose some improvements to the black-box accountable authority IBE in §5.5.

5.1 Key Escrow in Identity-based Cryptography

In the identity-based cryptosystems, there is a trusted party called the private key generator (PKG) who generates the secret key for each user identity. As the PKG generates and holds the secret key for all users, a complete trust must be placed on the PKG. Nonetheless, this may not be desirable in a real world scenario, where a malicious PKG can sell users' keys, sign messages or decrypt ciphertexts on behalf of users without being confronted in a court of law. This is known as the *key escrow*

problem. This problem seems to be inherent in identity-based cryptosystems.

5.1.1 Preventive Measure and Blaming Mechanism

We classify the existing solutions to the key escrow problem into two main categories: *preventive measure* and *blaming mechanism*, as shown in Figure 5.1.1. In the preventive measure technique, the chance that a malicious PKG can harm an honest user is minimised. In the cryptosystems with blaming mechanism, the PKG can be blamed if it attempts to harm an honest user. It is also important that the PKG will not be blamed if it does not harm any user.

In the category of preventive measure, Boneh and Franklin [34] proposed a threshold IBE scheme that the master secret key is jointly computed by multiple PKGs, such that no single PKG has the knowledge of it. It protects the users unless a large number of PKGs are corrupted. However, this approach requires an extra infrastructure and communication cost between users and different PKGs. Furthermore, maintaining multiple independent PKGs for a commercially used infrastructure is a daunting task. We also consider anonymous IBE [1] as another possible preventive measure. Without knowing the recipient's identity, the adversary does not know which identity-based secret key that needs to be used to decrypt.

The blaming mechanism is used in Accountable Authority IBE (A-IBE) [101]. Goyal [101] proposed a new system that the identity-based secret key, which is generated by the PKG interacting with the user, is different every time with a high probability. Therefore, the PKG can be caught when there exist two different identity-based secret keys for the same identity. Au *et al.* [8] further proposed an extension that the secret key of the PKG can even be extracted. However, a malicious PKG is still able to sell a signed message or a decrypted ciphertext instead, without being detected. This is clearly an issue that has not yet been addressed in Goyal's model [101]. Goyal *et al.* [102] proposed the black-box A-IBE to blame a malicious PKG selling a decoder box which can decrypt a ciphertext with non-negligible probability. It is an open problem to construct a similar blaming mechanism in the IBS setting.

We emphasise that *both* the preventive measure and the blaming mechanism are important to deal with the key escrow problem. These two methods have their own pros and cons. The blaming mechanism appears to be the more effective solution since it implies a "punishment" for the malicious PKG or the malicious user. However, the damage (e.g. leaking secret military information) has already been done when the punishment is executed, and the damage may be too severe to be fixed. Another

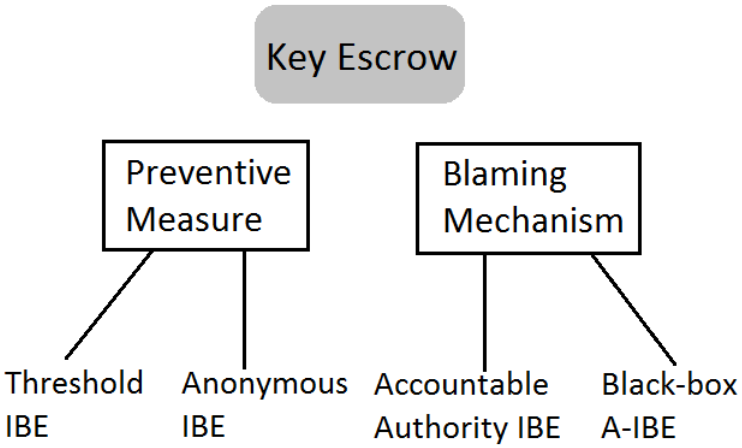


Figure 5.1: Classification of solutions to the key escrow problem

disadvantage is that an attacker may launch a denial of service attack to the blaming mechanism. The attacker, acting as some legitimate users, blames the target PKG of being dishonest by running the blaming mechanism thousands of times.

The preventive measure is useful in the practical aspects. Denial of service attack cannot be applied to the preventive measure in a similar fashion. The preventive measure can minimise the chance of leaking secret information in the first place. However, the preventive measure cannot guarantee that the system is completely secure. For example, it cannot stop hundreds of PKGs to collude together and to compute all users' keys, although the probability of this happening is relatively small.

Combining the preventive measure and the blaming mechanism provides a strong protection to honest users. It is similar to the situation in medical science. If you want to stay healthy, you should take some "preventive measures" (doing exercises, taking vitamins, inoculating with a vaccine). However, you may still have a flu even if you have inoculated with an influenza vaccine. When you are sick, you have to go through a "healing process" (visiting a doctor, taking pills). Although visiting a doctor can usually solve the problem, you may still want to avoid being sick. In order to stay healthy, doctors will suggest you to do both the preventive measures and the healing process. In the case of IBE, we suggest that the preventive measure and the blaming mechanism are *both* equally important to deal with the key escrow problem by similar reasons.

5.1.2 Non-identity-based Solutions

Some cryptosystems have been proposed to solve the key escrow problem. They use a "combination" of identity-based cryptography and the traditional public key cryptography, such as the certificateless cryptosystems [4], the certificate-based cryptosystems [91] and the self-certificated cryptosystems [96], in a non-trivial way. In these systems, a user possesses a user public key and a user secret key, together with his identity-based secret key computed by the PKG. The user secret key protects the user from the key escrow problem. The PKG acts like a certificate authority (CA) who authenticates the user public key using his master secret key. Unfortunately, these cryptosystems are *no longer* identity-based - the encryptor or the verifier has to know the user public key in addition to the user identity. Therefore these schemes lost the original advantages of identity-based cryptography.

Girault [96] defined three levels of trust to the PKG in these cryptosystems:

- Level 1: The PKG can compute users' secret keys and, therefore, can impersonate any user without being detected. Identity-based signature schemes are the examples.
- Level 2: The PKG cannot compute users' secret keys. However, the PKG can still impersonate any user without being detected. Certificateless signature schemes are the examples.
- Level 3: The PKG cannot compute users' secret keys, and the PKG cannot impersonate any user without being detected. Certificate-based signature schemes and self-certificated signature schemes are the examples.

The current schemes achieving level 2 or level 3 of trust are no longer identity-based.

5.2 Escrow-free Identity-based Signatures

In this section, we introduce the concept of *escrow-free identity-based signatures* to reduce the trust in the PKG. It is an open problem to construct an identity-based signatures with level 2 or level 3 of trust (according to Girault's definition [96] in §5.1), without publishing the user public key. We solve this problem by proposing a concrete construction of identity-based signatures with level 3 of trust.

In this section, we first introduce the security model of *escrow-free identity-based signatures*. In this model, each signer has his own public key and secret key. The

Table 5.1: Comparison of the public information known by the verifier and the level of trust to the PKG. ID is the identity, upk is the user public key, and W is the commitment of the user secret key using the public key of the PKG.

Schemes	Public Information	Level of Trust
Identity-based Signatures [189]	ID	Level 1
Certificateless Signatures [4]	ID, upk, W	Level 2
Certificate-based Signatures [126]	ID, upk	Level 3
Self-certificated Signatures [96]	ID, upk	Level 3
Our Scheme in §5.2.4	ID	Level 3

PKG generates the identity-based secret key for the signer with respect to the user public key (*à la* Goyal's approach [101]). Then, the signer uses both secret keys to sign a message. Therefore, the signer is protected against a malicious PKG. To verify the signature, it only requires the signer's identity and the message. This is the main difference between certificate-based signatures (CBS), certificateless signatures (CLS), self-certificated signatures (SCS) and our model. Their verification protocols require the signer's public key to verify.[1] Hence, our model mimics closely the original IBS in this regard, and solves the key escrow problem at the same time. We compare the public information known by the verifier and the level of trust to the PKG in traditional IBS, CBS, CLS and SCS in Table 5.1.

In this section, we give two constructions of escrow-free IBS, which achieve level 3 of trust to the PKG and are the best in the model proposed by Girault [96]. The first construction in §5.2.2 is a generic construction using standard signatures. It is similar to the construction of certificate-based IBS in [17]. The formalisation of the PKG non-frameability and user non-frameability in §5.2.1 introduces the main difference between our generic construction and the certificate-based IBS in [17]. The advantage of the first construction is that the scheme is secure in the standard model if the underlying signature scheme is also secure in the standard model (such as the Waters signature [206]). The disadvantage is that the user public key is part of the signature.

We give a concrete construction using pairings in §5.2.4. This escrow-free IBS scheme has an extra property called *user public key anonymity*. In CBS, CLS and SCS, user public keys are needed to verify a signature. Since the escrow-free IBS only uses the identity to verify a signature, it is possible for the signature to be anonymous with respect to the user public key. We provide an additional security model to capture

[1] We note that Chen *et al.* proposed an identity-based signature scheme without trusted PKG in [63]. In their scheme, the long-term user secret key is r and the signature includes rP. If we regard the user public key as rP, then their verification protocol also requires the "user public key" rP.

the user public key anonymity property. The advantage of our scheme in §5.2.4 is that it achieves user public key anonymity, and the public information only consists of the identity. The disadvantage of our concrete construction is that it uses the expensive proof of knowledge protocol.

Our escrow-free IBS schemes achieve level 3 of trust to the PKG in the model of [96]. Theoretically, the escrow-free IBS is more efficient than CBS, CLS and SCS since the user public key is not involved and is not sent to the verifier. In this paper, we give the *first* construction of the escrow-free IBS. When comparing with the multiple PKGs solution by Boneh and Franklin [34], our scheme interacts with at most two authorities. While Boneh and Franklin's scheme interacts with a large number of authorities, the communication complexity of their scheme is higher.

5.2.1 Security Model for Escrow-free Identity-based Signatures

An escrow-free identity-based signature scheme has six polynomial-time algorithms, namely **Setup**, **UserKeyGen**, **Extract**, **Sign**, **Verify**, **Blame**.

- **Setup**: On input a security parameter 1^λ, it generates the system parameters param, the master secret key msk and the master public key mpk.
- **UserKeyGen**: On input the system parameters param, the user generates the user secret key usk and the user public key upk.
- **Extract**: This is an efficient two-party protocol $(\mathbf{Extract}_p, \mathbf{Extract}_u)$ between the PKG and the user[2]. The common inputs are param, upk and an identity ID. The PKG's algorithm $\mathbf{Extract}_p$ private input is msk. The user's algorithm $\mathbf{Extract}_u$ private input is usk. The interaction includes the user giving the PKG a joining proof Pf which shows the user's participation with respect to upk[3]. Finally the user obtains the identity-based secret key $sk_{\mathsf{ID},upk}$ with respect to the identity ID and the user public key upk.
- **Sign**: On input param, usk, $sk_{\mathsf{ID},upk}$ and a message m, the user with identity ID generates a signature σ.
- **Verify**: On input param, mpk, ID, m and σ, it returns 1 or 0 for accept or reject, respectively.

[2]Two-party protocol is defined in definition 13
[3]The joining proof will be defined in section 5.2.1

- **Blame**: This is an interactive algorithm between the PKG, the user and the judge. The common inputs are param, mpk, ID, upk, m and σ. The user's algorithm $\mathbf{Blame}_u$ with private input usk sends a blame request φ to a judge. The judge's algorithm $\mathbf{Blame}_j$ outputs "PKG" if:
  - φ shows that σ is related to upk, and
  - the PKG's algorithm $\mathbf{Blame}_p$, with private input msk, fails to provide a public key upk', a joining proof Pf and a transcript ρ, such that:
    - upk' is related to σ,
    - Pf shows the user's participation with respect to upk', and
    - ρ is the transcript of the extract algorithm with upk'.

  Otherwise, the judge outputs "upk".

There are four types of entities involved in the escrow-free identity-based signatures: users, the PKG, the judge, and a Trusted Third Party (TTP) who will certify the joining proof Pf [4]. We assume that the PKG and the TTP do not collude. Otherwise, the malicious PKG can sign a message with respect to upk and the TTP can provide a fake joining proof Pf. As a result, the malicious PKG cannot be caught by the **Blame** algorithm.

We also assume that the user and the TTP do not collude. Otherwise, the malicious user can sign a message using his identity-based secret key and the TTP can intentionally disapprove the joining proof Pf. As a result, the malicious user cannot be caught by the **Blame** algorithm.

Joining Proof

The joining proof Pf can be either an online proof or a proof in the real world. A Trusted Third Party (TTP) certifies the joining proof. For the online proof, it can consist of a certificate issued by some authority (TTP) with respect to upk, and a proof of knowledge with respect to upk. For the real world proof, it can be the user's signature on an application form, or the photocopy of the user's documentation. The TTP will verify the user's signature or the user's documentation.

The joining proof Pf is needed to protect both the PKG and the user in the **Blame** protocol. If there is no such proof:

[4] The role of the TTP will be explained in section 5.2.1

- a malicious PKG can generate $sk_{\mathsf{ID},upk}$ using any upk generated by himself and an honest user cannot show that upk is not his public key;
- a malicious user can claim that the upk used in $sk_{\mathsf{ID},upk}$ is not his public key and frame an honest PKG.

The joining proof can be viewed as an authentication of user public key, which is separated from the identity-based secret key issuing. Similar concepts can be found in "anonymous identity-based key issuing" [198], where the duties of authentication and key issuing are separated to local registration authorities (LRA) and the PKG. Recently, Chow [65] proposed a new system architecture to realise "anonymous key issuing", by employing non-colluding identity-certifying authority (ICA) and PKG. However, these two systems only authenticate the user identity. If we modify the LRA or ICA to authenticate user public key as well, it can be used as a joining proof.

Correctness

Let $sk_{\mathsf{ID},upk} \leftarrow \mathbf{Extract}(\mathsf{param}, upk, \mathsf{ID})$ and $(usk, upk) \leftarrow \mathbf{UserKeyGen}(\mathsf{param})$. Then we define the *verification correctness* as follows:

$$\mathbf{Verify}(\mathsf{param}, mpk, \mathsf{ID}, m, \mathbf{Sign}(\mathsf{param}, usk, sk_{\mathsf{ID},upk}, m)) = 1.$$

We also define the *blaming correctness* as follows:

$$\mathbf{Blame}(\mathsf{param}, mpk, \mathsf{ID}, upk, m, \mathbf{Sign}(\mathsf{param}, usk, sk_{\mathsf{ID},upk}, m)) = upk,$$

if $\mathbf{Extract}(\mathsf{param}, upk, \mathsf{ID})$ was run between the user and the PKG. On the other hand,

$$\mathbf{Blame}(\mathsf{param}, mpk, \mathsf{ID}, upk', m, \mathbf{Sign}(\mathsf{param}, usk, sk_{\mathsf{ID},upk}, m)) = PKG,$$

if $\mathbf{Extract}(\mathsf{param}, upk, \mathsf{ID})$ has never been run between the user and the PKG.

Unforgeability

The security model for unforgeability captures the attack from the outsider to forge a signature when the PKG is honest. The adversary can obtain signatures of an honest user and can get the identity-based secret key of any identity except the challenge identity. We have the following game for unforgeability:

1. The challenger $\mathcal{C}$ runs the algorithms $(\mathsf{param}, msk, mpk) \leftarrow \mathbf{Setup}(1^\lambda)$ and $(usk', upk') \leftarrow \mathbf{UserKeyGen}(\mathsf{param})$. $\mathcal{C}$ gives param, mpk and upk' to the adversary $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the following oracles adaptively:

   - Key Extraction Oracle $\mathcal{KEO}(upk, \mathsf{ID})$: $\mathcal{A}$ runs the $\mathsf{Extract}_u$ protocol to query the oracle. Finally the oracle returns an identity-based secret key $sk_{\mathsf{ID},upk}$ with respect to ID and upk.
   - Signing Oracle $\mathcal{SO}(m, \mathsf{ID})$: it returns a valid signature $\sigma \leftarrow \mathbf{Sign}(\mathsf{param}, usk', sk_{\mathsf{ID},upk'}, m)$.

3. $\mathcal{A}$ returns a signature σ^* for a message m^* and an identity ID^*.

$\mathcal{A}$ wins the game if $\mathbf{Verify}(\mathsf{param}, mpk, \mathsf{ID}^*, m^*, \sigma^*) = 1$, such that there was no query that $\mathcal{SO}(m^*, \mathsf{ID}^*, \cdot)$ and there was no query that $\mathcal{KEO}(\cdot, \mathsf{ID}^*)$.

Definition 29. *An escrow-free IBS scheme is (ϵ, t, q_e, q_s)-secure against unforgeability if there is no t time adversary winning the above game with probability at least ϵ with q_e and q_s queries to $\mathcal{KEO}$ and $\mathcal{SO}$ respectively.*

PKG Non-frameability

The security model for PKG non-frameability captures the attack from a malicious user having an identity-based secret key that wants to frame an honest PKG. If the attacker without any identity-based secret key wants to frame an honest PKG, he must firstly forge a valid signature. Since this scenario has been captured in the model of unforgeability, we only consider the case that a malicious user, who already obtains an identity-based secret key, wants to frame an honest PKG. We have the following game for PKG non-frameability:

1. The challenger $\mathcal{C}$ runs the algorithms $(\mathsf{param}, msk, mpk) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives param and mpk to the adversary $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to adaptively query the Key Extraction Oracle $\mathcal{KEO}(upk, \mathsf{ID})$: $\mathcal{A}$ runs the $\mathsf{Extract}_u$ protocol to query the oracle. Finally the oracle returns an identity-based secret key $sk_{\mathsf{ID},upk}$ with respect to ID and upk. $\mathcal{C}$ saves the transcript ρ in this query and also the user's joining proof Pf.

3. $\mathcal{A}$ returns a signature σ^* for a message m^* and an identity ID^*, such that he can blame the PKG by the Blame_u protocol with a public key upk^* and a blame request φ^*.

$\mathcal{A}$ wins the game if **Verify**(param, mpk, ID^*, m^*, σ^*) = 1 and **Blame**$_j$(param, mpk, ID^*, upk^*, m^*, σ^*, φ^*) = PKG.

Definition 30. *An escrow-free IBS scheme is (ϵ, t, q_e)-secure against PKG non-frameability if there is no t time adversary winning the above game with probability at least ϵ with q_e queries to $\mathcal{KEO}$.*

We note that the PKG non-frameability is related to the unforgeability in the following sense. If the signature is forgeable by an outside attacker $\mathcal{A}$ (not the PKG nor the user), we denote that the "secret key" corresponds to this signature is $sk_{\mathcal{A}}$. If we assume that the attacker does not obtain the transcript of the Extract protocol, then $sk_{\mathcal{A}}$ is different from the real secret key generated in Extract with high probability. Therefore the signature from $\mathcal{A}$ can be used to frame the honest PKG with high probability. To summarise, the PKG non-frameability implies that the unforgeability and the transcript of the Extract protocol does not help $\mathcal{A}$ to generate the real identity-based secret key.

User Non-frameability

The security model for user non-frameability captures the attack from a malicious PKG that wants to frame an honest user. We have the following game for user non-frameability:

1. The challenger $\mathcal{C}$ gives param to the adversary $\mathcal{A}$. $\mathcal{A}$ gives a master public key mpk to $\mathcal{C}$. $\mathcal{C}$ runs the algorithm $(usk^*, upk^*) \leftarrow$ **UserKeyGen**(param). $\mathcal{C}$ gives the user public key upk^* and a joining proof Pf^* to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the following oracles adaptively:

   - User Join Oracle $\mathcal{JO}(\mathsf{ID})$: $\mathcal{C}$ acts as the Extract$_u$ protocol with input (upk^*, Pf^*) and interacts with $\mathcal{A}$ (running Extract$_p$) for the identity ID. Finally $\mathcal{C}$ obtains an identity-based secret key sk_{ID,upk^*} and $\mathcal{A}$ obtains a transcript ρ.
   - Signing Oracle $\mathcal{SO}(m, \mathsf{ID})$: it returns a valid signature $\sigma \leftarrow$ **Sign**(param, $usk^*, sk_{\mathsf{ID},upk^*}, m$).

3. $\mathcal{A}$ returns a signature σ^* for a message m^* and an identity ID^*.

$\mathcal{A}$ wins the game if **Verify**(param, mpk, ID^*, m^*, σ^*) = 1 and **Blame**(param, mpk, ID^*, upk, m^*, σ^*) = upk for all upk. The latter equation is always satisfied by $\mathcal{A}$

running $\mathbf{Blame}_p$ and giving (upk^*, Pf^*, ρ^*) to the judge (where ρ^* is the output of $\mathcal{JO}(\mathsf{ID}^*)$). We require that there was no query that $\mathcal{SO}(m^*, \mathsf{ID}^*)$.

Definition 31. *An escrow-free IBS scheme is (ϵ, t, q_j, q_s)-secure against user non-frameability if there is no t time adversary winning the above game with probability at least ϵ with q_j and q_s queries to $\mathcal{JO}$ and $\mathcal{SO}$ respectively.*

5.2.2 Generic Construction

We present a generic construction of escrow-free IBS from standard signatures. This is similar to the construction of certificate-based IBS in [17].

Our Scheme

Suppose there is a standard digital signature scheme $\mathcal{SS} = (\mathsf{SKg}, \mathsf{Sign}, \mathsf{Vf})$ which is unforgeable against chosen message attack (UF-CMA), we construct our escrow-free IBS scheme as follows:

- **Setup**: On input the security parameter 1^λ, it outputs $(mpk, msk) \leftarrow \mathsf{SKg}(1^\lambda)$. The system parameters param is just the security parameter 1^λ.
- **UserKeyGen**: On input param, the user obtains $(upk, usk) \leftarrow \mathsf{SKg}(1^\lambda)$.
- **Extract**: The PKG algorithm $\mathbf{Extract}_p$ has input $(\mathsf{param}, upk, \mathsf{ID}, msk)$. The user algorithm $\mathbf{Extract}_u$ has input $(\mathsf{param}, upk, \mathsf{ID}, usk)$. The user computes $s \leftarrow \mathsf{Sign}_{usk}(\mathsf{ID})$ and sends $(s, \mathsf{ID}, upk, Pf)$ to the PKG. The PKG checks if $1 \leftarrow \mathsf{Vf}_{upk}(\mathsf{ID}, s)$ and Pf is a joining proof. If they are correct, then the PKG computes the identity-based secret key $sk_{\mathsf{ID},upk} \leftarrow \mathsf{Sign}_{msk}(\mathsf{ID}||upk)$. The PKG saves the join transcript $\rho = (s, \mathsf{ID}, upk, Pf)$ and then sends $sk_{\mathsf{ID},upk}$ to the user.
- **Sign**: On input param, usk, $sk_{\mathsf{ID},upk}$ and a message m, the user computes $\sigma_1 \leftarrow \mathsf{Sign}_{usk}(m||\mathsf{ID})$. The user outputs the signature $\sigma = (\sigma_1, upk, sk_{\mathsf{ID},upk})$.
- **Verify**: On input param, mpk, ID, m and $\sigma = (\sigma_1, upk, sk_{\mathsf{ID},upk})$, it returns 1 if $1 \leftarrow \mathsf{Vf}_{upk}(m||\mathsf{ID}, \sigma_1)$ and $1 \leftarrow \mathsf{Vf}_{mpk}(\mathsf{ID}||upk, sk_{\mathsf{ID},upk})$.
- **Blame**: On common input param, mpk, ID, upk, m and $\sigma = (\sigma_1, upk, sk_{\mathsf{ID},upk})$, the user asks the judge to blame the PKG. The judge asks the PKG to provide a transcript $\rho = (s, \mathsf{ID}, upk, Pf)$. If $1 \leftarrow \mathsf{Vf}_{upk}(\mathsf{ID}, s)$ and Pf is a valid joining proof, the judge outputs upk. Otherwise, the judge outputs PKG.

Remarks. Although the user public key is part of the signature, the scheme is still considered as IBS. Similar approach is proposed by Shamir [189] and discussed in [17, 89].

Security Proofs

The correctness of the scheme is straightforward. We state the security of the above construction in the following theorems.

Theorem 9. *The scheme is unforgeable if $\mathcal{SS}$ is a UF-CMA secure signature scheme.*

Proof. Assume there is a (ϵ, t, q_e, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that uses $\mathcal{A}$ to forge a signature of $\mathcal{SS}$ with probability at least ϵ and in time at most t.

Setup. $\mathcal{B}$ runs the $\mathcal{SS}$ simulator twice and obtains two public keys pk_1 and pk_2. $\mathcal{B}$ gives $\mathcal{A}$ the master public key $mpk = pk_1$ and the honest user public key $upk' = pk_2$.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:

(*Key Extraction oracle.*) On input $(upk, \mathsf{ID}, s, Pf)$ from the $\mathsf{Extract}_u$ protocol, $\mathcal{B}$ first check if Pf is a valid joining proof for upk and $1 \leftarrow \mathsf{Vf}_{upk}(\mathsf{ID}, s)$. If they are correct, $\mathcal{B}$ queries the signing oracle of $\mathcal{SS}$ for pk_1 with input $(\mathsf{ID}||upk)$. $\mathcal{B}$ forwards the result to $\mathcal{A}$.

(*Signing oracle.*) On input (m, ID), $\mathcal{B}$ queries the signing oracle of $\mathcal{SS}$ for pk_1 with input $(\mathsf{ID}||pk_2)$ and obtains sk. $\mathcal{B}$ queries the signing oracle of $\mathcal{SS}$ for pk_2 with input $m||\mathsf{ID}$ and obtains σ_1. $\mathcal{B}$ returns (σ_1, pk_2, sk).

Output. Finally $\mathcal{A}$ outputs a signature $\sigma^* = (\sigma_1^*, upk^*, sk^*)$ for a message m^* and an identity ID^*.

- If $upk^* \neq pk_2$, then $\mathcal{B}$ returns sk^* to the $\mathcal{SS}$ simulator. It is the forgery for the message $\mathsf{ID}^*||upk^*$ with respect to the public key pk_1.
- If $upk^* = pk_2$, then $\mathcal{B}$ returns σ_1^* to the $\mathcal{SS}$ simulator. It is the forgery for the message $m^*||\mathsf{ID}^*$ with respect to the public key pk_2.

□

Theorem 10. *The scheme is PKG non-frameable if $\mathcal{SS}$ is a UF-CMA secure signature scheme and the joining proof is unforgeable.*

Proof. Assume there is a (ϵ, t, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that uses $\mathcal{A}$ to forge a signature of $\mathcal{SS}$ with probability at least ϵ and in time at most t.

Setup. $\mathcal{B}$ runs the $\mathcal{SS}$ simulator and obtains a public key pk. $\mathcal{B}$ gives $\mathcal{A}$ the master public key $mpk = pk$.

Oracles Simulation. The simulation of the *key extraction oracle* is the same as that of theorem 9.

Output. Finally $\mathcal{A}$ outputs a signature $\sigma^* = (\sigma_1^*, upk^*, sk^*)$ for a message m^* and an identity ID^*. $\mathcal{A}$ blames the PKG with a public key upk^*.

- If $(upk, \mathsf{ID}, \cdot, \cdot)$ was not successfully queried in the key extraction oracle, $\mathcal{B}$ returns sk^* as the forgery for the message $\mathsf{ID}^*||upk^*$ with respect to the public key mpk.
- Otherwise, $\mathcal{B}$ tries to reply to the judge with the transcript $\rho = (s', \mathsf{ID}^*, upk^*, Pf)$ with respect to the blame from $\mathcal{A}$. $\mathcal{A}$ wins the game if either Pf is not a valid joining proof or s' is not a valid signature. However it is not possible since the transcript is checked during the oracle query.

□

Theorem 11. *The scheme is user non-frameable if $\mathcal{SS}$ is a UF-CMA secure signature scheme and the joining proof is unforgeable.*

Proof. Assume there is a (ϵ, t, q_j, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that uses $\mathcal{A}$ to forge a signature of $\mathcal{SS}$ with probability at least ϵ and in time at most t.

Setup. $\mathcal{B}$ gives $\mathsf{param} = 1^\lambda$ to $\mathcal{A}$. $\mathcal{A}$ gives the master public key mpk and the target identity ID^* to $\mathcal{B}$. $\mathcal{B}$ runs the $\mathcal{SS}$ simulator with 1^λ and obtains a public key pk. $\mathcal{B}$ obtains a joining proof Pf^* for pk from an honest CA. $\mathcal{B}$ gives $\mathcal{A}$ the user public key $upk^* = pk$ and Pf^*.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:

(*Join oracle.*) On input ID, $\mathcal{B}$ queries the signing oracle of $\mathcal{SS}$ with input (ID) to obtain s. $\mathcal{B}$ sends $\rho = (s, \mathsf{ID}, upk^*, Pf^*)$ to $\mathcal{A}$. $\mathcal{A}$ stores the transcript ρ. $\mathcal{A}$ finally replies $\mathcal{B}$ with $sk_{\mathsf{ID},upk}$.

(*Signing oracle.*) On input (m, ID), $\mathcal{B}$ first runs as the join oracle with input ID. Finally $\mathcal{B}$ obtains $sk_{\mathsf{ID},upk}$. Then $\mathcal{B}$ queries the signing oracle of $\mathcal{SS}$ with input $m||\mathsf{ID}$ and obtains σ_1. $\mathcal{B}$ returns $(\sigma_1, upk^*, sk_{\mathsf{ID},upk})$.

Output. Finally $\mathcal{A}$ outputs a signature $\sigma^* = (\sigma_1^*, upk^*, sk^*)$ for a message m^* and an identity ID^*. $\mathcal{A}$ blames the user with a public key upk^* and a transcript ρ^*. $\mathcal{B}$ returns σ_1^* as the forgery of the $\mathcal{SS}$ signature for the message $(m^*||\mathsf{ID}^*)$. □

5.2.3 User Public Key Anonymity

In the previous section, we propose a generic construction of escrow-free IBS. However, the user public key is included in the signature. Therefore it is similar to the certificate-based signatures to some extent. In some applications, it may not be desirable to let the verifier knowing the user public key (but only the identity). For example, assume a student has a long-term user public key. He may apply for an identity-based secret key for his student ID from the university. He may also apply for an identity-based secret key for his email address from the internet service provider. When a user uses the escrow-free IBS, he may not want the signatures for two different identities to be linked to the same user public key.

In order to construct an escrow-free IBS scheme which is *fully identity-based*, we require that the signature contains no information about the user public key. We call this additional property as "user public key anonymity" [5]. In this section, we define the additional security model for the user public key anonymity.

Security Model for Anonymity

The security model for user public key anonymity captures the attack that wants to distinguish if a signature is signed by an honest user with a user public key upk. The attacker is given the master secret key, but cannot query any join oracle. In other words, the attacker can retrieve the master secret key from the real PKG, but not the join transcript from the real PKG. The users joining the real PKG will have anonymity even if the master secret key is stolen. We have the following game for anonymity:

1. The challenger $\mathcal{C}$ runs the algorithms $(\mathsf{param}, msk, mpk) \leftarrow \mathbf{Setup}(1^\lambda)$, $(usk_0, upk_0) \leftarrow \mathbf{UserKeyGen}(\mathsf{param})$ and $(usk_1, upk_1) \leftarrow \mathbf{UserKeyGen}(\mathsf{param})$. $\mathcal{C}$ gives param, the master public key mpk, the master secret key msk and two user public keys upk_0, upk_1 to the adversary $\mathcal{A}$.

[5] An escrow-free IBS scheme can either have the "user public key anonymity" property or not.

2. $\mathcal{A}$ is allowed to query the oracle adaptively: Signing Oracle $\mathcal{SO}(m, \mathsf{ID}, b)$: it returns a valid signature $\sigma \leftarrow \mathbf{Sign}(\mathsf{param}, usk_b, sk_{\mathsf{ID},upk_b}, m)$.

3. $\mathcal{A}$ sends a message m^* and an identity ID^* to $\mathcal{C}$. $\mathcal{C}$ picks a random bit b' and computes $\sigma^* \leftarrow \mathbf{Sign}(\mathsf{param}, usk_{b'}, sk_{\mathsf{ID}^*,upk_{b'}}, m)$. $\mathcal{C}$ sends σ^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the above oracles adaptively.

5. $\mathcal{A}$ returns a bit b^*.

$\mathcal{A}$ wins the game if $b' = b^*$. We require that there was no query that $\mathcal{SO}(m^*, \mathsf{ID}^*, \cdot)$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game over $1/2$.

Definition 32. *An identity-based signature scheme is (ϵ, t, q_s)-secure against anonymity if there is no t time adversary winning the above game with probability at least ϵ with q_s queries to $\mathcal{SO}$.*

Remark. The security model for *key-privacy* or *anonymity* in traditional public key encryption was proposed by Bellare *et al.* [15]. In this section, we follow their notion of "indistinguishability of keys under chosen-ciphertext attacks" and adopt the indistinguishability game into our IBS setting.

The main difference between Bellare *et al.*'s model and our model is that the challenge user secret keys and the user public keys are not chosen by the adversary in our model. It is because our **Blame** algorithm requires that the PKG is able to show that "the *upk* is related to the signature σ" if σ is signed by the corresponding *usk*. If both the msk, usk_0 and usk_1 are known to the adversary, he can generate the join transcript by himself and checks if upk_0 or upk_1 is related to the challenge signature σ^*. It will break the anonymity. Therefore in our anonymity model, the adversary is not given usk_0 and usk_1. The adversary is given the signing oracle for usk_0 and usk_1 instead.

5.2.4 Construction with User Public Key Anonymity

In this section, we provide a concrete construction with user public key anonymity. Our construction for escrow-free IBS is based on the signature schemes from Boneh and Boyen [29] and Boneh *et al.* [38]. We also use the "signatures of knowledge" (SoK) notion from Chase and Lysyanskaya [57] (refer to §3.2.2 for details).

Intuition

We use the signature scheme from Boneh and Boyen [29] as the identity-based secret key. Suppose the master secret key is α and the master public key is g^α. For a user with secret key x and public key $y = g^x$, his identity-based secret key is A where

$$A^{\alpha+\mathsf{ID}} v^x = u,$$

and g, u, v are a generator of $\mathbb{G}$.

For the signing protocol, the part of the signature useful for the blame protocol is derived from Boneh *et al.* [38]. Denote this part as S and we have

$$S = \hat{e}(v, H_2(m))^x,$$

where m is the message. The the signing protocol becomes:

$$SoK\{(A, x) : A^{\alpha+\mathsf{ID}} v^x = u \quad \wedge \quad S = \hat{e}(v, H_2(m))^x\}(m).$$

Our Scheme

We give the detailed construction of the escrow-free IBS with anonymity.

- **Setup**: The algorithm first chooses a random prime p of bit size $\Theta(k)$. Let $\mathbb{G}$, $\mathbb{G}_T$ be a bilinear group of order p and a pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. It also chooses generators $g, u, v \in \mathbb{G}$. It picks collision resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ for hashing the identity string, and $H_2 : \{0,1\}^* \to \mathbb{G}$ for hashing the message. It also chooses generators $g_0, g_1, g_2 \in \mathbb{G}$ used for the signature of knowledge. The system parameters param is ($\hat{e}$, $\mathbb{G}$, $\mathbb{G}_T$, p, g, u, v, g_0, g_1, g_2, H_1, H_2).

  The PKG randomly selects his master secret key $\alpha \in \mathbb{Z}_p^*$. He computes the master public key $g_a = g^\alpha$.

- **UserKeyGen**: The user randomly selects his user secret key $x \in \mathbb{Z}_p^*$. He computes $y = g^x$ as his user public key.

- **Extract**: The user calculates $v' = v^x$. He also computes a non-interactive zero-knowledge (NIZK) proof [6] Σ of x with respect to v' and v (We omit the details of the NIZK proof for discrete logarithm for simplicity). He sends v', ID, y, a joining

[6] Although v' can be used to prove the knowledge of x via pairing, we need the extractor of the NIZK proof to obtain x in the security proof.

proof Pf and the NIZK proof Σ to the PKG. The PKG checks the validity of Pf, Σ. If so, the PKG computes:

$$A = (uv'^{-1})^{\frac{1}{\alpha+i}},$$

where $i = H_1(\mathsf{ID})$ and returns A to the user. The PKG stores the transcript $\rho = (v', \Sigma, \mathsf{ID}, y, Pf)$.

- **Sign**: The user signs a message m with the user secret key x and the identity-based secret key A. He computes the signature of knowledge (SoK):

$$SoK\{(A, x) : A^{\alpha+i}v^x = u \quad \wedge \quad S = \hat{e}(v, H_2(m))^x\}(m)$$

The SoK is specified as follows. The user randomly chooses $s, r, r_2 \in \mathbb{Z}_p^*$, $R_1 \in \mathbb{G}$ and computes:

$$t_0 = g_0^s, \quad t_1 = Ag_1^s, \quad t_2 = v^x g_2^s, \quad \tau_0 = g_0^r, \quad \tau_1 = R_1 g_1^r,$$
$$\tau_2 = v^{r_2} g_2^r, \quad \tau_3 = [\hat{e}(g_1, g_a g^i) \cdot \hat{e}(g_2, g)]^r, \quad \tau_4 = \hat{e}(g_2, H_2(m))^r.$$

The user computes $c = H_3(t_0, t_1, t_2, \tau_0, \ldots, \tau_4, m, mpk, \mathsf{ID})$ and:

$$z_0 = r - cs, \quad Z_1 = R_1 A^{-c}, \quad z_2 = r_2 - cx.$$

The signature is $\sigma = (t_0, t_1, t_2, c, z_0, Z_1, z_2, S)$.

- **Verify**: Upon input a signature σ for a message m and an identity ID, it computes:

$$i = H_1(\mathsf{ID}), \quad t_3 = \hat{e}(t_1, g_a g^i) \cdot \hat{e}(t_2, g) \cdot \hat{e}(u, g)^{-1}, \quad t_4 = \hat{e}(t_2, H_2(m)) \cdot S^{-1},$$
$$\tau_0 = g_0^{z_0} t_0^c, \quad \tau_1 = Z_1 g_1^{z_0} t_1^c, \quad \tau_2 = v^{z_2} g_2^{z_0} t_2^c,$$
$$\tau_3 = [\hat{e}(g_1, g_a g^i) \cdot \hat{e}(g_2, g)]^{z_0} \cdot t_3^c, \quad \tau_4 = \hat{e}(g_2, H_2(m))^{z_0} \cdot t_4^c.$$

It outputs 1 if $c = H_3(t_0, t_1, t_2, \tau_0, \ldots, \tau_4, m, mpk, \mathsf{ID})$. Otherwise, it outputs 0.

- **Blame**: On common input the master public key mpk, an identity ID, a message m, a signature σ, a user public key y, the user with user secret key x first computes $\varphi = v^x$. The user sends φ to the judge as the blame request.

The judge checks if $\sigma = (t_0, t_1, t_2, c, z_0, Z_1, z_2, S)$ is a valid signature and:

$$\hat{e}(v, y) = \hat{e}(\varphi, g) \quad \wedge \quad \hat{e}(\varphi, H_2(m)) \neq S.$$

If the check does not hold, the judge returns "*upk*".

Otherwise, the judge requests the PKG to provide a transcript $\rho = (v', \Sigma, \mathsf{ID}, y', Pf')$. If Pf' is a valid joining proof for y' and

$$\hat{e}(v, y') = \hat{e}(v', g) \quad \wedge \quad \hat{e}(v', H_2(m)) = S,$$

the judge returns "*upk*". Otherwise, the judge returns "PKG".

Security Proofs

The correctness of the signature scheme is straightforward.

We first prove that the SoK protocol above is a secure signature of knowledge. We use the game-based definition (SimExt-secure) in [57] (refer to §3.2.2 for details). Chase and Lysyanskaya [57] proved the equivalence of the game-based definition and the UC framework definition.

Lemma 4. *The SoK protocol above is a SimExt-secure signature of knowledge of a witness* (A, x).

Proof. The *correctness* of the signature of knowledge scheme is straightforward.

For *simulatability*, after the simulator randomly picks A and x to compute t_0, t_1, t_2, S, he picks a challenge $c \in \mathbb{Z}_p^*$ and also $z_0, z_2 \in \mathbb{Z}_p^*$, $Z_1 \in \mathbb{G}$. Then compute:

$$\begin{aligned}
&t_3 = \hat{e}(t_1, g_a g^i) \cdot \hat{e}(t_2, g) \cdot \hat{e}(u, g)^{-1}, \quad t_4 = \hat{e}(t_2, H_2(m)) \cdot S^{-1}, \\
&\tau_0 = g_0^{z_0} t_0^c, \quad \tau_1 = Z_1 g_1^{z_0} t_1^c, \quad \tau_2 = v^{z_2} g_2^{z_0} t_2^c, \\
&\tau_3 = [\hat{e}(g_1, g_a g^i) \cdot \hat{e}(g_2, g)]^{z_0} \cdot t_3^c, \quad \tau_4 = \hat{e}(g_2, H_2(m))^{z_0} \cdot t_4^c.
\end{aligned}$$

Therefore the transcript above is simulatable.

For *extraction*, we now show that there exists an extractor for A and x. Assume there are two transcripts with the same commitment $(t_0, t_1, t_2, \tau_0, \ldots, \tau_4)$ and different challenges c, c' and responses (z_0, Z_1, z_2), (z_0', Z_1', z_2'). Let:

$$\tilde{s} = \frac{z_0' - z_0}{c - c'}, \quad \tilde{A} = (Z_1'/Z_1)^{\frac{1}{c-c'}}, \quad \tilde{x} = \frac{z_2' - z_2}{c - c'}.$$

Then, we have

$$t_0 = g_0^{\tilde{s}}, \quad t_1 = \tilde{A} g_1^{\tilde{s}}, \quad t_2 = v^{\tilde{x}} g_2^{\tilde{s}}.$$

From τ_3 and τ_4, we have the following relations:

$$\hat{e}(g_1^{z_0}t_1^c, g_a g^i) \cdot \hat{e}(g_2^{z_0}t_2^c, g) \cdot \hat{e}(u,g)^{-c} = \hat{e}(g_1^{z_0'}t_1^{c'}, g_a g^i) \cdot \hat{e}(g_2^{z_0'}t_2^{c'}, g) \cdot \hat{e}(u,g)^{-c'}$$
$$\hat{e}(g_1^{-\tilde{s}}t_1, g_a g^i) \cdot \hat{e}(g_2^{-\tilde{s}}t_2, g) = \hat{e}(u,g)$$
$$\hat{e}(\tilde{A}, g_a g^i) \cdot \hat{e}(v^{\tilde{x}}, g) = \hat{e}(u,g),$$

$$\hat{e}(g_2, H_2(m))^{z_0} \cdot [\hat{e}(t_2, H_2(m)) \cdot S^{-1}]^c = \hat{e}(g_2, H_2(m))^{z_0'} \cdot [\hat{e}(t_2, H_2(m)) \cdot S^{-1}]^{c'}$$
$$\hat{e}(g_2, H_2(m))^{\tilde{s}} = \hat{e}(t_2, H_2(m)) \cdot S^{-1}$$
$$S = \hat{e}(v^{\tilde{x}}, H_2(m)).$$

Therefore we extract $(\tilde{A}, \tilde{x})$ that satisfy the required relations. □

Theorem 12. *The scheme is (ϵ, t, q_e, q_s)-unforgeable if the (ϵ', t', q)-SDH assumption holds in $\mathbb{G}$ in the random oracle model, with:*

$$t \leq t' + \Theta((q_e + q_s)\delta + q_s\tau), \quad q = q_e + 1, \quad \epsilon' \geq (\frac{\epsilon}{q_{h_1} + q_e} - \frac{1}{p})^2,$$

where q_h is the number of query to the H_1 oracle, δ and τ are the time for computing exponentiation in $\mathbb{G}$ and pairing respectively.

Proof. Assume there is a (ϵ, t, q_e, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the q-SDH problem in $\mathbb{G}$ with probability at least ϵ' and in time at most t'. $\mathcal{B}$ is given a q-SDH problem instance $(\bar{g}, \bar{g}^a, \ldots, \bar{g}^{a^q})$.

Setup. $\mathcal{B}$ randomly selects $id_1, \ldots, id_{q_{h_1}+q_e}$. Let $f(a) = \prod_{k=1}^{q-1}(a + id_k)$. $\mathcal{B}$ computes $g = \bar{g}^{f(a)}$ and $g_a = \bar{g}^{af(a)}$ using the q-SDH problem instance. $\mathcal{B}$ randomly picks $\mu, \nu \in \mathbb{Z}_p^*$ and computes $u = g^\mu$, $v = g^\nu$. $\mathcal{B}$ honestly generates the rest of the system parameters.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:
(*H oracle.*) To respond to the queries $\mathcal{B}$ maintains a list of tuples called the H_i^{list} for $i = 1, 2, 3$. Initially the lists are all empty. $\mathcal{B}$ first selects a random $j \in \{1, \ldots, q_{h_1} + q_e\}$. Let ctr be the current size of the H_1^{list}. If the H_i query already appears on the H_i^{list}, then $\mathcal{B}$ responses with the same answer. If the query is new, then:

- $H_1(\mathsf{ID}_i)$: If $ctr = j - 1$, $\mathcal{B}$ picks a random $d_i \in \mathbb{Z}_p$. If $ctr \neq j - 1$, $\mathcal{B}$ picks a new id_k not used before and sets $d_i = id_k$. Then $\mathcal{B}$ adds the tuple $\langle ctr + 1, \mathsf{ID}_i, d_i \rangle$ to the H_1^{list} and responds to $\mathcal{A}$ with $H_1(\mathsf{ID}_i) = d_i$.

- $H_2(m_i)$: $\mathcal{B}$ just picks a random string S_i in the corresponding domain and adds the tuple $\langle m_i, S_i \rangle$ to the H_2^{list}. It responds to $\mathcal{A}$ with $H_2(m_i) = S_i$.

- $H_3(T_i)$: $\mathcal{B}$ just picks a random $c_i \in \mathbb{Z}_p$ and adds the tuple $\langle T_i, c_i \rangle$ to the H_3^{list}, where $T_i = (t_0, t_1, t_2, \tau_0, \ldots, \tau_4, m, mpk, \mathsf{ID})$. It responds to $\mathcal{A}$ with $H_3(T_i) = c_i$.

(*Key Extraction oracle.*) $\mathcal{A}$ sends $(v', \mathsf{ID}_i, y_i, Pf, \Sigma)$ according to the Extract protocol. If Pf or Σ is not valid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ use the extractor of the NIZK protocol to obtain $x_i = \log_g y_i$. Then $\mathcal{B}$ first looks through list H_1^{list}.

- If ID_i is not on the list, then $\mathcal{B}$ queries $H_1(\mathsf{ID}_i)$ and obtains d_i.
- Else $\mathcal{B}$ looks for $\langle \cdot, \mathsf{ID}_i, d_i \rangle \in H_1^{list}$ and returns d_i.

If $d_i \neq id_k$ for all $k \in \{1, \ldots, q-1\}$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ computes

$$A = (uv')^{\frac{1}{a+id_k}} = \bar{g}^{\frac{f(a)(\mu - \nu x_i)}{a+id_k}}.$$

$\mathcal{B}$ returns the identity-based secret key A to the adversary.

(*Signing oracle.*) $\mathcal{B}$ runs the simulator of the signature of knowledge as in lemma 4 to obtain a valid signature σ. $\mathcal{B}$ patches the corresponding value c to the oracle H_2. $\mathcal{B}$ returns σ to $\mathcal{A}$.

Output. Finally $\mathcal{A}$ outputs a signature σ for ID^*. $\mathcal{B}$ looks for $\langle i, \mathsf{ID}^*, d^* \rangle \in H_1^{list}$. If $i \neq j$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ rewinds and obtains another signature σ'. Similar to lemma 4, $\mathcal{B}$ can extract $\tilde{A}, \tilde{x}$. Since ID^* is not queried to the key extraction oracle,

$$\tilde{A} = g^{\frac{\mu - \nu\tilde{x}}{a+d^*}} = \bar{g}^{\frac{f(a)(\mu-\nu\tilde{x})}{a+d^*}} = \bar{g}^{\sum_{k=0}^{q-2} C_k a^k + \frac{C_{-1}}{a+d^*}}$$

where $C_{-1}, C_0, \ldots, C_{q-2}$ can be computed. Notice that $C_{-1} \neq 0$ if $d^* \neq id_k$ for all $k \in \{1, \ldots, q-1\}$. Then $\mathcal{B}$ computes:

$$A^* = (\tilde{A}\bar{g}^{\sum_{k=0}^{q-2} -C_k a^k})^{1/C_{-1}},$$

and returned (A^*, d^*) as the solution to the q-SDH problem.

Probability and Time Analysis. We consider the two events that $\mathcal{B}$ could fail:

- Event E_i: For the i-th query to the $\mathcal{KEO}$, $d_i \neq id_k$ for all $k \in \{1, \ldots, q-1\}$.
- Event E^*: $d^* = id_k$ for some $k \in \{1, \ldots, q-1\}$.

Notice that the event $\cap_{i=1}^{q_e}\neg\mathsf{E}_i$ implies $\neg\mathsf{E}^*$, since the model requires that the challenge identity has never been submitted to $\mathcal{KEO}$. Hence, E^* implies $\cup_{i=1}^{q_e}\mathsf{E}_i$. Therefore the probability of $\mathcal{B}$ wins is at least $\Pr[\mathsf{E}^*] = 1/(q_{h_1}+q_e)$. By the reset lemma [19], the probability that $\mathcal{B}$ solves the q-SDH problem is $\epsilon' \geq (\epsilon/(q_{h_1}+q_e) - 1/p)^2$.

For each key extraction oracle query, $\mathcal{B}$ runs $O(1)$ exponentiation in $\mathbb{G}$. For each signing oracle query, $\mathcal{B}$ runs $O(1)$ exponentiation in $\mathbb{G}$ and $O(1)$ pairing computation. □

Theorem 13. *The scheme is (ϵ, t, q_e)-PKG non-frameable if the (ϵ', t', q)-SDH assumption holds in $\mathbb{G}$ in the random oracle model, with:*

$$t \leq t' + \Theta(q_e\delta), \quad q = q_e + 1, \quad \epsilon' \geq (\frac{\epsilon}{q_{h_1}+q_e} - \frac{1}{p})^2,$$

where q_h is the number of query to the H_1 oracle, δ is the time for computing exponentiation in $\mathbb{G}$, respectively.

Proof. Assume there is a (ϵ, t, q_e)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the q-SDH problem in $\mathbb{G}$ with probability at least ϵ' and in time at most t'. $\mathcal{B}$ is given a q-SDH problem instance $(\bar{g}, \bar{g}^a, \ldots, \bar{g}^{a^q})$.

Setup. $\mathcal{B}$ randomly selects $i_1, \ldots, i_{q-1}$. Let $f(a) = \prod_{k=1}^{q-1}(a+i_k)$. $\mathcal{B}$ computes $g = \bar{g}^{f(a)}$ and $g_a = \bar{g}^{af(a)}$ using the q-SDH problem instance. $\mathcal{B}$ randomly picks $\mu \in \mathbb{Z}_p^*$ and computes $u = g^\mu$. $\mathcal{B}$ honestly generates the rest of the system parameters param.

Oracles Simulation. $\mathcal{B}$ simulates the H oracle and the key extraction oracle as in theorem 12.

Output. Finally $\mathcal{A}$ outputs a signature σ for a message m^* and an identity ID^*. If $\mathcal{A}$ wins without querying $\mathcal{KEO}(\cdot, \mathsf{ID}^*)$, he can also win as in the unforgeability game. Then $\mathcal{B}$ can calculate the solution to the q-SDH problem as in the proof of the theorem 12. We now focus on the case that $\mathcal{A}$ has queried $\mathcal{KEO}(\cdot, \mathsf{ID}^*)$.

Denote $i^* = H_1(\mathsf{ID}^*)$. $\mathcal{A}$ also returns $upk^* = g^{x^*}$ and $\varphi^* = v^{x^*}$ to blame the PKG. $\mathcal{B}$ rewinds and obtains another signature σ'. Similar to lemma 4, $\mathcal{B}$ can extract A_σ, x_σ. Denote $upk_\sigma = g^{x_\sigma}$. Let $\mathcal{A}$ has queried $\mathcal{KEO}(upk_e, \mathsf{ID}^*)$ before. We consider the following cases:

- $upk^* = upk_\sigma$. As the claim protocol outputs "PKG", it means that $\hat{e}(v^{x^*}, H_2(m^*)) \neq S$. However by lemma 4, we have $S = \hat{e}(v, H_2(m^*))^{x_\sigma}$ which is a contradiction to $x^* = x_\sigma$.

- $upk_e = upk_\sigma \neq upk^*$. $\mathcal{B}$ retrieves the transcript $(v'_e, \mathsf{ID}^*, upk_e, Pf_e, \Sigma_e)$ and sends Σ_e (the proof of knowledge of $\log_g upk_e$ and $\log_v v'_e$) and Pf_e to the judge. $\mathcal{A}$ cannot win the game unless he can break the proof of knowledge protocol.

- $upk_e \neq upk_\sigma$. We have $A_e^{a+i^*} v^{x_e} = A_\sigma^{a+i^*} v^{x_\sigma}$. Then it means $\mathcal{A}$ wins by forging a new identity-based secret key using a different user secret key. Assume there exist an adversary $\mathcal{A}_1$ wins a game that produces a new pair (A_σ, x_σ) for the identity i^* only. In this game the adversary $\mathcal{A}_1$ firstly sends the challenge identity i^* to $\mathcal{B}_1$. $\mathcal{B}_1$ sets $g = \bar{g}$, $v = g^\nu$, $A_e = g^\omega$ and $u = g^{(a+i^*)\omega + \nu x_e}$ for some random $\nu, \omega \in_R \mathbb{Z}_p^*$. $\mathcal{B}_1$ gives the master public key and the pair (A_e, x_e) to $\mathcal{A}_1$. Finally $\mathcal{A}_1$ returns a new pair (A_σ, x_σ). Then finally $\mathcal{B}$ can compute $g^{1/(a+i^*)} = (A_e/A_\sigma)^{1/\nu(x_\sigma - x_e)}$, which is the solution to the SDH problem.

The probability and the running time of the algorithms are similar to that of theorem 12. □

Theorem 14. *The scheme is (ϵ, t, q_j, q_s)-user non-frameable if the (ϵ', t')-DL assumption holds in $\mathbb{G}$ in the random oracle model, where:*

$$t \leq t' + \Theta((q_j + q_s)\nu + q_s\tau), \quad \epsilon' \geq (\epsilon - \frac{1}{p})^2,$$

where ν and τ are the time for computing exponentiation in $\mathbb{G}$ and pairing respectively.

Proof. Assume there is a (ϵ, t, q_j, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the DL problem in $\mathbb{G}$ with probability at least ϵ' and in time at most t'. $\mathcal{B}$ is given a DL problem instance $(g, \bar{y})$.

Setup. $\mathcal{B}$ generates the public parameters for verifying Σ, randomly chooses generators $g_0, g_1, g_2 \in \mathbb{G}$ and computes $v = g^\nu$ for some random $\nu \in_R \mathbb{Z}_p$. $\mathcal{B}$ honestly generates the rest of param and sends it to the adversary $\mathcal{A}$.

$\mathcal{A}$ gives a master public key mpk to $\mathcal{B}$. $\mathcal{B}$ sets the user public key $upk^* = \bar{y}$ and generates a joining proof Pf^* for upk^*. $\mathcal{B}$ gives upk^* and Pf^* to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:
(*H oracle.*) H_1, H_2 and H_3 are simulated as normal random oracles.

(*User Join oracle.*) Upon input ID, $\mathcal{B}$ runs the simulator of the proof of knowledge to obtains a valid Σ. $\mathcal{B}$ sends $(v' = \bar{y}^\nu, \mathsf{ID}, upk^*, Pf^*, \Sigma)$ to $\mathcal{A}$. $\mathcal{A}$ returns $sk_{\mathsf{ID},upk}$.

(*Signing oracle.*) $\mathcal{B}$ runs the simulator of the signature of knowledge as in lemma 4 to obtain a valid signature σ. $\mathcal{B}$ returns σ to $\mathcal{A}$. $\mathcal{B}$ fails and exits if the challenge c is already set in the H_2 query.

Output. Finally $\mathcal{A}$ outputs a signature σ. $\mathcal{B}$ rewinds and obtains another signature σ'. Similar to lemma 4, $\mathcal{B}$ can extract $\tilde{x}$ and returns it as the solution to the DL problem.

The probability and the time analysis are straightforward. □

Theorem 15. *The scheme is (ϵ, t, q_s)-anonymous if the (ϵ', t')-DBDH assumption holds in the random oracle model, with:*

$$t \leq t' + \Theta(q_s(\delta + \tau)), \quad \epsilon' \geq (\frac{\epsilon}{q_h + q_s} - \frac{1}{p})^2,$$

where q_h is the number of query to the H_2 oracle, δ and τ are the time for computing exponentiation in $\mathbb{G}$ and pairing respectively.

Proof. Assume there is a (ϵ, t, q_s)-adversary $\mathcal{A}$. We will construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the DBDH problem with probability at least ϵ' and in time at most t'. $\mathcal{B}$ is given a DBDH problem instance (g, g^a, g^b, g^c, T).

Setup. $\mathcal{B}$ randomly selects $x_0, x_1 \in \mathbb{Z}_p$. He computes $upk_0 = g^{ax_0}$ and $upk_1 = g^{ax_1}$. $\mathcal{B}$ sets $v = g^b$. $\mathcal{B}$ honestly generates the rest of param.

Oracles Simulation. $\mathcal{B}$ simulates the oracles as follows:
(*H oracle.*) To respond to the queries $\mathcal{B}$ maintains a list of tuples called the H_i^{list} for $i = 1, 2, 3$. Initially the lists are all empty. $\mathcal{B}$ first selects a random $j \in \{1, \ldots, q_{H_2} + q_s\}$. Let ctr be the current size of the H_2^{list}. If the H_i query already appears on the H_i^{list}, then $\mathcal{B}$ responses with the same answer. If the query is new, then:

- $H_1(\mathsf{ID}_i)$: $\mathcal{B}$ just picks a random string id_i in the corresponding domain and adds the tuple $\langle \mathsf{ID}_i, id_i \rangle$ to the H_1^{list}. It responds to $\mathcal{A}$ with $H_1(\mathsf{ID}_i) = id_i$.
- $H_2(m_i)$: If $ctr \neq j - 1$, $\mathcal{B}$ picks a random $S_i \in \mathbb{G}$. If $ctr = j - 1$, $\mathcal{B}$ sets $S_i = g^c$. Then $\mathcal{B}$ adds the tuple $\langle ctr + 1, m_i, S_i \rangle$ to the H_2^{list} and responds to $\mathcal{A}$ with $H_2(m_i) = S_i$.
- $H_3(T_i)$: $\mathcal{B}$ just picks a random $c_i \in \mathbb{Z}_p$ and adds the tuple $\langle T_i, c_i \rangle$ to the H_3^{list}, where $T_i = (t_0, t_1, t_2, \tau_0, \ldots, \tau_4, m, mpk, \mathsf{ID})$. It responds to $\mathcal{A}$ with $H_3(T_i) = c_i$.

(*Signing oracle.*) $\mathcal{B}$ runs the simulator of the signature of knowledge as in lemma 4 to obtain a valid signature σ. In the process, $\mathcal{B}$ has to query $H_2(m_i)$ by himself. Finally $\mathcal{B}$ patches the corresponding value c to the oracle H_3. $\mathcal{B}$ fails and exits if c is already set. $\mathcal{B}$ returns σ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends a message m^* and an identity ID^*. If $H_2(m^*) \neq g^c$, then $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ picks a random bit b' and calculates $S^* = T^{x_b}$. $\mathcal{B}$ runs the simulator of the signature of knowledge as in lemma 4 to obtain a valid signature σ using S^*. $\mathcal{B}$ sends σ^* to $\mathcal{A}$.

Output. Finally $\mathcal{A}$ outputs a bit b^*. If $b^* = b'$, then $\mathcal{B}$ outputs $T = \hat{e}(g,g)^{abc}$ as the answer to the DBDH problem. Otherwise, $\mathcal{B}$ outputs $T \neq \hat{e}(g,g)^{abc}$.

Probability and Time Analysis. The probability of $\mathcal{B}$ exits in the challenge phase is $1 - 1/q_{h_2} + q_s$. By the reset lemma [19], the probability is $\epsilon' \geq (\epsilon/(q_{h_2} + q_s) - 1/p)^2$.

For each signing oracle query, $\mathcal{B}$ runs $O(1)$ exponentiation in $\mathbb{G}$ and $O(1)$ pairing computation. □

5.2.5 Comparison

In this section, we provide a comparison of our scheme against the existing schemes. Denote (s, P) as a pair of secret key and public key computed by the user. Denote (d, I) as a pair of identity-based secret key and identity computed by the PKG. Let (α, β) be a pair of secret key and public key of the PKG. Let c be the secret key of a certificate authority. Let $Sig_a(b)$ be a signature of message b using the secret key a. Let $Com_a(b)$ be a commitment of the value a using the public parameter b. We compare the public information that a verifier needs to know (except β), the secret keys used by the signer and the witness to link the identity with the public key. We use W to represent a witness which is different from the above parameters. The comparison of our scheme against the existing schemes is shown in Table 5.2.

Notice that the certificateless signatures, the certificate-based signatures and the self-certificated signatures aim to resolve the key escrow problem. Nonetheless, these schemes are no longer identity-based since the user public key P has been introduced into the public information. On the contrary, our scheme in section 5.2.4 is *the only scheme* that solves this problem while staying at the framework of identity-based cryptography in a strict sense. However the price we have to pay is to include a joining proof involved in the extraction protocol.

Table 5.2: Comparison of our IBS schemes against the existing schemes.

Schemes	Public Information	Secret Key	Witness
IBS [189]	I	d	-
IBS + Cert	I, P, W	s, d	$W = Sig_c(I, P)$
Certificateless Sig [4]	I, P, W	s, d	$W = Com_s(\beta)$
Certificate-based Sig [126]	I, P	s, d	$d = Sig_\alpha(I, P)$
Self-Certificated Sig [96]	I, P	s	$P = d = Sig_\alpha(I)$
Our Scheme in §5.2.2	I, P, d	s	$d = Sig_\alpha(I, P)$
Our Scheme in §5.2.4	I	s, d	$d = Sig_\alpha(I, P)$

On the other hand, our generic construction in section 5.2.2 provides a more efficient solution than our scheme in section 5.2.4. The signature of the escrow-free IBS in section 5.2.2 only consists of two standard signatures and a user public key. The computational cost of signing is the same as signing one standard signature; the computational cost of verifying is the same as verifying two standard signatures. It is as efficient as the generic IBS scheme in [17].

Our construction solves the open problem of key escrow in identity-based signatures, without requiring multiple PKGs. Our scheme is the *first* to achieve level 3 of trust of the PKG in Girault's model [96], in the identity-based setting.

5.3 Impossibility of Ideal Escrow-free IBE

The ideal solution to the key escrow problem in identity-based encryption is to avoid a malicious PKG to decrypt a ciphertext when he is given the recipient identity and the ciphertext. We first formalise the notion for ideal escrow-free IBE. After that we give a weak security model for ideal escrow-free IBE. Finally we prove that ideal escrow-free IBE is not secure even in a weak security model. Therefore it is not possible to construct a ideal escrow-free IBE scheme secure in a model which is stronger than our model defined below.

In order to avoid the key escrow problem from a powerful PKG who knows all public information, ciphertexts, identities and the master secret key, a user must have a private sequence of random coin toss which is not known by the PKG. Otherwise, the PKG can just play the role of the user and runs the **Extract** and **Dec** algorithms completely by himself. We suppose the user computes a private secret *sec* and an optional public auxiliary information *aux* from the random coin toss. We define an "ideal escrow-free IBE" as an IBE scheme which a malicious PKG cannot decrypt any ciphertext even if he knows the master secret key and the recipient identity.

5.3.1 Ideal Escrow-free IBE

An ideal escrow-free IBE scheme has five polynomial-time algorithms, namely **Setup**, **UserSetup**, **Extract**, **Enc**, **Dec**.

- **Setup**: On input a security parameter 1^λ, it generates a master secret key msk and a master public key mpk.
- **UserSetup**: On input the master public key mpk, a user generates a secret sec and an (optional) auxiliary information aux.
- **Extract**: This is an efficient two-party protocol $(\mathbf{Extract}_p, \mathbf{Extract}_u)$ between the PKG and the user[7]. The common input are mpk, aux and an identity ID. The PKG's algorithm $\mathbf{Extract}_p$ private input is msk. The user's algorithm $\mathbf{Extract}_u$ private input is sec. Additionally, the PKG and the user may use a sequence of random coin tosses as private inputs. At the end of the protocol, the user obtains the identity-based secret key sk_{ID} or outputs $\perp$ if the secret key that he receives is not valid.
- **Enc**: On input mpk, ID and a message m, it outputs a ciphertext C.
- **Dec**: On input mpk, ID, sk_{ID}, and C, it outputs a message m or outputs $\perp$ if it is not valid.

5.3.2 Security Notions

We define the *correctness* as follows:

$$\mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) = m.$$

where sk_{ID} is the output of $\mathbf{Extract}_u(\mathsf{mpk}, aux, \mathsf{ID}, sec)$ interacting with $\mathbf{Extract}_p(\mathsf{mpk}, aux, \mathsf{ID}, \mathsf{msk})$, and $(sec, aux) \leftarrow \mathbf{UserSetup}(\mathsf{mpk})$, and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$.

We define a "PKG one-wayness" (PKG-OW) security model where an adversary $\mathcal{A}$ wants to decrypt a challenge ciphertext. $\mathcal{A}$ acts as the role of a malicious PKG. Consider the following PKG-OW game:

1. The challenger $\mathcal{C}$ computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$ and $(sec, aux) \leftarrow \mathbf{UserSetup}(\mathsf{mpk})$. $\mathcal{A}$ is given (mpk, msk, aux) from the challenger $\mathcal{C}$.

[7]Two-party protocol is defined in definition 13

2. $\mathcal{C}$ picks a message m^* randomly from the message space and an identity ID^* randomly from the identity space. $\mathcal{C}$ computes the challenge ciphertext $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m^*)$.

3. $\mathcal{A}$ is given ID^* and C^*. Finally $\mathcal{A}$ outputs a message m'.

$\mathcal{A}$ wins the game if $m' = m^*$. We say that an ideal escrow-free IBE is PKG-OW secure if no polynomial time adversary wins the above game with non-negligible probability.

Notice that the above game is a relatively weak model. All parameters are chosen by the challenger, but not the adversary $\mathcal{A}$. $\mathcal{A}$ is not given any oracle access. $\mathcal{A}$ has to output m^* instead of distinguishing between two messages. We will show that $\mathcal{A}$ always wins in this model. Therefore the ideal escrow-free IBE is not possible even in this weak security model.

Theorem 16. *All ideal escrow-free IBE schemes are not PKG-OW secure.*

Proof. Notice that C^* is computed by using the input $\mathsf{mpk}, \mathsf{ID}^*, m^*$, and also the random coin toss used if Enc is a probabilistic encryption algorithm. We analyse the input as follows:

- Since mpk is the output of the $\mathbf{Setup}(1^\lambda)$ algorithm, it is independent of sec and aux.
- The identity ID^* (resp. the message m^*) is randomly chosen from the identity (resp. message) space by the definition of the PKG-OW model.
- (If Enc is not a deterministic encryption) By the PKG-OW model, C^* is computed according to the Enc algorithm. So the random coin used in Enc is picked uniformly from the corresponding domain.

Therefore C^* has no information about the sec and the aux.

By the correctness definition, we have

$$\mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}^*, sk_{\mathsf{ID}}, C^*) = m^*.$$

Therefore any valid sk_{ID} calculated from the $\mathbf{Extract}_u$ algorithm can be used to decrypt C^* and obtain the message m^*. This sk_{ID} does not need to be related to sec or aux. The adversary $\mathcal{A}$ can run both $\mathbf{Extract}_u$ and $\mathbf{Extract}_p$ by himself, by randomly picking a user secret and auxiliary information. Hence, $\mathcal{A}$ always wins the PKG-OW game. □

5.3.3 Exceptional Cases

From the above proof, we can still observe some special cases that ideal escrow-free IBE is not insecure, which are outside the PKG-OW model. We discuss the exceptions below:

- The challenge message m^* or the random coin used in **Enc** is not randomly chosen. If the encryptor knows the user auxiliary information aux when he tries to encrypt, he can use the aux as a part of the message, or as a part of the randomness in **Enc** (if **Enc** is not a deterministic encryption). Then it is possible that a malicious PKG, with the knowledge of msk and the challenge identity ID^*, is not able to decrypt the challenge ciphertext.

  For example, we let $\mathcal{E}$ be a public key encryption scheme and let pk be a public key. The user sets the aux as $(\mathcal{E}, pk)$. During **Enc**, the "message" of the ideal escrow-free IBE is the output of $\mathcal{E}_{pk}(m)$. Therefore a malicious PKG cannot obtain m without the corresponding secret key.

  However, our current definition of ideal escrow-free IBE defines that the **Enc** algorithm does not take aux as the input. Since aux is the public information provided by the user, it would violate the concept of IBE if the **Enc** algorithm takes aux as the input.

- The challenge identity ID^* is not randomly chosen. In this case a part of the user identity is dependent on the user auxiliary information aux. We can view certificate-based encryption as an example of this. However, the scheme is no longer identity-based. The encryptor needs to know the aux prior to encryption.

Therefore we can see that both of the special cases are essentially violating the concept of IBE.

5.4 IBE with Anonymity against the PKG

Following the negative result of the ideal escrow-free IBE in §5.3, it is natural to ask what the best we can do against a malicious PKG. In §5.1, we classify the existing solutions into two main categories: *preventive measure* and *blaming mechanism*. In this section, we focus on the one of the preventive measure of the key escrow problem.

Anonymous identity-based encryption, formalised by Abdalla *et al.* [1], provides privacy to the recipient since the ciphertext does not leak the identity of the recipient.

Anonymous IBE can be considered as a preventive measure of the key escrow problem. Without knowing the recipient's identity, the adversary does not know which identity-based secret key that needs to be used to decrypt. However, the traditional model for anonymous IBE [1] does not consider the case that the PKG is malicious. Therefore, we need to modify and to strengthen the security model of anonymous IBE in order to use it as a preventive measure to the key escrow problem.

Anonymous IBE can be leveraged to construct Public key Encryption with Keyword Search (PEKS) [33, 1]. PEKS allows the searching of keywords on encrypted data, and it can be used to build practical applications like encrypted and searchable audit log. Key-privacy in public key encryption [15] is suited for *anonymous communication*, since eavesdroppers are prevented from learning the public key of the target recipient. However, the sender needs to query a directory for the public key of the recipient and it may lead to the traffic analysis attack on directory lookups. Boyen and Waters [45] suggested that anonymous IBE can be used to prevent this attack since the user identity is used as the public key. Furthermore, they also suggested that *untraceable anonymous communication* is a practical and compelling application of anonymous IBE scheme [45]. However, the traditional definition for anonymous IBE does not consider the anonymity property with respect to the PKG. Nonetheless, in Boneh-Waters' anonymous IBE [45], the PKG can distinguish the recipient identity. Therefore, their anonymous IBE cannot achieve the untraceability property with respect to the PKG[8]. In order to achieve untraceable anonymous communication, we need to define a new security notion for *anonymity against the PKG*.

Using anonymous IBE to prevent the key escrow problem is useful in the practical aspects. In [102, 213], they use a tracing algorithm to detect if the PKG is malicious. In case of dispute, an honest user runs the tracing algorithm with a judge and tries to show that the PKG is malicious. Denial of service attack can be applied to the tracing algorithms in [102, 213]. The attacker, acting as some legitimate users, blames the target PKG of being dishonest by running the tracing algorithm thousands of times. However, it cannot be applied to the anonymous IBE in a similar fashion. The anonymous IBE can minimise the chance of leaking secret information (e.g. military top secret) in the first place, where the damage of leaking such secret may be too severe to be fixed. However, the anonymous IBE cannot guarantee that the system is completely secure against the PKG. If the PKG knows the recipient identity of the

[8]We note that traceability maybe desirable in some anonymous encryption schemes, such as [129]. Nevertheless, in general, when aiming for an untraceable anonymous communication, it is desirable to achieve the untraceability property with respect to the PKG as well.

ciphertext, the PKG can always generate the corresponding secret key to decrypt.

Our Contributions. Escrow-free IBE has been an open problem for decades. It is natural to ask: What is the best defense against a malicious PKG? Since the PKG should hold the master secret key and also the target ciphertext, the remaining information of the encryption algorithm in IBE are the recipient identity and the message. Therefore, we consider the following two questions:

1. If the malicious PKG does not know the recipient identity, can it distinguish the encrypted message?

2. If the malicious PKG does not know the encrypted message, can it distinguish the recipient?

The first question is closer to the real world situation. We hope that the malicious PKG does not know any information about the encrypted message. However, how can we ensure that the malicious PKG indeed has no information about the recipient identity? Therefore, the second question is as important as the first question. Although at the first glance it looks like one implies another, we can find IBE schemes that satisfy the first question but not the second (e.g. Gentry's IBE [92]).

Recently, Chow [65] formalised the first question by proposing a new security notion called "anonymous ciphertext indistinguishability" against the PKG (ACI-PKG). It is similar to the traditional IND-ID-CCA security notion [34] that the adversary has to distinguish between two messages from the challenge ciphertext. In the definition of ACI-PKG, the adversary is allowed to choose the master secret key, instead of the challenge identity in the IND-ID-CCA. Furthermore, the recipient anonymity (ANON-ID-CCA) for IBE was formalised by Abdalla *et al.* [1]. The adversary has to distinguish between two recipient identities from the challenge ciphertext. The adversary is allowed to choose the challenge message.

In this section, we formalise the second question by proposing a new security notion called "Anonymity against the PKG" (ANON-PKG). It is similar to the recipient anonymity model, except that the adversary is allowed to choose the master secret key, instead of the challenge message[9].

We believe that the four security notions mentioned above form a complete treatment of security definition for indistinguishability of messages and identities for IBE,

[9]We notice that the PKG anonymity is similar to the "key anonymity with respect to authority" (KwrtA) notion for identity-based KEM by Izabachène and Pointcheval [119]. However, Chow [65] commented that KwrtA cannot be applied directly to IBE and its real-world impact on IBE is not clear.

Table 5.3: Information that the adversary has in different security models of IBE.

	Master Secret Key	Challenge Message	Challenge Identity	Oracles
IND-ID-CCA	×	m_0, m_1	ID^*	Key Extraction, Decryption.
ANON-ID-CCA	×	m^*	ID_0, ID_1	Key Extraction, Decryption.
ACI-PKG	√	m_0, m_1	×	Encryption.
ANON-PKG	√	×	ID_0, ID_1	

even against a malicious PKG who does not know the recipient identity. It captures the combination of having the master secret key or not, and the choice of distinguishing the challenge messages or the challenge identites. In this section, we introduce the notion of anonymity against the PKG in the IBE. We formalise the four models together and give a concrete security model for IBE. This set of security notions defines full anonymity for IBE and acts a *preventive measure* to the key escrow problem.

5.4.1 Security Model for Fully Anonymous IBE

In this section, we propose a new security notion for IBE which provides anonymity even against malicious PKG. We combine the security notions for IND-ID-CCA [34], recipient anonymity [1], anonymous ciphertext indistinguishability [65] and our own definition for anonymity against the PKG. We name this new model as "Fully Anonymous IBE".

An IBE scheme has four polynomial-time algorithms, namely **Setup**, **Extract**, **Enc**, **Dec**.

- **Setup**: On input a security parameter 1^λ, it generates a master public key mpk and a master secret key msk.
- **Extract**: On input msk and an identity ID, it outputs an identity-based secret key sk_{ID}.
- **Enc**: On input mpk, ID and a message m from the message space $\mathcal{M}$, it outputs a ciphertext C.
- **Dec**: On input sk_{ID}, and C, it outputs a message m or outputs $\perp$ if it is invalid.

Correctness. We define the *correctness* as:

$$\mathbf{Dec}(sk_{\mathsf{ID}}, \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) = m,$$

for all message $m \in \mathcal{M}$, where $sk_{\mathsf{ID}} \leftarrow \mathbf{Extract}(\mathsf{msk}, \mathsf{ID})$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$.

Security Model for Confidentiality

The security model for confidentiality captures the attack from an adversary who wants to distinguish which message is encrypted in the ciphertext.We have the following IND-ID-CCA game for confidentiality:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives mpk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to query the oracle adaptively:
   - Key Extraction Oracle $\mathcal{KEO}(\mathsf{ID})$: On input an identity ID, it returns the identity-based secret key $sk_{\mathsf{ID}} \leftarrow \mathbf{Extract}(\mathsf{msk}, \mathsf{ID})$.
   - Decryption Oracle $\mathcal{DO}(C, \mathsf{ID})$: On input a ciphertext C and an identity ID, it returns a message $m/\perp \leftarrow \mathbf{Dec}(sk_{\mathsf{ID}}, C)$.
3. $\mathcal{A}$ sends two messages $m_0, m_1 \in \mathcal{M}$ and an identity ID^* to the challenger. The challenger picks a random bit b' and computes $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m_{b'})$. The challenger sends C^* to $\mathcal{A}$.
4. $\mathcal{A}$ is allowed to query the above oracles adaptively.
5. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. We require that there was no query that $\mathcal{KEO}(\mathsf{ID}^*)$ and $\mathcal{DO}(C^*, \mathsf{ID}^*)$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game minus 1/2. An IBE scheme is (ϵ, t, q_k, q_d)-IND-ID-CCA secure if there is no t time adversary has advantage at least ϵ with q_k and q_d queries to $\mathcal{KEO}$ and $\mathcal{DO}$, respectively.

Security Model for Recipient Anonymity

The security model for recipient anonymity captures the attack from an adversary who wants to distinguish the recipient identity from a ciphertext. We have the following ANON-ID-CCA game for recipient anonymity:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives mpk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to query the key extraction oracle and the decryption oracle (as defined in the IND-ID-CCA game) adaptively.

3. $\mathcal{A}$ sends a message $m^* \in \mathcal{M}$ and two identities $\mathsf{ID}_0, \mathsf{ID}_1$ to the challenger. The challenger picks a random bit b' and computes $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}_{b'}, m^*)$. $\mathcal{S}$ sends C^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the above oracles adaptively.

5. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. We require that there was no query that $\mathcal{KEO}(\mathsf{ID}_b)$ and $\mathcal{DO}(C^*, \mathsf{ID}_b)$ for $b = 0/1$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game minus 1/2. An IBE scheme is (ϵ, t, q_k, q_d)-ANON-ID-CCA secure if there is no t time adversary has advantage at least ϵ with q_k and q_d queries to $\mathcal{KEO}$ and $\mathcal{DO}$, respectively.

Security Model for Anonymous Ciphertext Indistinguishability

The security model for anonymous ciphertext indistinguishability captures the attack from an adversary (acting as a PKG) who wants to distinguish which message is encrypted in the ciphertext, with the knowledge of the master secret key. We have the following ACI-PKG game for anonymous ciphertext indistinguishability against the PKG:

1. The adversary $\mathcal{A}$ gives mpk to the challenger. The challenger aborts if mpk is not valid. Otherwise, it chooses an identity ID^* at random.

2. $\mathcal{A}$ is allowed to adaptively query the Encryption Oracle $\mathcal{EO}(m)$: for any message $m \in \mathcal{M}$, it returns a ciphertext $C \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m)$.

3. $\mathcal{A}$ sends two messages $m_0, m_1 \in \mathcal{M}$ to the challenger. The challenger picks a random bit b' and computes $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m_{b'})$. It sends C^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the above oracle adaptively.

5. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game minus 1/2. An IBE scheme is (ϵ, t, q_e)-ACI-PKG secure if there is no t time adversary has advantage at least ϵ with q_e queries to $\mathcal{EO}$.

Improvement over Chow's ACI-PKG Model In the traditional IBE definition and in our model, the PKG only outputs mpk and msk. In Chow's IBE model [65], the PKG also outputs a system parameters param. The system parameters param are always generated by the challenger in the security game in [65]. It implies that the system parameters are always trusted in Chow's model. Therefore our current model for anonymous ciphertext indistinguishability is stronger than that of Chow's model.

We further show that Chow's version of Gentry-IBE scheme is not secure in our ACI-PKG model. We briefly review a part of their scheme as follows:

- **Setup**: The (trusted) initialiser picks (g, h_1, h_2, h_3) randomly from $\mathbb{G}$ and chooses a hash function H from a family of universal one-way hash function. The system parameters param are (g, h_1, h_2, h_3, H). The PKG chooses an exponent α randomly from $\mathbb{Z}_p$. The PKG sets $\mathsf{mpk} = g_1 = g^\alpha$ and $\mathsf{msk} = \alpha$.

- **Enc**: To encrypt $m \in \mathbb{G}_T$ for identity $\mathsf{ID} \in \mathbb{Z}_p$, the sender picks a random $r \in \mathbb{Z}_p$ and computes the ciphertext:

$$u = (g_1 g^{-\mathsf{ID}})^r, \quad v = \hat{e}(g,g)^r, \quad w = m/\hat{e}(g,h_1)^r, \quad y = \hat{e}(g,h_2)^r \hat{e}(g,h_3)^{r \cdot H(u,v,w)}.$$

If the systems parameters param are also generated by the malicious PKG, then the malicious PKG knows $\beta = \log_g h_1$. When it receives the challenger ciphertext $C^* = (u^*, v^*, w^*, y^*)$, it can check if the challenge message m_0^* or m_1^* is equal to $w^* v^{*\beta}$, without knowing the challenge identity.

Security Model for Anonymity against the PKG

The security model for anonymity against the PKG captures the attack from an adversary (acting as a PKG) who wants to distinguish the recipient identity from a ciphertext, with the knowledge of the master secret key. We have the following ANON-PKG game for anonymity against the PKG:

1. The adversary $\mathcal{A}$ gives mpk and two identities $\mathsf{ID}_0, \mathsf{ID}_1$ to the challenger. The challenger aborts if mpk is not valid.

2. The challenger chooses a random message m^* from the message space $\mathcal{M}$. The challenger picks a random bit b' and computes $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}_{b'}, m^*)$. It sends C^* to $\mathcal{A}$.

3. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game minus 1/2. An IBE scheme is (ϵ, t)-ANON-PKG secure if there is no t time adversary has advantage at least ϵ.

Note that the challenger does not need to provide the decryption oracle or the key extraction oracle to the adversary, since the adversary can calculate the secret key of the challenge identities and can decrypt by itself.

5.4.2 Our First Construction

In this Chapter, we will give two concrete constructions of fully anonymous identity based encryption. We prove the security of our constructions and compare with the existing IBE schemes in the literature. The first construction has a tight security reduction to a standard intractability assumption. However, this construction has a large number of public parameters. We can use the structure of the extraction of identity-based secret key to construct a new signature in the standard model. The second construction is reduced to a standard intractability assumption and has (small) constant number of public parameters. However, the security reduction is not tight. We now give our first construction as follows.

Intuition behind the Construction

We observe that most existing IBE schemes do not achieve anonymity against the PKG because their ciphertexts contain information about the recipient identity, such that the PKG can check the identity using pairings. For example in the Waters-IBE [206], the ciphertext is

$$C = \big(\quad c_1 = m \cdot \hat{e}(g_1, g_2)^t, \quad c_2 = g^t, \quad c_3 = (u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i})^t \quad \big),$$

where the user identity is $\{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$. Anyone (including the PKG) can check if $\hat{e}(g, c_3) = \hat{e}(c_2, u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i})$. On the other hand, the BF-IBE [34] achieves anonymity against the PKG and the ciphertext is

$$C = \big(\quad c_1 = m \cdot \hat{e}(H(\mathsf{ID}), g^a)^t, \quad c_2 = g^t \quad \big),$$

where the user identity is ID. We observe that if the internal session key (e.g. $\hat{e}(H(\mathsf{ID}), g^a)^t$ in the BF-IBE) consists of the user identity but not the rest of the ciphertext,

then the IBE scheme is possibly anonymous against the PKG. However, there is no IBE scheme in the standard model which has this kind of ciphertext structure.

As a result, we need a different identity-based secret key extraction algorithm to achieve anonymity against the PKG in the standard model. Suppose the master secret key is x and the master public key includes $g_1, u_0, \ldots, u_n \in \mathbb{G}_1,\ g_2, g_2^x \in \mathbb{G}_2$ and a pseudorandom function G_K that maps arbitrary strings to n-bit strings (refer to §3.1.4 for the definition). Denote the user identity as ID. Let $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ and r is a random number in $\mathbb{Z}_p$. Then the identity-based secret key is (d, r), where

$$d = (g_1^r u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i})^{1/x}.$$

To encrypt a message m, the sender picks a random $s \in \mathbb{Z}_p$ and calculate the ciphertext

$$C = \big(\quad c_1 = (g_2^x)^s, \quad c_2 = \hat{e}(g_1, g_2)^s, \quad c_3 = m \cdot \hat{e}(u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i}, g_2)^s \quad \big).$$

The internal session key, $\hat{e}(u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i}, g_2)^s$, contains the user identity but the identity is not in the rest of the ciphertext. We observe that our security proof has a significant difference with all existing IBE schemes. Denote q as the number of key extraction oracle query. The existing IBE schemes either have an IND-ID-CCA security reduction loss proportional to q (e.g. BF-IBE [34], Waters-IBE [206]), or reduced to an intractability assumption which has a size of q (e.g. SK-IBE [184], Gentry-IBE [92]). Our scheme has a tight security reduction and the IND-ID-CCA security is reduced to the decisional bilinear Diffie-Hellman Inversion problem. However, the size of the public parameters is proportional to q. Although our IBE scheme is not efficient, it offers a tight security reduction to a standard intractability assumption. This stands in sharp contrast to the existing IBE schemes. Our IBE scheme can be viewed as a proof of the existence of IBE anonymous against the PKG in the standard model.

New Intractability Assumption

We introduce a new candidate hard problem inspired by the decisional bilinear problem [65] and the known-target RSA inversion problem [18]. It will be used in the security theorem.

Known-Target Decisional Bilinear Problem. The ℓ-Known-Target Decisional Bilinear problem states: the adversary $\mathcal{A}$ gives ℓ elements $h_1, \ldots, h_\ell \in \mathbb{G}_2$. After that the adversary is given $h_1^a, \ldots, h_\ell^a \in \mathbb{G}_2$ and $W, Z \in \mathbb{G}_T$, for some random unknown $a \in \mathbb{Z}_p^*$.

The adversary outputs 1 if $Z = W^a$, and outputs 0 otherwise. Consider the following experiment:

Experiment $\mathbf{Exp}^{ktdb}_{\mathcal{A},\ell}(k, b)$:

1. Use the security parameter k to generate the pairing groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.
2. An algorithm $\mathcal{A}$ is given the pairing groups and returns ℓ elements $h_1, \ldots, h_\ell \in \mathbb{G}_2$.
3. Pick a random $a \in_R \mathbb{Z}_p^*$ and $W, Z \in_R \mathbb{G}_T$.
   - If the bit $b = 0$, the algorithm $\mathcal{A}$ is given $(h_1^a, \ldots, h_\ell^a, W, Z)$.
   - If the bit $b = 1$, the algorithm $\mathcal{A}$ is given $(h_1^a, \ldots, h_\ell^a, W, W^a)$.
4. Finally the algorithm $\mathcal{A}$ outputs a bit b'. This is the output of the experiment.

$\mathcal{A}$ has advantage ϵ in solving the ℓ-known-target decisional bilinear problem if

$$\left|\Pr[\mathbf{Exp}^{ktdb}_{\mathcal{A},\ell}(k, 1) = 1] - \Pr[\mathbf{Exp}^{ktdb}_{\mathcal{A},\ell}(k, 0) = 1]\right| \geq \epsilon,$$

where the probability is over the random choice of $W, Z \in \mathbb{G}_T$, the random choice of $a \in \mathbb{Z}_p^*$, and the random bits consumed by $\mathcal{A}$.

We say that the (ϵ, t)-ℓ-known-target decisional bilinear assumption holds in ($\mathbb{G}_2$, $\mathbb{G}_T$) if no t-time algorithm has advantage at least ϵ in solving the ℓ-known-target decisional bilinear problem.

Note that if $\ell = 1$ then the known-target decisional bilinear assumption is equivalent to the decisional bilinear assumption [65]. It is similar to the relationship between the known-target RSA Inversion assumption [18] and the standard RSA assumption. In this section, we need to use the known-target decisional bilinear assumption where $\ell = 2$ only. It is different from schemes which use the q-SDH assumption, where q is proportional to the number of oracle queries.

CPA-secure Construction

We construct our fully anonymous IBE scheme as follows:

- **Setup**: The setup algorithm selects bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing. Let n be an security parameter derived from 1^λ. The message space $\mathcal{M}$ is $\mathbb{G}_T$. The algorithm picks some random generators $g_1, u_0, u_1, \ldots, u_n \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. The algorithm also picks a random $x \leftarrow_R \mathbb{Z}_p$,

and computes $v = g_2^x$ and $z = \hat{e}(g_1, g_2)$. The algorithm chooses a random key K for the pseudorandom function $G_K : \{0,1\}^* \to \{0,1\}^n$. The master public keys are $(g_1, g_2, u_0, u_1, \ldots, u_n, v, z, K)$. The master secret key is x.

- **Extract**: Given a master secret key x and an identity ID, pick a random $r \in \mathbb{Z}_p^*$ and calculate $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\} \in \{0,1\}^*$ and

$$d = (g_1^r u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i})^{1/x}.$$

  Output the identity-based secret key $sk_{\mathsf{ID}} = (d, r)$. We require that the PKG always use the same random value r for the same ID. This can be achieved, for example, by using a pseudorandom function.

- **Encrypt**: To encrypt a message $m \in \mathbb{G}_T$ for a user ID, the sender picks a random $s \in \mathbb{Z}_p^*$ and calculates $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ and:

$$c_1 = v^s, \quad c_2 = z^s, \quad c_3 = m \cdot \hat{e}(u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i}, g_2)^s.$$

  The ciphertext is $C = (c_1, c_2, c_3)$. Notice that the encryption algorithm does not require any pairing computation once $\hat{e}(u_i, g_2)$ for $i = 0, \ldots, n$ are pre-computed.

- **Decrypt**: Given a ciphertext $C = (c_1, c_2, c_3)$, and a secret key $sk_{\mathsf{ID}} = (d, r)$ for an identity ID, the recipient outputs:

$$m = \frac{c_3 \cdot c_2^r}{\hat{e}(d, c_1)}.$$

The correctness of the above construction is straightforward.

Theorem 17. *Suppose the (ϵ, t')-decisional BDHI assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the above IBE scheme is (ϵ, t, q_k)-IND-ID-CPA secure provided that*

$$q_k < \frac{n+1}{2}, \quad \textit{and} \quad t \leq t' - O((q_k + n)T),$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$.

Proof. Suppose there is an IND-ID-CPA adversary $\mathcal{A}$ that (ϵ, t, q_k)-breaks the above IBE scheme. We construct an algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the decisional BDHI problem in time t' with advantage ϵ. $\mathcal{B}$ is given a random instance $(g_1, g_2, g_1^x, g_2^x, T)$ of the problem instance, for some unknown $x \in \mathbb{Z}_p$. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ sets the public key as follows:

1. Choose a random key K for the PRF function G_K.
2. Select some random $\mu_0, \ldots, \mu_n, \nu_0, \ldots, \nu_n \in \mathbb{Z}_p^*$. For $i = 0, \ldots, n$, set $u_i = g_1^{\mu_i} g_1^{x\nu_i}$. For any identity ID, calculate $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$. Denote $F(\mathsf{ID}) = \mu_0 + \sum_{i=1}^n \mu_i \mathsf{id}_i$ and $J(\mathsf{ID}) = \nu_0 + \sum_{i=1}^n \nu_i \mathsf{id}_i$. Therefore, we have:
$$u_0 \prod_{i=1}^n u_i^{\mathsf{id}_i} = g_1^{\mu_0 + \sum_{i=1}^n \mu_i \mathsf{id}_i} (g_1^x)^{\nu_0 + \sum_{i=1}^n \nu_i \mathsf{id}_i} = g_1^{F(\mathsf{ID})} (g_1^x)^{J(\mathsf{ID})}.$$
3. Set $v = g_2^x$ and calculate $z = \hat{e}(g_1, g_2)$.
4. The public key is $(g_1, g_2, u_0, u_1, \ldots, u_n, v, z, K)$. The corresponding secret key is x.

Oracles Simulation. $\mathcal{B}$ simulates the key extraction oracle as follows. The adversary $\mathcal{A}$ issues a query for an identity ID. $\mathcal{B}$ sets $r = -F(\mathsf{ID})$ and $\sigma = g_1^{J(\mathsf{ID})}$. Observe that (σ, r) is a valid signature:
$$d = (g_1^{-F(\mathsf{ID})} (g_1^{F(\mathsf{ID})} (g_1^x)^{J(\mathsf{ID})}))^{1/x} = g_1^{J(\mathsf{ID})}.$$
$\mathcal{B}$ returns $sk_{\mathsf{ID}} = (d, r)$ as the output.

Challenge: Eventually, $\mathcal{A}$ will output two messages (m_0^*, m_1^*) and a challenge identity ID^*. $\mathcal{B}$ picks a random bit b' and a random $s^* \in \mathbb{Z}_p^*$. $\mathcal{B}$ calculates:
$$c_1^* = g_2^{s^*}, \quad c_2^* = Z^{s^*}, \quad c_3^* = m_{b'}^* \cdot Z^{F(\mathsf{ID}^*)s^*} \cdot \hat{e}(g_1, g_2)^{J(\mathsf{ID}^*)s^*}.$$
$\mathcal{B}$ returns the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$.

Guess: Finally $\mathcal{A}$ outputs its guess $b^* \in \{0, 1\}$. If $b^* = b'$, $\mathcal{B}$ outputs 1; otherwise, it outputs 0.

Analysis: We now argue that any adversary $\mathcal{A}$ against the signature scheme will have success in the game presented by $\mathcal{B}$. We define a sequence of games and show that if $\mathcal{A}$ is successful against Game j, then it will also be successful against Game $j+1$. The first game models the real security game and the last game is exactly the view of $\mathcal{A}$ when interacting with $\mathcal{B}$.

Game 1: The game is defined to be the same as the security game.

Game 2: The same as Game 1, with the exception that at the beginning of the game $\mathcal{B}$ sets $u_i = g_1^{\mu_i} g_1^{x\nu_i}$ and $v = g_2^x$.

Game 3: The same as Game 2, with the exception that $\mathcal{B}$ answers the key extraction oracle as described in the above simulation.

Game 4: The same as Game 3, with the exception that $\mathcal{B}$ returns the challenge ciphertext as described in the above simulation.

Define $\mathbf{Adv}_{\mathcal{A}}[\text{Game } x]$ as the advantage of $\mathcal{A}$ in Game x. We have the following claims that will lead to the main theorem.

Claim 1.

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 1}].$$

Proof. These games are identical. The only difference is in the presentation of how u_i are chosen. In Game 1, they are randomly chosen from $\mathbb{G}_1$. In Game 2, $u_i = g_1^{\mu_i} g_1^{x\nu_i}$ where μ_i and ν_i are randomly chosen from $\mathbb{Z}_p$. In both games, u_i are uniformly distributed in $\mathbb{G}_1$.

□

Claim 2. *If G is a secure pseudorandom function and $q_k < (n+1)/2$, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 2}].$$

Proof. We analysis the distribution of the key extraction oracle output by $\mathcal{B}$. $\mathcal{B}$ outputs (d_i, r_i) such that $d_i = g_1^{J(\mathsf{ID}_i)}$ and $r_i = -F(\mathsf{ID}_i)$. Observe that $J(\mathtt{M}_i)$ is a polynomial with $n+1$ random variables $\nu_0, \ldots, \nu_n$ unknown to $\mathcal{A}$; and $F(\mathsf{ID}_i)$ is a polynomial with $n+1$ random variables $\mu_0, \ldots, \mu_n$ unknown to $\mathcal{A}$. We represent the knowledge gained by the adversary as a matrix product:

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{q_k} \end{bmatrix} = \begin{bmatrix} 1 & \mathsf{id}_{1,1} & \ldots & \mathsf{id}_{1,n} \\ 1 & \mathsf{id}_{2,1} & \ldots & \mathsf{id}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathsf{id}_{q_k,1} & \ldots & \mathsf{id}_{q_k,n} \end{bmatrix} \cdot \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \begin{bmatrix} \log_{g_1} d_1 \\ \log_{g_1} d_2 \\ \vdots \\ \log_{g_1} d_{q_k} \end{bmatrix} = \begin{bmatrix} 1 & \mathsf{id}_{1,1} & \ldots & \mathsf{id}_{1,n} \\ 1 & \mathsf{id}_{2,1} & \ldots & \mathsf{id}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathsf{id}_{q_k,1} & \ldots & \mathsf{id}_{q_k,n} \end{bmatrix} \cdot \begin{bmatrix} \nu_0 \\ \nu_1 \\ \vdots \\ \nu_n \end{bmatrix},$$

where $G_K(\mathsf{ID}_i) = \{\mathsf{id}_{i,1}, \ldots, \mathsf{id}_{i,n}\}$ for $i = 1, \ldots, q_e$. On input an identity ID_i, the output of $G_K(\mathsf{ID}_i)$ is a random n-nit string by the pseudorandomness property of the PRF (Otherwise, it would admit a distinguishing attack against the PRF). If $q_k < n+1$, the number of columns is more than the number of rows in the above matrix. Therefore the distribution of the key extraction output is indistinguishable from the real output.

Furthermore, u_i also contains information about μ_i and ν_i. We analysis on the distribution of u_i and the key extraction oracle output. We represent the knowledge

gained by the adversary as a matrix product:

$$\begin{bmatrix} \log_{g_1} u_0 \\ \log_{g_1} u_1 \\ \vdots \\ \log_{g_1} u_n \\ \log_{g_1} d_1 \\ r_1 \\ \vdots \\ \log_{g_1} d_{q_k} \\ r_{q_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 & x & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & x & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & x \\ 0 & 0 & \dots & 0 & 1 & \mathsf{id}_{1,1} & \dots & \mathsf{id}_{1,n} \\ 1 & \mathsf{id}_{1,1} & \dots & \mathsf{id}_{1,n} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 & \mathsf{id}_{q_k,1} & \dots & \mathsf{id}_{q_k,n} \\ 1 & \mathsf{id}_{q_k,1} & \dots & \mathsf{id}_{q_k,n} & 0 & 0 & \dots & 0 \end{bmatrix} \cdot \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_n \\ \nu_0 \\ \nu_1 \\ \vdots \\ \nu_n \end{bmatrix}.$$

Therefore the adversary has $2q_k + n + 1$ equations with $2n + 2$ random variables. By applying Gaussian elimination to the above $(2q_k + n + 1)$-by-$(2n + 2)$ matrix, the rank of the matrix is is $2q_k + n + 1$. Therefore, the $2q_k + n + 1$ rows are linearly independent. Since we restrict that $q_k < (n + 1)/2$, the number of columns is more than the number of rows in the above matrix. Therefore, the distribution of u_i and the key extraction oracle outputs are indistinguishable from the real output from the Extract protocol. □

Claim 3.

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 4}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 3}].$$

Proof. Note that the challenge ciphertext C^* in Game 4 is a valid ciphertext with respect to the randomness $\tilde{s} = s^*/x$ if $Z = \hat{e}(g_1, g_2)^{1/x}$:

$$\begin{aligned} c_1^* &= g_2^{\tilde{s}x} = v^{\tilde{s}}, \\ c_2^* &= \hat{e}(g_1, g_2)^{\tilde{s}} = z^{\tilde{s}}, \\ c_3^* &= m_{b'}^* \cdot \hat{e}(g_1, g_2)^{F(\mathsf{ID}^*)\tilde{s}} \cdot \hat{e}(g_1, g_2)^{J(\mathsf{ID}^*)\tilde{s}x} = m_b^* \cdot \hat{e}(g_1^{F(\mathsf{ID}^*)+xJ(\mathsf{ID}^*)}, g_2)^{\tilde{s}}. \end{aligned}$$

Therefore the challenge ciphertext in Game 4 is indistinguishable from the challenge ciphertext in Game 3. □

□

Theorem 18. *Suppose the (ϵ, t')-decisional BDHI assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the above IBE scheme is (ϵ, t, q_k)-ANON-ID-CPA secure provided that*

$$q_k < \frac{n+1}{2}, \quad \textit{and} \qquad t \leq t' - O((q_k + n)T),$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$.

Proof. (Sketch) The proof is almost the same as the IND-ID-CPA proof. The only difference is during the challenge phase, the adversary $\mathcal{A}$ outputs a challenge message m^* and two challenge identity $(\mathsf{ID}_0^*, \mathsf{ID}_1^*)$. $\mathcal{B}$ picks a random bit b' and a random $s^* \in \mathbb{Z}_p^*$. $\mathcal{B}$ calculates:

$$c_1^* = g_2^{s^*}, \quad c_2^* = Z^{s^*}, \quad c_3^* = m^* \cdot Z^{F(\mathsf{ID}_{b'}^*)s^*} \cdot \hat{e}(g_1, g_2)^{J(\mathsf{ID}_{b'}^*)s^*}.$$

$\mathcal{B}$ returns the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$. Finally $\mathcal{A}$ outputs its guess $b^* \in \{0, 1\}$. If $b^* = b'$, $\mathcal{B}$ outputs 1; otherwise, it outputs 0. □

Theorem 19. *Suppose the (ϵ, t')-2-known-target decisional bilinear assumption holds in $(\mathbb{G}_2, \mathbb{G}_T)$. Then the above IBE scheme is (ϵ, t, q_e)-ACI-PKG secure provided that $t \leq t' - O((q_e + n)T)$, where T is the maximum time for an exponentiation in $\mathbb{G}_2$ and $\mathbb{G}_T$.*

Proof. Suppose there is an adversary $\mathcal{A}$ who (ϵ, t, q_e)-breaks the ACI-PKG security. We construct an algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the 2-known-target decisional bilinear problem in time t' with advantage ϵ. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows:

Setup: The adversary $\mathcal{A}$ gives the master public key mpk to $\mathcal{B}$. $\mathcal{B}$ aborts if the mpk is not valid. $\mathcal{B}$ gives v and g_2 to the challenger of the 2-known-target decisional bilinear problem instance. After that, $\mathcal{B}$ obtains $a_1 = v^a, a_2 = g_2^a$, and $W, Z \in \mathbb{G}_T$ from the problem instance.

Oracles Simulation. $\mathcal{B}$ simulates the encryption oracle as follows. On input a message m in the message space $\mathcal{M}$, $\mathcal{B}$ picks a random $s \in \mathbb{Z}_p^*$ and returns the ciphertext:

$$C = \left(c_1 = v^s, \quad c_2 = z^s, \quad c_3 = m \cdot W^s\right)$$

Notice that this encryption oracle output implicitly sets the $\{\mathsf{id}_1^*, \ldots, \mathsf{id}_n^*\}$ such that $\hat{e}(u_0 \prod_{i=1}^n u_i^{\mathsf{id}_i^*}, g_2) = W$. It implies the the challenge identity is ID^* such that $G_K(\mathsf{ID}^*) = \{\mathsf{id}_1^*, \ldots, \mathsf{id}_n^*\}$. ID^* and $\{\mathsf{id}_1^*, \ldots, \mathsf{id}_n^*\}$ are not known to both $\mathcal{A}$ and $\mathcal{B}$. Even if such ID^* does not exist according to the definition of G_K, the adversary $\mathcal{A}$ cannot discover it by the pseudorandomness property of the output of G_K.

Challenge: Eventually, $\mathcal{A}$ will output two messages (m_0^*, m_1^*). $\mathcal{B}$ picks a random bit b' and a random $s^* \in \mathbb{Z}_p^*$. $\mathcal{B}$ returns the ciphertext:

$$C^* = \left(c_1 = a_1^{s^*}, \quad c_2 = \hat{e}(g_1, a_2)^{s^*}, \quad c_3 = m_{b'} \cdot Z^{s^*}\right)$$

Guess: Finally $\mathcal{A}$ outputs its guess $b^* \in \{0,1\}$. If $b^* = b'$, $\mathcal{B}$ outputs 1; otherwise, it outputs 0.

Analysis: It is straightforward to see that if $\mathcal{A}$ has ϵ advantage, $\mathcal{B}$ solves the known-target decisional bilinear problem with probability ϵ. □

Theorem 20. *The above IBE scheme is information theoretically secure against the ANON-PKG attack.*

Proof. In the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$, c_1^* and c_2^* does not contain any information about the user identity. On the other hand, c_3^* appears random to the adversary $\mathcal{A}$ since it contains the challenge message m^* which is unknown to $\mathcal{A}$. □

CCA-secure Construction

To convert our CPA-secure IBE scheme into a CCA-secure construction, a trivial way is to use the conversion method in Gentry-IBE [92]. It would give us an IND-ID-CCA secure scheme. However, the extra element in the ciphertext has some information about the recipient identity. The resulting scheme is no longer ANON-PKG secure.

As a result, we need to use an alternative approach. We can build a hybrid 2-level hierarchical IBE (HIBE) scheme that uses our IBE scheme at the first level and the scheme of Boneh and Boyen (BB_1-IBE) [27] at the second level. Boneh *et al.* [32] showed how to build a CCA-secure IBE scheme from a 2-level HIBE scheme.

Let $\mathsf{Sig} = (\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme in which the verification key output by $\mathcal{G}(1^\lambda)$ is in $\mathbb{Z}_p$.

- **Setup**: The setup algorithm selects bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing. Let n be an security parameter derived from 1^λ. The PKG picks random generators $g_1, u_0, u_1, \ldots, u_n, g_3, g_4 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. The PKG also picks random $x \leftarrow_R \mathbb{Z}_p$, and computes $v = g_2^x$ and $z = \hat{e}(g_1, g_2)$. Choose a random key K for the pseudorandom function $G_K : \{0,1\}^* \leftarrow \{0,1\}^n$. The master public keys are $(g_1, g_2, g_3, g_4, u_0, u_1, \ldots, u_n, v, z, K)$. The master secret key is x.

- **Extract**: Given a master secret key x and an identity ID, pick a random $r \in \mathbb{Z}_p^*$ and calculate $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\} \in \{0,1\}^*$ and

$$d = (g_1^r u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i})^{1/x}.$$

Output the identity-based secret key $sk_{\mathsf{ID}} = (d, r)$. We require that the PKG always use the same random value r for the same ID. This can be achieved, for example, by using a pseudorandom function.

- **Encrypt**: To encrypt a message $m \in \mathbb{G}_T$ for a user ID, the sender picks a random $s \in \mathbb{Z}_p^*$ and calculates $G_K(\mathsf{ID}) = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$. The sender runs $(sk, vk) \leftarrow \mathcal{G}(1^\lambda)$ and calculates:

$$c_1 = v^s, \quad c_2 = z^s, \quad c_3 = m \cdot \hat{e}(u_0 \prod_{i=1}^{n} u_i^{\mathsf{id}_i}, g_2)^s, \quad c_4 = (g_3^{vk} g_4)^s.$$

  Denote $C = (c_1, c_2, c_3, c_4)$. The sender signs $\sigma \leftarrow \mathsf{Sign}_{sk}(C)$ and sends the ciphertext (C, σ, vk). Notice that the encryption algorithm does not require any pairing computation once $\hat{e}(u_i, g_2)$ for $i = 0, \ldots, n$ are pre-computed.

- **Decrypt**: Given a ciphertext $(C = (c_1, c_2, c_3, c_4), \sigma, vk)$, and a secret key (d, r) for an identity ID, the recipient outputs $\perp$ if $\mathsf{Vrfy}_{vk}(C, \sigma) \neq 1$. Otherwise, the recipient calculates:

$$m = \frac{c_3 \cdot c_2^r \cdot \hat{e}(c_4, v)}{\hat{e}(d, c_1)}.$$

The IND-ID-CCA security follows from the theorem in [32]. We can show that our scheme achieves ANON-PKG. Consider an adversary $\mathcal{A}$ in the ANON-PKG game choosing the challenge identities $\mathsf{id}_0, \mathsf{id}_1$. $\mathcal{A}$ can try to use the identity-based secret key of both challenge identities to decrypt the challenge ciphertext (C^*, σ^*, vk^*). Note that the decryption will not output $\perp$ since σ^* is a valid one-time signature. Therefore the decryption of the challenge ciphertext will always give a message. Therefore the scheme achieves ANON-PKG if the message space is dense.

Signature Based on the DHI Problem

Naor observed any IBE scheme can be converted into a signature scheme with the identity-based secret key playing the role of signatures [34]. The key extraction algorithm of our first fully anonymous IBE is based on a new signature scheme based on the DHI problem. The signature size is as small as the short signatures by Boneh and Boyen [29], and is significantly shorter than the Waters signatures [206]. Unlike the Boneh and Boyen signatures, the security of our signatures is reduced to a standard intractability assumption: the Diffie-Hellman Inversion assumption. However, the trade-off is that the size of the public parameters is proportional to q_s, where q_s is the number of signing oracle query and is about 2^{30} in common settings.

- **Setup**: The setup algorithm selects bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing. Let n be an security parameter derived from 1^λ. It picks random generators $g_1, u_0, u_1, \ldots, u_n \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. It chooses a random key K for the pseudorandom function $G_K : \{0,1\}^* \to \{0,1\}^n$. It picks a random $x \in \mathbb{Z}_p$, and computes $v = g_2^x$ and $z = \hat{e}(g_1, g_2)$. The public key is $(g_1, g_2, u_0, u_1, \ldots, u_n, v, z, K)$. The secret key is x.

- **Sign**: Given a secret key x and a message $m \in \{0,1\}^*$, the signer picks a random $r \in \mathbb{Z}_p^*$ and calculate $G_K(m) = \{m_1, \ldots, m_n\}$ and $\sigma = (g_1^r u_0 \prod_{i=1}^n u_i^{m_i})^{1/x}$. The signer outputs the signature as (σ, r).

- **Verify**: Given a public key $(g_1, g_2, u_0, u_1, \ldots, u_n, v, z)$, a message m, and a signature (σ, r), verify that $\hat{e}(\sigma, g_2^x) = z^r \cdot \hat{e}(u_0 \prod_{i=1}^n u_i^{m_i}, g_2)$, where $G_K(m) = \{m_1, \ldots, m_n\}$. If equality holds output valid. Otherwise, output invalid.

Security Proof We prove the security of our signature scheme under the standard *existential unforgeability with respect to adaptive chosen message attacks* (UF-CMA) model, which was formalised by Goldwasser *et al.* [99].

Theorem 21. *Suppose the (ϵ', t')-DHI assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then our signature scheme above is (ϵ, t, q_S)-unforgeable against chosen message attacks provided that*

$$q_S < \frac{n+1}{2}, \quad \epsilon \geq \epsilon'(1 - \frac{1}{p}) \quad \textit{and} \quad t \leq t' - O((q_S + n)T),$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$.

Proof. Suppose there is an adversary $\mathcal{A}$ that (ϵ, t, q_S)-breaks the signature scheme. We construct an algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the DHI problem in time t' with advantage ϵ. $\mathcal{B}$ is given a random instance (g_1, g_2, g_1^x, g_2^x) of the DHI problem, for some unknown $x \in \mathbb{Z}_p$. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ sets the public key as follows:

1. Choose a random key K for the PRF function G_K.

2. Select some random $\mu_0, \ldots, \mu_n, \nu_0, \ldots, \nu_n \in \mathbb{Z}_p^*$. For $i = 0, \ldots, n$, set $u_i = g_1^{\mu_i} g_1^{x\nu_i}$. For any message m, the PRF $G_K(m)$ outputs $\{m_1, \ldots, m_n\}$. Denote $F(m) = \mu_0 + \sum_{i=1}^n \mu_i m_i$ and $J(m) = \nu_0 + \sum_{i=1}^n \nu_i m_i$. Therefore, we have:

$$u_0 \prod_{i=1}^n u_i^{m_i} = g_1^{\mu_0 + \sum_{i=1}^n \mu_i m_i} (g_1^x)^{\nu_0 + \sum_{i=1}^n \nu_i m_i} = g_1^{F(m)} (g_1^x)^{J(m)}.$$

3. Set $v = g_2^x$ and calculate $z = \hat{e}(g_1, g_2)$.
4. The public key is $(g_1, g_2, u_0, u_1, \ldots, u_n, v, z, K)$. The corresponding secret key is x.

Signing Oracle: The adversary $\mathcal{A}$ issues a query for a message $\mathtt{M}$. $\mathcal{B}$ sets $r = -F(\mathtt{M})$ and $\sigma = g_1^{J(\mathtt{M})}$. Observe that (σ, r) is a valid signature:

$$\sigma = (g_1^{-F(\mathtt{M})}(g_1^{F(\mathtt{M})}(g_1^x)^{J(\mathtt{M})}))^{1/x} = g_1^{J(\mathtt{M})}.$$

$\mathcal{B}$ returns (σ, r) as the output.

Extract from Forgery: Eventually, $\mathcal{A}$ will output a forgery $(\mathtt{M}^*, (\sigma^*, r^*))$. $\mathcal{B}$ proceeds as follows:

1. If $r^* + F(\mathtt{M}^*) = 0$, declare failure and exit.
2. Otherwise, output $W = (\sigma^* g_1^{-J(\mathtt{M})})^{\frac{1}{r^* + F(\mathtt{M}^*)}}$ as the solution to the DHI problem. Observe that:
$$\sigma^* = (g_1^{r^*} g_1^{F(\mathtt{M}^*)}(g_1^x)^{J(\mathtt{M}^*)})^{1/x} = g_1^{\frac{r^* + F(\mathtt{M}^*)}{x}} g_1^{J(\mathtt{M})}.$$

Analysis: We now argue that any adversary $\mathcal{A}$ against the signature scheme will have success in the game presented by $\mathcal{B}$. We define a sequence of games and show that if $\mathcal{A}$ is successful against Game j, then it will also be successful against Game $j+1$. The first game models the real security game and the last game is exactly the view of $\mathcal{A}$ when interacting with $\mathcal{B}$.

Game 1: The game is defined to be the same as the security game.

Game 2: The same as Game 1, with the exception that at the beginning of the game $\mathcal{B}$ sets $u_i = g_1^{\mu_i} g_1^{x\nu_i}$.

Game 3: The same as Game 2, with the exception that $\mathcal{A}$ fails if $r^* + F(\mathtt{M}^*) = 0$.

Game 4: The same as Game 3, with the exception that $\mathcal{B}$ answers the signing oracle as the above simulation.

Define $\mathbf{Adv}_{\mathcal{A}}[\text{Game } x]$ as the advantage of $\mathcal{A}$ in Game x. We have the following claims that will lead to the main theorem.

Claim 4.

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 1}].$$

Proof. These games are identical. The only difference is in the presentation of how u_i are chosen. In Game 1, they are randomly chosen from $\mathbb{G}_1$. In Game 2, $u_i = g_1^{\mu_i} g_1^{x\nu_i}$ where μ_i and ν_i are randomly chosen from $\mathbb{Z}_p$. In both games, u_i are uniformly distributed in $\mathbb{G}_1$.

□

Claim 5. *If G is a secure pseudorandom function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] - 1/p.$$

Proof. By the pseudorandomness property of G_K, the output of $G_K(\mathtt{M}^*)$ is a random n-nit string (Otherwise, it would admit a distinguishing attack against the PRF). Therefore, $\Pr[r^* + F(\mathtt{M}^*) = 0] = 1/p$. □

Claim 6. *If G is a secure pseudorandom function and $q_S < \frac{n+1}{2}$, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 4}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 3}].$$

Proof. We first analysis on the distribution of the signing oracle output by $\mathcal{B}$. $\mathcal{B}$ outputs (σ_i, r_i) such that $\sigma_i = g_1^{J(\mathtt{M}_i)}$ and $r_i = -F(\mathtt{M}_i)$. Observe that $J(\mathtt{M}_i)$ is a polynomial with $n+1$ random variables $\nu_0, \ldots, \nu_n$ unknown to $\mathcal{A}$; and $F(\mathtt{M}_i)$ is a polynomial with $n+1$ random variables $\mu_0, \ldots, \mu_n$ unknown to $\mathcal{A}$. We represent the knowledge gained by the adversary as a matrix product:

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{q_S} \end{bmatrix} = \begin{bmatrix} 1 & m_{1,1} & \ldots & m_{1,n} \\ 1 & m_{2,1} & \ldots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & m_{q_S,1} & \ldots & m_{q_S,n} \end{bmatrix} \cdot \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \begin{bmatrix} \log_{g_1} \sigma_1 \\ \log_{g_1} \sigma_2 \\ \vdots \\ \log_{g_1} \sigma_{q_S} \end{bmatrix} = \begin{bmatrix} 1 & m_{1,1} & \ldots & m_{1,n} \\ 1 & m_{2,1} & \ldots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & m_{q_S,1} & \ldots & m_{q_S,n} \end{bmatrix} \cdot \begin{bmatrix} \nu_0 \\ \nu_1 \\ \vdots \\ \nu_n \end{bmatrix}.$$

On input a message $\mathtt{M}_i$, the output of $G_K(\mathtt{M}_i) = \{m_{i,1}, \ldots m_{i,n}\}$ is a random n-nit string by the pseudorandomness property of the PRF (Otherwise, it would admit a distinguishing attack against the PRF). If $q_S < n+1$, the number of columns is more than the number of rows in the above matrix. Therefore the distribution of the signing oracle output is indistinguishable from the real output from the Sign protocol.

However, u_i also contains information about μ_i and μ_i. Therefore, we also have to analysis on the distribution of u_i and the signing oracle output. We can represent the knowledge gained by the adversary as a matrix product, similar to the proof of claim 2. Therefore the adversary has $2q_S + n + 1$ equations with $2n + 2$ random variables. Since we restrict that $q_S < (n+1)/2$, the number of columns is more than the number of rows in the above matrix.

Table 5.4: Comparison of efficiency and the loss due to the number of signing oracle query q. mpk stands for the master public key. E_i represents the time of an exponentiation in the group $\mathbb{G}_i$. M_i represents the time of a point multiplication in the group $\mathbb{G}_i$. P represents the time of a pairing computation. The signature size is larger when we take into account the Cheon's attack on the q-SDH problem.

	Signature Size	Signing Time	Verification Time
Waters [206]	1760 bits	1 E_1, 1 E_2, $(n'+1)$ M_2	2 P + $(n'+1)$ M_T
Boneh-Boyen [29]	320 bits*	1 E_1	1 P + 2 E_2 + 2 M_2
This section	320 bits	2 E_1, $(n+1)$ M_1	1 P + 1 E_T + $(n+1)$ M_T

	mpk Size	EU-CMA Assumption	Reduction Loss
Waters [206]	$O(n')$	CDH	$\frac{\epsilon}{16q(n'+1)}$
Boneh-Boyen [29]	O(1)	q-SDH	$\frac{\epsilon}{2} - \frac{q}{p}$
This section	$O(q)$	DHI	$\epsilon(1 - \frac{1}{p})$

The output of $G_K(\mathtt{M}_i)$ is a random n-nit string by the pseudorandomness property of the PRF (Otherwise, it would admit a distinguishing attack against the PRF). Therefore, the distribution of u_i and the signing oracle outputs are indistinguishable from the real output from the Sign protocol. □

As a result, $\mathcal{B}$ can solve the DHI problem with probability $1 - \frac{1}{p}$. □

Comparison We compare our signature scheme with two other signatures from pairings: the Waters signatures [206] and the Boneh-Boyen signatures [29]. The Waters signatures lead to the Waters-IBE [206] and the Boneh-Boyen signatures lead to the Gentry-IBE [92].

We summarise the results in Table 5.4. The Waters signatures consist of 1 element in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. We use a MNT curve to implement an asymmetric pairing at security level 80, with embedding degree 10. The Waters signatures consist of 1 element in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. The signature size is 1760 bit in this case. For our signatures and the Boneh-Boyen signatures, they both consist of a $\mathbb{G}_1$ element and an integer in $\mathbb{Z}_p$. The signature size is 320 bits.

We compare the pros and cons of the three signature schemes:

- The Waters signatures:
  - Pros: The security is reduced to the standard CDH problem.
  - Cons: The public parameters size is proportional to n', where n' is inversely

proportional to the probability of solving the CDH problem. The signature size is about 3 times larger. It has a loose security reduction.

- The Boneh-Boyen signatures:
  - Pros: It has a short signature and a short public parameters . The security reduction is tight.
  - Cons: The security relies on the q-SDH problem, where q is the number of signing oracle query. Cheon [64] proved that the security of the q-SDH problem on an abelian group of order p can be reduced up to $O(\log p \cdot p^{1/3})$ (resp. $O(\log p \cdot p^{1/3})$) for large q if $p-1$ (resp. $p+1$) has a divisor $d = O(p^{1/2})$ (resp. $d = O(p^{1/3})$). Because of this, Cheon suggested to use 220-bit prime p for 80-bit symmetric security level. When we take into account the Cheon's attack, the signature size will be larger.
- Our signatures:
  - Pros: The security is reduced to the DHI problem, which is equivalent to the CDH problem. It has a short signature. The security reduction is tight.
  - Cons: The public parameters size is proportional to n, where $q \leq (n+1)/2$ and q is the number of signing oracle query. The number of multiplication in signing and verification are also proportional to q.

For our signature scheme, the public parameters size n is larger than q, which is about 2^{30} in general case. In the Waters signatures, Waters suggested that identities are bitstrings of arbitrary length and n' be the output length of a collision-resistant hash function, which is about 160 bits.

Notice that both the Waters signatures, the Boneh-Boyen signatures and our signatures have some loss related to the number of signing oracle query q. In the Waters signatures, the reduction loss in the security proof is proportional to q. The intractability assumption of the Boneh-Boyen signatures is the q-SDH assumption, which has a problem instance of size q. In our signatures, the public parameters size is proportional to q. We summarise the loss related to q in Table 5.4.

5.4.3 Our Second Construction

Our first construction has a tight security reduction to static intractability assumptions (which is not related to q). However, the large number of public parameter makes

the first construction not practical to use. We propose another construction which is practical to use, with the price of having a security reduction loss of q. We use the recent dual system encryption technique to construct our second proposal.

Dual System Encryption The dual system encryption technique introduced by Waters [208] to be used to construct fully secure IBE under simple assumptions. Their IBE has ciphertexts, private keys and public parameters each consisting of a constant number of group elements. In a dual system, identity-based secret keys and ciphertexts can be in one of two indistinguishable forms. A key or ciphertext is *normal* if it is generated from the real system. *Semi-functional* keys and ciphertexts are only involved in the security proofs. A normal key can decrypt normal or semi-functional ciphertexts; and a normal ciphertext can be decrypted by normal or semi-functional keys. When a semi-functional ciphertext is decrypted by a semi-functional key, the decryption will fail (with an overwhelming probability). Security of dual systems is proved using a sequence of indistinguishable games.

There are two IBE schemes [208, 139] which use the dual system encryption technique. However, both of them are *not* anonymous. It is advantageous to construct a fully secure anonymous IBE using the dual system encryption technique. It can directly achieve the adaptive identity security without any security degradation, without using long public parameters, and without depending on non-static complexity assumptions.

Composite Order Bilinear Groups and Intractability Assumptions The dual system encryption uses the composite order bilinear groups. We first introduce the important properties of composite order bilinear groups and the related intractability assumptions used in this section.

Composite order bilinear groups were firstly introduced by Boneh *et al.* [37]. They defined a composite order bilinear group with order $N = p_1p_2$, where p_1, p_2 are distinct primes. Lewko and Waters [139] further defined a composite order bilinear group whose order is a product of three distinct primes. We first review their definition of composite order bilinear groups.

We define a group generator $\mathcal{G}$, which takes a security parameter 1^λ as input, outputs a description of bilinear group $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where p_1, p_2, p_3 are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map such that:

- (Bilinear) For all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$, we have $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.

- (Non-degenerate) If g is a generator of $\mathbb{G}$, then $\hat{e}(g,g)$ generates $\mathbb{G}_T$.

We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the bilinear map $\hat{e}$ are computable in polynomial time with respect to λ. We also denote $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ as the subgroups of order p_1, p_2 and p_3 in $\mathbb{G}$ respectively. Let g_1, g_2, g_3 be the generators of the subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. Note that for all $h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$, if $i \neq j$,

$$\hat{e}(h_i, h_j) = 1.$$

We now give our complexity assumptions which will be used in our security proofs. We denote $\mathbb{G}_{p_1 p_2}$ as the subgroup of order $p_1 p_2$ in $\mathbb{G}$. For all $T \in \mathbb{G}_{p_1 p_2}$, T can be written uniquely as the product of an element of $\mathbb{G}_{p_1}$ and an element of $\mathbb{G}_{p_2}$. We refer to these elements as the "$\mathbb{G}_{p_1}$ part of T" and the "$\mathbb{G}_{p_2}$ part of T" respectively. We also define $\mathbb{G}_{p_1 p_3}$ and $\mathbb{G} = \mathbb{G}_{p_1 p_2 p_3}$similarly.

We now give our complexity assumptions which will be used in this paper. Our Assumption 2 is the same as that in [139]. Our Assumption 1 and Assumption 3 is slightly different from that in [139]. In our Assumption 1, the adversary is given an extra element in $\mathbb{G}_{p_1 p_2}$. The reason will be explained in the security proof. In the Assumption 3 of [139], the algorithm $\mathcal{A}_3$ has to decide if $T = \hat{e}(g,g)^{\alpha s}$. In our Assumption 3, the algorithm $\mathcal{A}_3$ has to decide if $T = g^{\alpha s}$. Therefore, we can view our Assumption 3 as the decisional Diffie-Hellman problem in the subgroup setting; and the Assumption 3 in [139] as the decisional bilinear Diffie-Hellman problem in the subgroup setting. Our Assumption 4 is the same as the Decisional Bilinear assumption in [65], except that it is defined in the subgroup setting.

Assumption 1 (Subgroup decision problem for $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_1 p_2}$). Given a group generator $\mathcal{G}$, we define the following experiment:

Experiment $\mathbf{Exp}^{\text{Assumption1}}_{\mathcal{G},\mathcal{A}_1,\beta}(1^\lambda)$

$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}(1^\lambda), \quad g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, \quad X_2 \xleftarrow{R} \mathbb{G}_{p_2}, \quad X_3 \xleftarrow{R} \mathbb{G}_{p_3},$

$T_0 \xleftarrow{R} \mathbb{G}_{p_1 p_2}, \quad T_1 \xleftarrow{R} \mathbb{G}_{p_1}.$

Return $\beta' \leftarrow \mathcal{A}_1(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, X_1 X_2, X_3, T_\beta)$.

We define the advantage of an algorithm $\mathcal{A}_1$ in breaking Assumption 1 to be:

$$Adv_{\mathcal{G},\mathcal{A}_1}(\lambda) := \left| \Pr[\mathbf{Exp}^{\text{Assumption1}}_{\mathcal{G},\mathcal{A}_1,1}(1^\lambda) = 1] - \mathbf{Exp}^{\text{Assumption1}}_{\mathcal{G},\mathcal{A}_1,0}(1^\lambda) = 1] \right|.$$

Definition 33. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 1 if $Adv_{\mathcal{G},\mathcal{A}_1}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_1$.*

Assumption 2 (Subgroup decision problem for $\mathbb{G}_{p_1p_3}$ and $\mathbb{G}_{p_1p_2p_3}$). Given a group generator $\mathcal{G}$, we define the following experiment:

$$\begin{aligned}&\text{Experiment } \mathbf{Exp}^{\text{Assumption2}}_{\mathcal{G},\mathcal{A}_2,\beta}(1^\lambda)\\&\quad(N=p_1p_2p_3,\mathbb{G},\mathbb{G}_T,\hat{e})\xleftarrow{R}\mathcal{G}(1^\lambda),\quad g,X_1\xleftarrow{R}\mathbb{G}_{p_1},\\&\quad X_2,Y_2\xleftarrow{R}\mathbb{G}_{p_2},\quad X_3,Y_3\xleftarrow{R}\mathbb{G}_{p_3},\quad T_0\xleftarrow{R}\mathbb{G},\quad T_1\xleftarrow{R}\mathbb{G}_{p_1p_3}.\\&\quad\text{Return } \beta'\leftarrow\mathcal{A}_2(N,\mathbb{G},\mathbb{G}_T,\hat{e},g,X_1X_2,X_3,Y_2Y_3,T_\beta).\end{aligned}$$

We define the advantage of an algorithm $\mathcal{A}_2$ in breaking Assumption 2 to be:

$$Adv_{\mathcal{G},\mathcal{A}_2}(\lambda):=\left|\Pr[\mathbf{Exp}^{\text{Assumption2}}_{\mathcal{G},\mathcal{A}_2,1}(1^\lambda)=1]-\mathbf{Exp}^{\text{Assumption2}}_{\mathcal{G},\mathcal{A}_2,0}(1^\lambda)=1]\right|.$$

Definition 34. *We say that $\mathcal{G}$ satisfies (ϵ,t)-Assumption 2 if $Adv_{\mathcal{G},\mathcal{A}_2}(\lambda)=\epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_2$.*

Assumption 3 (Subgroup DDH problem). Given a group generator $\mathcal{G}$, we define the following experiment:

$$\begin{aligned}&\text{Experiment } \mathbf{Exp}^{\text{Assumption3}}_{\mathcal{G},\mathcal{A}_3,\beta}(1^\lambda)\\&\quad(N=p_1p_2p_3,\mathbb{G},\mathbb{G}_T,\hat{e})\xleftarrow{R}\mathcal{G}(1^\lambda),\quad \alpha,s\xleftarrow{R}\mathbb{Z}_N,\quad g\xleftarrow{R}\mathbb{G}_{p_1},\\&\quad X_2,Y_2,Z_2\xleftarrow{R}\mathbb{G}_{p_2},\quad X_3\xleftarrow{R}\mathbb{G}_{p_3},\quad T_0=g^{\alpha s},\quad T_1\xleftarrow{R}\mathbb{G}_T.\\&\quad\text{Return } \beta'\leftarrow\mathcal{A}_3(N,\mathbb{G},\mathbb{G}_T,\hat{e},g,g^\alpha X_2,g^sY_2,Z_2,X_3,T_\beta).\end{aligned}$$

We define the advantage of an algorithm $\mathcal{A}_3$ in breaking Assumption 3 to be:

$$Adv_{\mathcal{G},\mathcal{A}_3}(\lambda):=\left|\Pr[\mathbf{Exp}^{\text{Assumption3}}_{\mathcal{G},\mathcal{A}_3,1}(1^\lambda)=1]-\mathbf{Exp}^{\text{Assumption3}}_{\mathcal{G},\mathcal{A}_3,0}(1^\lambda)=1]\right|.$$

Definition 35. *We say that $\mathcal{G}$ satisfies (ϵ,t)-Assumption 3 if $Adv_{\mathcal{G},\mathcal{A}_3}(\lambda)=\epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_3$.*

Assumption 4 (Subgroup decisional bilinear problem). Given a group generator $\mathcal{G}$, we define the following experiment:

$$\begin{aligned}&\text{Experiment } \mathbf{Exp}^{\text{Assumption4}}_{\mathcal{G},\mathcal{A}_4,\beta}(1^\lambda)\\&\quad((N,\mathbb{G},\mathbb{G}_T,\hat{e})\leftarrow\mathcal{G}(1^\lambda),g,v,w\in\mathbb{G}^3)\leftarrow\mathcal{A}_4(1^\lambda),\\&\quad x,s\xleftarrow{R}\mathbb{Z}_N,\quad T_0=\hat{e}(g,g)^{sx},\quad T_1\xleftarrow{R}\mathbb{G}_T.\\&\quad\text{Return } \beta'\leftarrow\mathcal{A}_4(g^s,v^s,w^s,\hat{e}(g,g)^x,T_\beta).\end{aligned}$$

We define the advantage of an algorithm $\mathcal{A}_4$ in breaking Assumption 4 to be:

$$Adv_{\mathcal{G},\mathcal{A}_4}(\lambda):=\left|\Pr[\mathbf{Exp}^{\text{Assumption4}}_{\mathcal{G},\mathcal{A}_4,1}(1^\lambda)=1]-\mathbf{Exp}^{\text{Assumption4}}_{\mathcal{G},\mathcal{A}_4,0}(1^\lambda)=1]\right|.$$

Definition 36. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 4 if $Adv_{\mathcal{G},\mathcal{A}_4}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_4$.*

Intuition behind the Construction To construct an anonymous IBE scheme using the Boneh-Franklin type session key, we tried to use the dual system encryption technique. Suppose $u, h, g_1, g_1^\alpha \in \mathbb{G}_{p_1}$ are the master public keys and α is the master secret key. We do not use the full domain hash as in BF-IBE [34], since it is hard to prove the security without random oracles. We set $H(\mathsf{ID}) = u^{\mathsf{ID}}h$. It provides collision resistant if $\log_u h \mod N$ is not known. The identity-based secret key is

$$d = (u^{\mathsf{ID}}h)^\alpha R_3,$$

where R_3 is randomly chosen from a group $\mathbb{G}_{p_3}$. To encrypt a message m, the sender picks a random $s \in \mathbb{Z}_N$ and calculate the ciphertext

$$C = \big(\quad c_1 = g_1^s, \quad c_2 = m \cdot \hat{e}(g_1^\alpha, u^{\mathsf{ID}}h)^s \quad \big).$$

However, this simple construction does not work, since we cannot construct nominally semi-functional keys and ciphertexts that will cancel out each other during the simulation of the proof as in [139]. Interestingly, when we try to convert this IBE into a two-level hierarchical IBE (HIBE) scheme, we are able to construct nominally semi-functional keys and ciphertexts. We uses the BF-IBE key structure at both levels:

1st level secret key for $ID_1 = H_1(ID_1)^\alpha$,

2nd level secret key for $(ID_1, ID_2) = (H_1(ID_1)^\alpha H_2(ID_2)^r, g_1^r)$,

where $ID_1, ID_2, r \in_R \mathbb{Z}_N$. Note that we use the commutative blinding to join the second level secret keys together. The two-level HIBE provides anonymity to the first level identity only. The identity-based secret key for the identity $(\mathsf{ID}_1, \mathsf{ID}_2)$ is

$$D_1 = (u^{\mathsf{ID}_1}h)^\alpha (v^{\mathsf{ID}_2}w)^r R_3, \quad D_2 = g^r R_3',$$

where $v, w \in \mathbb{G}_{p_1}$ are also in the master public keys, R_3, R_3' are randomly chosen from a group $\mathbb{G}_{p_3}$ and r is randomly chosen from $\mathbb{Z}_N$. To encrypt a message m, the sender picks a random $s \in \mathbb{Z}_N$ and calculate the ciphertext

$$\hat{C} = \big(\quad C_1 = g_1^s, \quad C_2 = (v^{\mathsf{ID}_2}w)^s, \quad C_3 = m \cdot \hat{e}(g_1^\alpha, u^{\mathsf{ID}_1}h)^s \quad \big).$$

The extra elements C_2 and D_2 enable the formulation of semi-functional keys and ciphertexts. However, the new ID_2 is no longer anonymous since everyone can use pairings to check if $\hat{e}(C_1, v^{\mathsf{ID}_2}w) = \hat{e}(g_1, C_2)$. Therefore, we set the second level identity ID_2 to be the verification key of one-time signature. It has two usages:

1. Provide CCA security using the transformation in [54],
2. Avoid the lost of anonymity since the verification key is not related to the real identity ID_1.

Since the second level identity is now chosen by the encryptor, the user ID_1 must be able to "delegate" his secret key to the new "identity" (ID_1, ID_2). Therefore, the secret key for ID_1 becomes

$$D_1 = (u^{ID_1}h)^{\alpha}w^r, \quad D_2 = g_1^r, \quad D_3 = v^r.$$

Then the user ID_1 can compute the delegated secret key as

$$D_1' = D_1 D_3^{ID_2}, \quad D_2' = D_2.$$

This key can be used to decrypt ciphertext for the "identity" (ID_1, ID_2).

CCA-secure Construction Let $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme in which the verification key outputted by $\mathsf{Gen}(1^\lambda)$ is in $\mathbb{Z}_N$.

- **Setup**: The setup algorithm runs bilinear group generator $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. Suppose the group generator $\mathcal{G}$ also gives the generator g_1 and g_3 of the subgroups $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$ respectively. The PKG picks random a random $\alpha \leftarrow \mathbb{Z}_N$ and random generators $u, h, v, w \in \mathbb{G}_{p_1}$. (Random elements of $\mathbb{G}_{p_1}$ can be obtained by taking a generator of $\mathbb{G}_{p_1}$ and raising it to random exponents modulo N.) The master public key is

  $$(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, u, h, v, w, \hat{e}(g_1^\alpha, u), \hat{e}(g_1^\alpha, h)).$$

  The master secret key is (α, g_3).

- **Extract**: Given the master secret key (α, g_3) and an identity ID, pick a random $r \in \mathbb{Z}_N$ and some random $R_3, R_3', R_3'' \in \mathbb{G}_{p_3}$. Then calculate the identity-based secret key

  $$sk_{\mathsf{ID}} = (D_1 = (u^{\mathsf{ID}}h)^{\alpha}w^r R_3, \quad D_2 = g_1^r R_3', \quad D_3 = v^r R_3'').$$

- **Encrypt**: To encrypt a message $m \in \mathbb{G}_T$ for a user ID, the sender picks a random $s \in \mathbb{Z}_N$. The sender runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and calculates:

  $$C_1 = g_1^s, \quad C_2 = (v^{\mathsf{vk}}w)^s, \quad C_3 = m \cdot (\hat{e}(g_1^\alpha, u)^{\mathsf{ID}} \cdot \hat{e}(g_1^\alpha, h))^s.$$

  Denote $\hat{C} = (C_1, C_2, C_3)$. The sender signs $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\hat{C})$ and sends the ciphertext $(\hat{C}, \sigma, \mathsf{vk})$.

- **Decrypt**: Given a ciphertext $(\hat{C} = (C_1, C_2, C_3), \sigma, \mathsf{vk})$, and a secret key $sk_{\mathsf{ID}} = (D_1, D_2, D_3)$ for an identity ID, the recipient outputs $\perp$ if $\mathsf{Vrfy}_{\mathsf{vk}}(\hat{C}, \sigma) \neq 1$. Otherwise, the recipient calculates:

$$m = \frac{C_3 \cdot \hat{e}(D_2, C_2)}{\hat{e}(C_1, D_1 D_3^{\mathsf{vk}})}.$$

Security To prove the security of our IBE scheme, we first define two additional structures: semi-functional keys and semi-functional ciphertexts. They are used in the security proofs only.

Semi-functional keys: We let g_2 be a generator of the subgroup $\mathbb{G}_{p_2}$. A semi-functional key can be created from a normal key (D_1, D_2, D_3). Firstly, random exponents $\gamma, z_k, z_k' \in \mathbb{Z}_N$ are chosen. Then the semi-functional key is set to be $(D_1' = D_1 g_2^{\gamma z_k}, D_2' = D_2 g_2^{\gamma}, D_3' = D_3 g_2^{\gamma z_k'})$.

Semi-functional ciphertexts: A semi-functional ciphertext can be created by running $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and calculating $\hat{C} = (C_1, C_2, C_3)$ as in **Encrypt**. After that, random exponents $\delta, z_c \in \mathbb{Z}_N$ are chosen. Set $C' = (C_1' = C_1 g_2^{\delta}, C_2' = C_2 g_2^{\delta z_c}, C_3' = C_3)$ and sign $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(C')$. Then the semi-functional ciphertext is set to be $(C', \mathsf{vk}, \sigma)$.

Notice that decryption will succeed if a semi-functional key is used to decrypt a normal ciphertext, or a normal key is used to decrypt a semi-functional ciphertext. However, if a semi-functional key is used to decrypt a semi-functional ciphertext, there will be an extra blinding factor $\hat{e}(g_2, g_2)^{\gamma\delta(z_k + \mathsf{vk} \cdot z_k' - z_c)}$ to hinder the decryption. If $z_c = z_k + \mathsf{vk} \cdot z_k'$, decryption still works. We call this key and ciphertext as nominally semi-functional.

Theorem 22. *Our IBE scheme (ϵ, t, q_k, q_d)-IND-ID-CCA secure if (ϵ', t')-Assumption 1, (ϵ', t')-Assumption 2 and (ϵ', t')-Assumption 3 hold, where*

$$\epsilon' \geq \frac{\epsilon}{q_k + 3}, \quad t' = t + O(q_k \rho + q_d(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}$ and a pairing operation respectively.

Proof. We prove the IND-ID-CCA security by a hybrid argument using a sequence of games. The first game Game_{real} is the real IND-ID-CCA game and we denote the challenge ciphertext as $(C^*, \mathsf{vk}^*, \sigma^*)$ and the challenge identity as ID^*. The second game $\text{Game}_{restricted}$ is the same as Game_{real} except that the adversary cannot ask for the secret key of identity ID such that $\mathsf{ID} \equiv \mathsf{ID}^* \mod p_1$, and cannot ask for decryption

of ciphertext $(\cdot, \mathsf{vk}, \cdot)$ for identity ID such that $\mathsf{ID} \equiv \mathsf{ID}^* \mod p_1$ and $\mathsf{vk} \equiv \mathsf{vk}^* \mod p_1$. This strong restriction will be retained throughout the subsequent games [10]. After that, we denote q_e as the number of extraction oracle queries. For $k = 0$ to q_e, we define Game_k as:

Game$_k$: It is the same as $\text{Game}_{restricted}$, except that the challenge ciphertext is semi-functional and the first k keys outputted from the extraction oracle are semi-functional. The rest of the keys are normal.

In other words, all keys are normal and the challenge ciphertext is semi-functional in Game_0. In Game_{q_k}, all keys and the challenge ciphertext are semi-functional. The last game is Game_{final}, which is the same as Game_{q_e} except that:

- the decryption oracle uses the semi-functional key to decrypt;
- the challenge ciphertext is a semi-functional encryption of a random message, instead of one of the two challenge messages.

We will prove the indistinguishability between all these games.

Lemma 5. *If there exists an adversary $\mathcal{A}$ such that*

$$Adv_{\mathcal{A}}(\text{Game}_{real}) - Adv_{\mathcal{A}}(\text{Game}_{restricted}) = \epsilon,$$

then we can construct an algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 1 or Assumption 2.

Proof. Given g and X_3 from either Assumption 1 or Assumption 2, $\mathcal{B}$ can simulate Game_{real} or $\text{Game}_{restricted}$ with $\mathcal{A}$. Denote the challenge ciphertext as $(C^*, \mathsf{vk}^*, \sigma^*)$ and the challenge identity as ID^*. $\mathcal{A}$ can make the following oracle queries:

- Extraction Oracle. If $\mathcal{A}$ made a query ID to the extraction oracle, and
  - if $\mathsf{ID} = \mathsf{ID}^*$ and $p_1 \nmid (\mathsf{ID} - \mathsf{ID}^*)$, then $\mathcal{B}$ generates the secret key of ID^* by using the master secret key (α, X_3);
  - (For Game_{real} only.) if $\mathsf{ID} \neq \mathsf{ID}^*$ and $p_1 \mid (\mathsf{ID} - \mathsf{ID}^*)$, then $\mathcal{B}$ can calculate a non-trivial factor of N by computing $a = \gcd(\mathsf{ID} - \mathsf{ID}^*, N)$. We denote $b = N/a$. Consider the following three cases:
    1. $a = p_1$ and $b = p_2 p_3$,

[10] In [139], the IND-ID-CPA proof only restrict the extraction query of $\mathsf{ID} \equiv \mathsf{ID}^*$ modulus p_2 but not p_1. We believe that the restriction in p_1 is also necessary for the scheme in [139], since if $\mathsf{ID} \equiv \mathsf{ID}^* \mod p_1$, they will have the same key.

2. $a = p_1 p_2$ and $b = p_3$,
3. $a = p_1 p_3$ and $b = p_2$.

For case 1 and case 3, $\mathcal{B}$ will break Assumption 1. Given g, X_1X_2, X_3, T, $\mathcal{B}$ firstly checks if $a \neq p_1p_2$ by testing whether $(X_1X_2)^a$ is not the identity[11]. If so, $\mathcal{B}$ test whether T^a is the identity. If it is, then $T \in \mathbb{G}_{p_1}$. Otherwise, $T \in \mathbb{G}_{p_1p_2}$.

In case 2, $\mathcal{B}$ will break Assumption 2. Given $g, X_1X_2, X_3, Y_2Y_3, T$, $\mathcal{B}$ firstly checks if $a = p_1p_2$ by testing whether $(X_1X_2)^a$ is the identity. If so, $\mathcal{B}$ test whether $\hat{(}Y_2Y_3, T)^b$ is the identity. If it is, then $T \in \mathbb{G}_{p_1p_3}$. Otherwise, $T \in \mathbb{G}$.

- Decryption Oracle. If $\mathcal{A}$ made a query $((\hat{C}, \mathsf{vk}, \sigma), \mathsf{ID})$ to the decryption oracle, and
  - if $\mathsf{ID} \not\equiv \mathsf{ID}^* \mod p_1$, then $\mathcal{B}$ firstly generates the secret key of ID^* by using the master secret key (α, X_3) and then decrypts;
  - if $\mathsf{ID} \equiv \mathsf{ID}^* \mod p_1$ and $\mathsf{vk} \not\equiv \mathsf{vk}^* \mod p_1$, then $\mathcal{B}$ firstly generates the secret key of ID^* by using the master secret key (α, X_3) and then decrypts;
  - (For Game_{real} only.) if $\mathsf{ID} \equiv \mathsf{ID}^* \mod p_1$, $\mathsf{vk} \neq \mathsf{vk}^*$ and $p_1 \mid (\mathsf{vk} - \mathsf{vk}^*)$, then $\mathcal{B}$ can calculate a nontrivial factor of N by computing $a = \gcd(\mathsf{vk} - \mathsf{vk}^*, N)$. Similar to above, $\mathcal{B}$ will break either Assumption 1 or Assumption 2.
  - (For Game_{real} only.) if $\mathsf{ID} = \mathsf{ID}^*$ and $\mathsf{vk} = \mathsf{vk}^*$, it implies $(C^*, \mathsf{vk}^*, \sigma^*) \neq (C, \mathsf{vk}, \sigma)$. Then σ is a forgery of the strong one time signature with message C, signature σ and verification key vk^*. It contradicts that (Gen, Sign, Vrfy) is a strong one time signature scheme;
  - (For Game_{real} only.) if $\mathsf{ID} \neq \mathsf{ID}^*$, $p_1 \mid (\mathsf{ID} - \mathsf{ID}^*)$ and $\mathsf{vk} = \mathsf{vk}^*$, then $\mathcal{B}$ can calculate a non-trivial factor of N by computing $a = \gcd(\mathsf{ID} - \mathsf{ID}^*, N)$. Similar to above, $\mathcal{B}$ will break either Assumption 1 or Assumption 2.

To sum up, $\mathcal{B}$ will break either Assumption 1 or Assumption 2 with non-negligible advantage. □

Lemma 6. *If there exists an adversary $\mathcal{A}$ such that*

$$Adv_{\mathcal{A}}(\text{Game}_{restricted}) - Adv_{\mathcal{A}}(\text{Game}_0) = \epsilon,$$

[11] We need to use the term X_1X_2 to ensure that $a \neq p_1p_2$ (which implies that it is either case 1 or case 3). Only in case 1 and case 3, we can use a to distinguish if $T \in \mathbb{G}_{p_1}$ or $T \in \mathbb{G}_{p_1p_2}$. Therefore our Assumption 1 is slightly different from the Assumption 1 in [139].

then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.

Proof. Given g, X_3, T, $\mathcal{B}$ can simulate $\text{Game}_{restricted}$ or Game_0 with $\mathcal{A}$. $\mathcal{B}$ chooses random $a, b, c, d, \alpha \in \mathbb{Z}_N$ and sets

$$g_1 = g, \quad v = g^a, \quad w = g^b, \quad u = g^c, \quad h = g^d.$$

$\mathcal{B}$ sends the master public key $(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, u, h, v, w, \hat{e}(g_1, u)^\alpha, \hat{e}(g_1, h)^\alpha)$ to $\mathcal{A}$.

For the i-th extraction oracle query with input ID_i, $\mathcal{B}$ chooses random $r_i, \rho_i, \mu_i, \theta_i \in \mathbb{Z}_N$ and sets

$$D_1 = (u^{\mathsf{ID}_i} h)^\alpha w^{r_i} X_3^{\rho_i}, \quad D_2 = g_1^{r_i} X_3^{\mu_i}, \quad D_3 = v^{r_i} X_3^{\theta_i}.$$

For each decryption oracle query with input (C_i, ID_i), $\mathcal{B}$ extracts the secret key for ID_i as above and then decrypts following the **Decrypt** algorithm.

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0, 1\}$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{B}$ calculates:

$$C_1^* = T, \quad C_2^* = T^{a \cdot \mathsf{vk}^* + b}, \quad C_3^* = M_{b'}^* \cdot \hat{e}(T^\alpha, u^{\mathsf{ID}^*} h),$$

and $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(C_1^*, C_2^*, C_3^*)$. The challenge ciphertext is $((C_1^*, C_2^*, C_3^*), \sigma^*, \mathsf{vk}^*)$.

If $T \in \mathbb{G}_{p_1}$, this is a normal ciphertext and hence $\mathcal{B}$ simulates $\text{Game}_{restricted}$. If $T \in \mathbb{G}_{p_1 p_2}$, this is a semi-functional ciphertext with $z_c = a \cdot \mathsf{vk}^* + b$ and hence $\mathcal{B}$ simulates Game_0. Note that the value of $z_c \mod p_2$ is not correlated with the values of a and b modulo p_1 by the Chinese Remainder Theorem.

On the other hand, we restrict $\mathcal{A}$ in $\text{Game}_{restricted}$ for not asking for keys with identity $\mathsf{ID}_i \equiv \mathsf{ID}^* \mod p_1$. It prevents $\mathcal{A}$ to obtain the secret key for ID^*, which obviously helps $\mathcal{A}$ to decrypt the challenge ciphertext. Moreover, we also restrict $\mathcal{A}$ in $\text{Game}_{restricted}$ for not asking for decryption of ciphertext $(\hat{C}, \mathsf{vk}, \sigma)$ for identity ID, where $\mathsf{ID}_i \equiv \mathsf{ID}^* \mod p_1$ and $\mathsf{vk} \equiv \mathsf{vk}^* \mod p_1$. It prevents $\mathcal{A}$ to ask for the decryption of $(C^*, \mathsf{vk}, \sigma)$, which leads to a valid decryption query of the challenge ciphertext. Therefore if $\mathcal{A}$ can distinguish between $\text{Game}_{restricted}$ and Game_0, then $\mathcal{B}$ can break Assumption 1. □

Lemma 7. *If there exists an adversary $\mathcal{A}$ such that*

$$Adv_{\mathcal{A}}(\text{Game}_{k-1}) - Adv_{\mathcal{A}}(\text{Game}_k) = \epsilon,$$

then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.

Proof. Given $g, X_1X_2, X_3, Y_2Y_3, T$, $\mathcal{B}$ can simulate Game_{k-1} or Game_k with $\mathcal{A}$. $\mathcal{B}$ chooses random $a, b, c, d, \alpha \in \mathbb{Z}_N$ and sets

$$g_1 = g, \quad v = g^a, \quad w = g^b, \quad u = g^c, \quad h = g^d.$$

$\mathcal{B}$ sends the master public key $(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, u, h, v, w, \hat{e}(g_1, u)^\alpha, \hat{e}(g_1, h)^\alpha)$ to $\mathcal{A}$.

For the i-th extraction oracle query with input ID_i:

- if $i > k$, $\mathcal{B}$ chooses random $r_i, \rho_i, \mu_i, \theta_i \in \mathbb{Z}_N$ and returns the normal key

$$D_1 = (u^{\mathsf{ID}_i} h)^\alpha w^{r_i} X_3^{\rho_i}, \quad D_2 = g_1^{r_i} X_3^{\mu_i}, \quad D_3 = v^{r_i} X_3^{\theta_i}.$$

- if $i < k$, $\mathcal{B}$ chooses random $r_i, \rho_i, \mu_i, \theta_i \in \mathbb{Z}_N$ and returns the semi-functional key

$$D_1 = (u^{\mathsf{ID}_i} h)^\alpha w^{r_i} (Y_2Y_3)^{\rho_i}, \quad D_2 = g_1^{r_i} (Y_2Y_3)^{\mu_i}, \quad D_3 = v^{r_i} (Y_2Y_3)^{\theta_i}.$$

  This is a semi-functional key with $g_2^\gamma = Y_2^{\mu_i}$, $z_k = \rho_i/\mu_i$ and $z'_k = \theta_i/\mu_i$. Note that the value of ρ_i, μ_i, θ_i modulo p_2 and modulo p_3 are uncorrelated.

- if $i = k$, $\mathcal{B}$ chooses random $\rho_k, \mu_k, \theta_k \in \mathbb{Z}_N$ and returns the key:

$$D_1 = (u^{\mathsf{ID}_i} h)^\alpha T^b X_3^{\rho_k}, \quad D_2 = T X_3^{\mu_k}, \quad D_3 = T^a X_3^{\theta_k}.$$

  If $T \in \mathbb{G}_{p_1p_3}$, then this is a normal key and hence $\mathcal{B}$ simulates Game_{k-1}. If $T \in \mathbb{G}$, then this is a semi-functional key with g_2^γ is the $\mathbb{G}_{p_2}$ part of T, $z_k = b$ and $z'_k = a$. Hence $\mathcal{B}$ simulates Game_k. Note that the value of a and b modulo p_1 and modulo p_2 are uncorrelated.

For each decryption oracle query with input (C_i, ID_i), $\mathcal{B}$ extracts the normal key for ID_i as above and then decrypts following the **Decrypt** algorithm.

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0, 1\}$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{B}$ calculates:

$$C_1^* = X_1X_2, \quad C_2^* = (X_1X_2)^{a \cdot \mathsf{vk}^* + b}, \quad C_3^* = M_{b'}^* \cdot \hat{e}((X_1X_2)^\alpha, u^{\mathsf{ID}^*} h),$$

and $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(C_1^*, C_2^*, C_3^*)$. The challenge ciphertext is $((C_1^*, C_2^*, C_3^*), \sigma^*, \mathsf{vk}^*)$. It is a semi-functional ciphertext with $g^s = X_1$ and $z_c = a \cdot \mathsf{vk}^* + b$.

If $\mathcal{B}$ attempts to test if the key k is semi-functional by creating a semi-functional ciphertext for ID_i and trying to decrypt, then decryption will work no matter key k is semi-functional or not, since $z_c = a \cdot \mathsf{vk}^* + b$, $z_k = b$ and $z'_k = a$. It means that $\mathcal{B}$ can only create a nominally semi-functional key k. Note that the values z_k and z'_k

mod p_2 is hidden from $\mathcal{A}$ by the random $\mathbb{G}_{p_3}$ elements $X_3^{\theta_k}$ and $X_3^{\rho_k}$ respectively. If the subgroup decision assumption holds, $\mathcal{A}$ cannot remove the $\mathbb{G}_{p_3}$ part of D_1 and D_2[12]. Therefore $z_c \mod p_2$ appears to be randomly distributed to $\mathcal{A}$, as the value of a and b modulo p_1 and modulo p_2 are uncorrelated.

Therefore if $\mathcal{A}$ can distinguish between Game_{k-1} and Game_k, then $\mathcal{B}$ can break Assumption 2. □

Lemma 8. *If there exists an adversary $\mathcal{A}$ such that*

$$Adv_{\mathcal{A}}(\text{Game}_{q_k}) - Adv_{\mathcal{A}}(\text{Game}_{final}) = \epsilon,$$

then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.

Proof. Given $g, g^{\alpha}X_2, X_3, g^{s}Y_2, Z_2, T$, $\mathcal{B}$ can simulate Game_{q_k} or Game_{final} with $\mathcal{A}$. $\mathcal{B}$ chooses random $a, b, c, d, \in \mathbb{Z}_N$ and sets

$$g_1 = g, \quad v = g^a, \quad w = g^b, \quad u = g^c, \quad h = g^d.$$

$\mathcal{B}$ sends the master public key $(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, u, h, v, w, \hat{e}(g_1^{\alpha}X_2, u), \hat{e}(g_1^{\alpha}X_2, h))$ to $\mathcal{A}$.

For the i-th extraction oracle query with input ID_i, $\mathcal{B}$ chooses random $r_i, \phi_i, \rho_i, \eta_i, \mu_i$, $\nu_i, \theta_i \in \mathbb{Z}_N$ and returns the semi-functional key

$$D_1 = (g_1^{\alpha}X_2)^{c\cdot\mathsf{ID}_i+d} w^{r_i} Z_2^{\phi_i} X_3^{\rho_i}, \quad D_2 = g_1^{r_i} Z_2^{\eta_i} X_3^{\mu_i}, \quad D_3 = v^{r_i} Z_2^{\nu_i} X_3^{\theta_i}.$$

For each decryption oracle query with input $((C_i, \mathsf{vk}_i, \sigma_i), \mathsf{ID}_i)$, $\mathcal{B}$ extracts the semi-functional key for ID_i as above and then decrypts following the **Decrypt** algorithm. If the ciphertext is normal, then the semi-functional key can decrypt correctly. It is indistinguishable from the decryption oracle using the normal key[13].

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0, 1\}$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^{\lambda})$. $\mathcal{B}$ calculates:

$$C_1^* = g^s Y_2, \quad C_2^* = (g^s Y_2)^{a\cdot\mathsf{vk}^*+b}, \quad C_3^* = M_{b'}^* \cdot \hat{e}(T, u^{\mathsf{ID}^*}h),$$

This sets $z_c = a \cdot \mathsf{vk}^* + b$. Note that the value of z_c modulo p_2 and the value of a and b modulo p_1 are uncorrelated.

[12]For example in Assumption 2, if $\mathcal{A}$ can remove the $\mathbb{G}_{p_3}$ part of Y_2Y_3, then $\mathcal{A}$ can obtain Y_2. If $\hat{e}(Y_2, T)$ is the identity, then $T \in \mathbb{G}_{p_1p_3}$. Otherwise, $T \in \mathbb{G}$. Then Assumption 2 is broken. Note that in [139], they used a different "pairwise independent" argument for the random distribution of z_c. It cannot be applied here since $z_k = b \mod p_2$ and $z'_k = a \mod p_2$.

[13]In [139], their IND-ID-CPA proof does not have this additional restriction, since they do not have the decryption oracle.

If $T = g^{\alpha s}$, then this is a proper semi-functional ciphertext with message M_b^* and hence $\mathcal{B}$ simulates Game_{q_k}. If T is a random element in $\mathbb{G}_T$, then this is a semi-functional ciphertext with a random message and hence $\mathcal{B}$ simulates Game_{final}.

We have restricted that $\mathcal{A}$ cannot query decryption oracle query with input $((C_i, \mathsf{vk}_i, \sigma_i), \mathsf{ID}_i)$, where $\mathsf{vk}_i \equiv \mathsf{vk}^* \mod p_1$ and $\mathsf{ID}_i \equiv \mathsf{ID}^* \mod p_1$. It prevents the semi-functional challenge ciphertext from being queried for the identity ID^*,

Therefore if $\mathcal{A}$ can distinguish between Game_{q_k} and Game_{final}, then $\mathcal{B}$ can break Assumption 3. □

Finally in Game_{final}, the value of b is information theoretically hidden from $\mathcal{A}$. Hence $\mathcal{A}$ has no advantage in winning Game_{final}. If Assumption 1, Assumption 2 and Assumption 3 hold, then Game_{real} is indistinguishable from the Game_{final}. Hence the attacker has negligible advantage in winning Game_{real}.

For the success probability of $\mathcal{B}$, $\mathcal{B}$ has to guess from which point $\mathcal{A}$ can distinguish the games: $\text{Game}_{restrict}$, Game_0, Game_1,...,Game_{q_k} or Game_{final}. $\mathcal{B}$ can only first take a guess and setup the game accordingly. $\mathcal{B}$ guess correctly with probability at least $1/(q_k + 3)$. □

Theorem 23. *Our IBE scheme (ϵ, t, q_k, q_d)-ANON-ID-CCA secure if (ϵ', t')-Assumption 1, (ϵ', t')-Assumption 2 and (ϵ', t')-Assumption 3 hold, where*

$$\epsilon' \geq \frac{\epsilon}{q_k + 3}, \quad t' = t + O(q_k\rho + q_d(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}$ and a pairing operation respectively.

Proof. The proof is almost the same as the proof of the IND-ID-CCA security. In the challenge phase of the IND-ID-CCA proof, the adversary returns a challenge identity ID^* and two challenge messages m_0^*, m_1^*. The simulator $\mathcal{B}$ computes the challenge ciphertext using ID^* and $m_{b'}^*$, where $b' \in_R \{0, 1\}$. Now in the challenge phase of the ANON-ID-CCA proof, the adversary returns a challenge message m^* and two challenge identities $\mathsf{ID}_0^*, \mathsf{ID}_1^*$. The simulator $\mathcal{B}$and two challenge identities $\mathsf{ID}_0^*, \mathsf{ID}_1^*$. The simulator $\mathcal{B}$ computes the challenge ciphertext using $\mathsf{ID}_{b'}^*$ and m^*, where $b' \in_R \{0, 1\}$. It will not change the distribution of the challenge ciphertext in both the proof of lemma 6, lemma 7 and lemma 8. Therefore if Assumption 1, Assumption 2 and Assumption 3 hold, then Game_{real} is indistinguishable from the Game_{final}. Hence the attacker has negligible advantage in winning Game_{real}. □

Theorem 24. *Our IBE scheme (ϵ, t, q_e)-ACI-PKG secure if (ϵ, t')-Assumption 4 holds, where*

$$t' = t + O(q_e(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}$ and an exponentiation in $\mathbb{G}_T$ respectively.

Proof. Suppose there is an adversary $\mathcal{A}$ who breaks the ACI-PKG security. We construct an algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the subgroup decisional bilinear problem with non-negligible advantage. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows:

Setup: The adversary $\mathcal{A}$ gives the master public key $\mathsf{mpk} = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, u, h, v, w, \hat{e}(g_1^\alpha, u), \hat{e}(g_1^\alpha, h))$ to $\mathcal{B}$. $\mathcal{B}$ aborts if the mpk is not valid. $\mathcal{B}$ gives $((N, \mathbb{G}, \mathbb{G}_T, \hat{e}), g_1, v, w)$ to the challenger of the subgroup decisional bilinear problem instance. After that, $\mathcal{B}$ obtains $g_1^s, v^s, w^s\mathbb{G}$ and $\hat{e}(g_1, g_1)^x, T \in \mathbb{G}_T$ from the problem instance.

Encryption Oracle: On input a message m_i in the message space $\mathcal{M}$, $\mathcal{B}$ runs $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, picks a random $s_i \in \mathbb{Z}_p^*$ and calculates:

$$C_1 = g_1^{s_i}, \quad C_2 = (v^{\mathsf{vk}_i} w)^{s_i}, \quad C_3 = m_i \cdot (\hat{e}(g_1, g_1)^x)^{s_i}.$$

Denote $\hat{C}_i = (C_1, C_2, C_3)$. $\mathcal{B}$ signs $\sigma_i \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(\hat{C}_i)$ and returns the ciphertext $(\hat{C}_i, \sigma_i, \mathsf{vk}_i)$.

Notice that this encryption oracle output implicitly sets the challenge identity ID^* such that $\hat{e}(g_1^\alpha, u^{\mathsf{ID}^*} h) = \hat{e}(g_1, g_1)^x$. It implies that ID^* is not known to both $\mathcal{A}$ and $\mathcal{B}$.

Challenge: Eventually, $\mathcal{A}$ will output two messages (m_0^*, m_1^*). $\mathcal{B}$ runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$, picks a random bit b' and calculates:

$$C_1^* = g_1^s, \quad C_2^* = (v^s)^{\mathsf{vk}^*} w^s, \quad C_3^* = m_{b'}^* \cdot T.$$

Denote $\hat{C}^* = (C_1^*, C_2^*, C_3^*)$. $\mathcal{B}$ signs $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(\hat{C}^*)$ and returns the challenge ciphertext $(\hat{C}^*, \sigma^*, \mathsf{vk}^*)$.

Guess: Finally $\mathcal{A}$ outputs its guess $b^* \in \{0, 1\}$. If $b^* = b'$, $\mathcal{B}$ outputs $T = \hat{e}(g_1, g_1)^{xs}$; otherwise, it outputs $T \in \mathbb{G}_T$. Therefore if Assumption 4 holds, then the adversary has negligible advantage in winning the ACI-PKG game. □

Theorem 25. *Our IBE scheme is information theoretically secure against the ANON-PKG attack.*

Proof. In the challenge ciphertext $(\hat{C}^* = (C_1^*, C_2^*, C_3^*), \sigma^*, \mathsf{vk}^*)$, C_1^*, C_2^*, σ^* and vk^* does not contain any information about the user identity. On the other hand, C_3^* appears random to the adversary $\mathcal{A}$ since it contains the challenge message m^* which is unknown to $\mathcal{A}$. □

5.4.4 Our Second Construction in Prime Order Groups

Our second IBE construction in the previous Section uses the composite order bilinear groups, in order to give a simple construction for the ease of understanding. However, the size of group elements in these groups are much larger than that in prime order groups. We use the approach by Lewko and Waters [139] to convert our second IBE in prime order groups. We first convert our assumptions 1, 2, 3, 4 into assumptions 1', 2', 3', 4' in prime order groups respectively. We then prove the security of our IBE in prime order groups under these new static assumptions.

We now define a group generator $\mathcal{G}$, which takes a security parameter 1^λ as input, outputs a description of bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. We require that the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and the bilinear map $\hat{e}$ are computable in polynomial time with respect to λ.

CCA-secure Construction

Let $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme in which the verification key outputted by $\mathsf{Gen}(1^\lambda)$ is in $\mathbb{Z}_N$.

- **Setup**: The setup algorithm runs bilinear group generator $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. The PKG picks a random $\alpha, \beta, \gamma, a, \tau \in \mathbb{Z}_p$ and random generators $g_1 \in \mathbb{G}_1, g_2, u_2, h_2, x_2, y_2 \in \mathbb{G}_2$. It sets

$$v_1 = g_1^\beta, \quad w_1 = g_1^\gamma, \quad v_2 = g_2^\beta, \quad w_2 = g_2^\gamma, \quad f_2 = (x_2 y_2^a)^{1/\tau}.$$

  The master public key is

$$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, v_1, w_1, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau, \hat{e}(g_1, u_2)^\alpha, \hat{e}(g_1, h_2)^\alpha).$$

  The master secret key is $(\alpha, g_2, u_2, h_2, v_2, w_2, x_2, y_2, f_2)$.

- **Extract**: Given the master secret key $(\alpha, g_2, u_2, h_2, v_2, w_2, f_2)$ and an identity ID, pick some random $r, k_1, k_2, k_3 \in \mathbb{Z}_p$. Then calculate the identity-based secret key sk_{ID}:

$$\begin{aligned}
&D_{1,1} = (u_2^{\mathsf{ID}} h_2)^\alpha w_2^r x_2^{k_1}, && D_{1,2} = y_2^{k_1}, && D_{1,3} = f_2^{k_1}, \\
&D_{2,1} = g_2^r x_2^{k_2}, && D_{2,2} = y_2^{k_2}, && D_{2,3} = f_2^{k_2}, \\
&D_{3,1} = v_2^r x_2^{k_3}, && D_{3,2} = y_2^{k_3}, && D_{3,3} = f_2^{k_3}.
\end{aligned}$$

- **Encrypt**: To encrypt a message $m \in \mathbb{G}_T$ for a user ID, the sender picks a random $s \in \mathbb{Z}_p$. The sender runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and calculates:

$$\begin{aligned}
&C_{1,1} = g_1^s, && C_{1,2} = g_1^{as}, && C_{1,3} = g_1^{-\tau s},\\
&C_{2,1} = (v_1^{\mathsf{vk}} w_1)^s, && C_{2,2} = (v_1^{\mathsf{vk}} w_1)^{as}, && C_{2,3} = (v_1^{\mathsf{vk}} w_1)^{-\tau s},\\
&C_3 = m \cdot (\hat{e}(g_1, u_2)^{\alpha \mathsf{ID}} \cdot \hat{e}(g_1, h_2)^{\alpha})^s.
\end{aligned}$$

  Denote $\hat{C} = (C_{1,1}, C_{1,2}, C_{1,3}, C_{2,1}, C_{2,2}, C_{2,3}, C_3)$. The sender signs $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\hat{C})$ and sends the ciphertext $(\hat{C}, \sigma, \mathsf{vk})$.

- **Decrypt**: Given a ciphertext $(\hat{C}, \sigma, \mathsf{vk})$, and a secret key sk_{ID} for an identity ID, the recipient outputs $\perp$ if $\mathsf{Vrfy}_{\mathsf{vk}}(\hat{C}, \sigma) \neq 1$. Otherwise, the recipient calculates:

$$m = \frac{C_3 \cdot \hat{e}(C_{2,1}, D_{2,1}) \cdot \hat{e}(C_{2,2}, D_{2,2}) \cdot \hat{e}(C_{2,3}, D_{2,3})}{\hat{e}(C_{1,1}, D_{1,1} D_{3,1}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,2}, D_{1,2} D_{3,2}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,3}, D_{1,3} D_{3,3}^{\mathsf{vk}})}.$$

  The correctness is shown as follows:

$$\begin{aligned}
&\hat{e}(C_{2,1}, D_{2,1}) \cdot \hat{e}(C_{2,2}, D_{2,2}) \cdot \hat{e}(C_{2,3}, D_{2,3})\\
=&\hat{e}((v_1^{\mathsf{vk}} w_1)^s, g_2^r x_2^{k_2}) \cdot \hat{e}((v_1^{\mathsf{vk}} w_1)^{as}, y_2^{k_2}) \cdot \hat{e}((v_1^{\mathsf{vk}} w_1)^{-\tau s}, f_2^{k_2})\\
=&\hat{e}(v_1^{\mathsf{vk}} w_1, g_2)^{rs} \cdot \hat{e}(v_1^{\mathsf{vk}} w_1, x_2 y_2^a f_2^{-\tau})^{s k_2}\\
=&\hat{e}(g_1, v_2^{\mathsf{vk}} w_2)^{rs},
\end{aligned}$$

and also

$$\begin{aligned}
&\hat{e}(C_{1,1}, D_{1,1} D_{3,1}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,2}, D_{1,2} D_{3,2}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,3}, D_{1,3} D_{3,3}^{\mathsf{vk}})\\
=&\hat{e}(g_1^s, (u_2^{\mathsf{ID}} h_2)^{\alpha} w_2^r x_2^{k_1} (v_2^r x_2^{k_3})^{\mathsf{vk}}) \cdot \hat{e}(g_1^{as}, y_2^{k_1} y_2^{k_3 \cdot \mathsf{vk}}) \cdot \hat{e}(g_1^{-\tau s}, f_2^{k_1} f_2^{k_3 \cdot \mathsf{vk}})\\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}} h_2)^{\alpha} (v_2^{\mathsf{vk}} w_2)^r)^s \cdot \hat{e}(g_1^s, x_2 y_2^a f_2^{-\tau})^{k_1 + k_3 \cdot \mathsf{vk}}\\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}} h_2)^{\alpha} (v_2^{\mathsf{vk}} w_2)^r)^s.
\end{aligned}$$

Security

To prove the security of our IBE scheme, we first define two additional structures: semi-functional keys and semi-functional ciphertexts. They are used in the security proofs only.

Semi-functional keys: A semi-functional key can be created from a normal key $(D_{1,1}, D_{1,2}, D_{1,3}, D_{2,1}, D_{2,2}, D_{2,3}, D_{3,1}, D_{3,2}, D_{3,3})$. Firstly, random exponents $\gamma, z_k, z_k' \in$

$\mathbb{Z}_p$ are chosen. Then the semi-functional key is set to be

$$\begin{aligned}
D'_{1,1} &= D_{1,1} f_2^{-a\gamma z_k}, & D'_{1,2} &= D_{1,2} f_2^{\gamma z_k}, & D'_{1,3} &= D_{1,3},\\
D'_{2,1} &= D_{2,1} f_2^{-a\gamma}, & D'_{2,2} &= D_{2,2} f_2^{\gamma}, & D'_{2,3} &= D_{2,3},\\
D'_{3,1} &= D_{3,1} f_2^{-a\gamma z'_k}, & D'_{3,2} &= D_{3,2} f_2^{\gamma z'_k}, & D'_{3,3} &= D_{3,3}.
\end{aligned}$$

The semi-functional key can decrypt the normal ciphertext correctly, since

$$\begin{aligned}
&\hat{e}(C_{2,1}, D'_{2,1}) \cdot \hat{e}(C_{2,2}, D'_{2,2}) \cdot \hat{e}(C_{2,3}, D'_{2,3})\\
=&\hat{e}((v_1^{\mathsf{vk}} w_1)^s, D_{2,1} f_2^{-a\gamma}) \cdot \hat{e}((v_1^{\mathsf{vk}} w_1)^{as}, D_{2,2} f_2^{\gamma}) \cdot \hat{e}((v_1^{\mathsf{vk}} w_1)^{-\tau s}, D_{2,3})\\
=&\hat{e}(g_1, v_2^{\mathsf{vk}} w_2)^{rs} \cdot \hat{e}(v_1^{\mathsf{vk}} w_1, f_2^{-a\gamma} f_2^{a\gamma})^s\\
=&\hat{e}(g_1, v_2^{\mathsf{vk}} w_2)^{rs},
\end{aligned}$$

and also

$$\begin{aligned}
&\hat{e}(C_{1,1}, D'_{1,1} {D'_{3,1}}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,2}, D'_{1,2} {D'_{3,2}}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,3}, D'_{1,3} {D'_{3,3}}^{\mathsf{vk}})\\
=&\hat{e}(g_1^s, D_{1,1} f_2^{-a\gamma z_k} (D_{3,1} f_2^{-a\gamma z'_k})^{\mathsf{vk}}) \cdot \hat{e}(g_1^{as}, D_{1,2} f_2^{\gamma z_k} (D_{3,2} f_2^{\gamma z'_k})^{\mathsf{vk}}) \cdot \hat{e}(g_1^{-\tau s}, D_{1,3} D_{3,3}^{\mathsf{vk}})\\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}} h_2)^{\alpha} (v_2^{\mathsf{vk}} w_2)^r)^s \cdot \hat{e}(g_1, f_2^{-(z_k + z'_k \cdot \mathsf{vk})} f_2^{z_k + z'_k \cdot \mathsf{vk}})^{sa\gamma}\\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}} h_2)^{\alpha} (v_2^{\mathsf{vk}} w_2)^r)^s.
\end{aligned}$$

Semi-functional ciphertexts: Denote $f_1, y_1 \in \mathbb{G}_1$ such that $\log_{f_1} y_1 = \log_{f_2} y_2$. A semi-functional ciphertext can be created by running $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and calculating $\hat{C} = (C_{1,1}, C_{1,2}, C_{1,3}, C_{2,1}, C_{2,2}, C_{2,3}, C_3)$ as in **Encrypt**. After that, random exponents $\delta, z_c \in \mathbb{Z}_p$ are chosen. Calculate

$$\begin{aligned}
C'_{1,1} &= C_{1,1}, & C'_{1,2} &= C_{1,2} f_1^{\delta}, & C'_{1,3} &= C_{1,3} y_1^{-\delta},\\
C'_{2,1} &= C_{2,1}, & C'_{2,2} &= C_{2,2} f_1^{\delta z_c}, & C'_{2,3} &= C_{2,3} y_1^{-\delta z_c},\\
C'_3 &= C_3.
\end{aligned}$$

Set $C' = (C'_{1,1}, C'_{1,2}, C'_{1,3}, C'_{2,1}, C'_{2,2}, C'_{2,3}, C'_3)$ and sign $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(C')$. Then the semi-functional ciphertext is set to be $(C', \mathsf{vk}, \sigma)$.

The normal key can decrypt the semi-functional ciphertext correctly, since

$$\begin{aligned}
&\hat{e}(C'_{2,1}, D_{2,1}) \cdot \hat{e}(C'_{2,2}, D_{2,2}) \cdot \hat{e}(C'_{2,3}, D_{2,3})\\
=&\hat{e}(C_{2,1}, D_{2,1}) \cdot \hat{e}(C_{2,2} f_1^{\delta z_c}, y_2^{k_2}) \cdot \hat{e}(C_{2,3} y_1^{-\delta z_c}, f_2^{k_2})\\
=&\hat{e}(g_1, v_2^{\mathsf{vk}} w_2)^{rs} \cdot \hat{e}(f_1^{\delta z_c}, y_2^{k_2}) \cdot \hat{e}(y_1^{-\delta z_c}, f_2^{k_2})\\
=&\hat{e}(g_1, v_2^{\mathsf{vk}} w_2)^{rs},
\end{aligned}$$

and also

$$\begin{aligned}
&\hat{e}(C'_{1,1}, D_{1,1}D_{3,1}^{\mathsf{vk}}) \cdot \hat{e}(C'_{1,2}, D_{1,2}D_{3,2}^{\mathsf{vk}}) \cdot \hat{e}(C'_{1,3}, D_{1,3}D_{3,3}^{\mathsf{vk}}) \\
=&\hat{e}(C_{1,1}, D_{1,1}D_{3,1}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,2}f_1^{\delta}, y_2^{k_1}y_2^{k_3\cdot\mathsf{vk}}) \cdot \hat{e}(C_{1,3}y_1^{-\delta}, f_2^{k_1}f_2^{k_3\cdot\mathsf{vk}}) \\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}}h_2)^{\alpha}(v_2^{\mathsf{vk}}w_2)^{r})^{s} \cdot \hat{e}(f_1^{\delta}, y_2)^{k_1+k_3\cdot\mathsf{vk}} \cdot \hat{e}(y_1^{-\delta}, f_2)^{k_1+k_3\cdot\mathsf{vk}} \\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}}h_2)^{\alpha}(v_2^{\mathsf{vk}}w_2)^{r})^{s}.
\end{aligned}$$

Nominally semi-functional keys and ciphertexts: If a semi-functional key is used to decrypt a semi-functional ciphertext:

$$\begin{aligned}
&\hat{e}(C'_{2,1}, D'_{2,1}) \cdot \hat{e}(C'_{2,2}, D'_{2,2}) \cdot \hat{e}(C'_{2,3}, D'_{2,3}) \\
=&\hat{e}(C_{2,1}, D'_{2,1}) \cdot \hat{e}(C_{2,2}f_1^{\delta z_c}, D'_{2,2}) \cdot \hat{e}(C_{2,3}y_1^{-\delta z_c}, D'_{2,3}) \\
=&\hat{e}(g_1, v_2^{\mathsf{vk}}w_2)^{rs} \cdot \hat{e}(f_1^{\delta z_c}, D_{2,2}f_2^{\gamma}) \cdot \hat{e}(y_1^{-\delta z_c}, D_{2,3}) \\
=&\hat{e}(g_1, v_2^{\mathsf{vk}}w_2)^{rs} \cdot \hat{e}(f_1^{\delta z_c}, y_2^{k_2}f_2^{\gamma}) \cdot \hat{e}(y_1^{-\delta z_c}, f_2^{k_2}) \\
=&\hat{e}(g_1, v_2^{\mathsf{vk}}w_2)^{rs} \cdot \hat{e}(f_1, f_2)^{\gamma\delta z_c},
\end{aligned}$$

and also

$$\begin{aligned}
&\hat{e}(C'_{1,1}, D'_{1,1}{D'_{3,1}}^{\mathsf{vk}}) \cdot \hat{e}(C'_{1,2}, D'_{1,2}{D'_{3,2}}^{\mathsf{vk}}) \cdot \hat{e}(C'_{1,3}, D'_{1,3}{D'_{3,3}}^{\mathsf{vk}}) \\
=&\hat{e}(C_{1,1}, D'_{1,1}{D'_{3,1}}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,2}f_1^{\delta}, D'_{1,2}{D'_{3,2}}^{\mathsf{vk}}) \cdot \hat{e}(C_{1,3}y_1^{-\delta}, D'_{1,3}{D'_{3,3}}^{\mathsf{vk}}) \\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}}h_2)^{\alpha}(v_2^{\mathsf{vk}}w_2)^{r})^{s} \cdot \hat{e}(f_1^{\delta}, D_{1,2}f_2^{\gamma z_k}(D_{3,2}f_2^{\gamma z'_k})^{\mathsf{vk}}) \cdot \hat{e}(y_1^{-\delta}, D_{1,3}D_{3,3}^{\mathsf{vk}}) \\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}}h_2)^{\alpha}(v_2^{\mathsf{vk}}w_2)^{r})^{s} \cdot \hat{e}(f_1, f_2)^{\gamma\delta(z_k+z'_k\cdot\mathsf{vk})} \cdot \hat{e}(f_1^{\delta}, y_2^{k_1}y_2^{k_3\cdot\mathsf{vk}}) \cdot \hat{e}(y_1^{-\delta}, f_2^{k_1}f_2^{k_3\cdot\mathsf{vk}}) \\
=&\hat{e}(g_1, (u_2^{\mathsf{ID}}h_2)^{\alpha}(v_2^{\mathsf{vk}}w_2)^{r})^{s} \cdot \hat{e}(f_1, f_2)^{\gamma\delta(z_k+z'_k\cdot\mathsf{vk})}.
\end{aligned}$$

There will be an extra blinding factor $\hat{e}(f_1, f_2)^{\gamma\delta(z_k+\mathsf{vk}\cdot z'_k-z_c)}$ to hinder the decryption. If $z_c = z_k + \mathsf{vk}\cdot z'_k$, decryption still works. We call this key and ciphertext as nominally semi-functional.

Complexity Assumptions We have the following new assumptions for the security proofs.

Assumption 1'. Given a group generator $\mathcal{G}$, we define the following experiment:

Experiment $\mathbf{Exp}^{\text{Assumption1}'}_{\mathcal{G},\mathcal{A}_{1'},\beta}(1^{\lambda})$

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}(1^{\lambda}), \quad f_1 \xleftarrow{R} \mathbb{G}_1, \quad f_2 \xleftarrow{R} \mathbb{G}_2, \quad a, b, s \xleftarrow{R} \mathbb{Z}_p,$

$T_0 = f_1^{asb^s}, \quad T_1 \xleftarrow{R} \mathbb{G}_1.$

Return $\beta' \leftarrow \mathcal{A}_{1'}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, f_1, f_1^a, f_1^b, f_1^s, f_1^{as},$
$f_1^{bs}, f_1^{b^2}, f_1^{ab^2}, f_1^{b^2s}, f_1^{b^3}, f_1^{b^3s}, f_2, f_2^b, T_{\beta}).$

We define the advantage of an algorithm $\mathcal{A}_{1'}$ in breaking Assumption 1' to be:

$$Adv_{\mathcal{G},\mathcal{A}_{1'}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\text{Assumption1}'}_{\mathcal{G},\mathcal{A}_{1'},1}(1^\lambda) = 1] - \mathbf{Exp}^{\text{Assumption1}'}_{\mathcal{G},\mathcal{A}_{1'},0}(1^\lambda) = 1]\right|.$$

Definition 37. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 1 if $Adv_{\mathcal{G},\mathcal{A}_1}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_1$.*

Assumption 2'. Given a group generator $\mathcal{G}$, we define the following experiment:

Experiment $\mathbf{Exp}^{\text{Assumption2}'}_{\mathcal{G},\mathcal{A}_{2'},\beta}(1^\lambda)$

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}(1^\lambda), \quad b, c, d, x \xleftarrow{R} \mathbb{Z}_p,$

$f_1 \xleftarrow{R} \mathbb{G}_1, \quad f_2 \xleftarrow{R} \mathbb{G}_2, \quad T_0 = f_2^{bc}, \quad T_1 \xleftarrow{R} \mathbb{G}_2.$

Return $\beta' \leftarrow \mathcal{A}_{2'}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, f_1, f_1^d, f_1^{d^2}, f_1^{bx}, f_1^{bdx}, f_1^{d^2x}, f_2, f_2^b, f_2^c, f_2^d, T_\beta)$.

We define the advantage of an algorithm $\mathcal{A}_{2'}$ in breaking Assumption 2' to be:

$$Adv_{\mathcal{G},\mathcal{A}_{2'}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\text{Assumption2}'}_{\mathcal{G},\mathcal{A}_{2'},1}(1^\lambda) = 1] - \mathbf{Exp}^{\text{Assumption2}'}_{\mathcal{G},\mathcal{A}_{2'},0}(1^\lambda) = 1]\right|.$$

Definition 38. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 2' if $Adv_{\mathcal{G},\mathcal{A}_{2'}}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_{2'}$.*

Assumption 3'. Given a group generator $\mathcal{G}$, we define the following experiment:

Experiment $\mathbf{Exp}^{\text{Assumption3}'}_{\mathcal{G},\mathcal{A}_{3'},\beta}(1^\lambda)$

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}(1^\lambda), \quad a, b, c \xleftarrow{R} \mathbb{Z}_p,$

$f_1 \xleftarrow{R} \mathbb{G}_1, \quad f_2 \xleftarrow{R} \mathbb{G}_2, \quad T_0 = f_1^{abc}, \quad T_1 \xleftarrow{R} \mathbb{G}_1.$

Return $\beta' \leftarrow \mathcal{A}_{3'}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, f_1, f_1^a, f_1^b, f_1^c, f_2^a, f_2^b, f_2^c, T_\beta)$.

We define the advantage of an algorithm $\mathcal{A}_3$ in breaking Assumption 3 to be:

$$Adv_{\mathcal{G},\mathcal{A}_3}(\lambda) := \left|\Pr[\mathbf{Exp}^{\text{Assumption3}}_{\mathcal{G},\mathcal{A}_3,1}(1^\lambda) = 1] - \mathbf{Exp}^{\text{Assumption3}}_{\mathcal{G},\mathcal{A}_3,0}(1^\lambda) = 1]\right|.$$

Definition 39. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 3 if $Adv_{\mathcal{G},\mathcal{A}_3}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_3$.*

Assumption 4'. Given a group generator $\mathcal{G}$, we define the following experiment:

Experiment $\mathbf{Exp}^{\text{Assumption4}'}_{\mathcal{G},\mathcal{A}_{4'},\beta}(1^\lambda)$

$((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda), g_A, v_A, w_A, g_B, v_B, w_B, g_C, v_C, w_C, \in \mathbb{G}_1^9) \leftarrow \mathcal{A}_{4'}(1^\lambda),$

$x, s \xleftarrow{R} \mathbb{Z}_p, \quad g_2 \xleftarrow{R} \mathbb{G}_2, \quad T_0 = \hat{e}(g_A, g_2)^{sx}, \quad T_1 \xleftarrow{R} \mathbb{G}_T.$

Return $\beta' \leftarrow \mathcal{A}_{4'}(g_A^s, v_A^s, w_A^s, g_B^s, v_B^s, w_B^s, g_C^s, v_C^s, w_C^s, \hat{e}(g_A, g_2)^x, T_\beta)$.

We define the advantage of an algorithm $\mathcal{A}_4$ in breaking Assumption 4 to be:

$$Adv_{\mathcal{G},\mathcal{A}_{4'}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\text{Assumption4}'}_{\mathcal{G},\mathcal{A}_{4'},1}(1^\lambda) = 1] - \mathbf{Exp}^{\text{Assumption4}'}_{\mathcal{G},\mathcal{A}_{4'},0}(1^\lambda) = 1]\right|.$$

Definition 40. *We say that $\mathcal{G}$ satisfies (ϵ, t)-Assumption 4' if $Adv_{\mathcal{G},\mathcal{A}_{4'}}(\lambda) = \epsilon$ is a negligible function of λ for any polynomial time t algorithm $\mathcal{A}_{4'}$.*

Theorem 26. *Our IBE scheme (ϵ, t, q_k, q_d)-IND-ID-CCA secure if (ϵ', t')-Assumption 1', (ϵ', t')-Assumption 2' and (ϵ', t')-Assumption 3' hold, where*

$$\epsilon' \geq \frac{\epsilon}{q_k + 2}, \quad t' = t + O(q_k\rho + q_d(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}_2$ and a pairing operation respectively.

Proof. We prove the IND-ID-CCA security by a hybrid argument using a sequence of games. The first game Game_{real} is the real IND-ID-CCA game. For $k = 0$ to q_k, we define Game_k as:

Game$_k$: It is the same as Game_{real}, except that the challenge ciphertext is semi-functional and the first k keys outputted from the extraction oracle are semi-functional. The rest of the keys are normal.

In other words, all keys are normal and the challenge ciphertext is semi-functional in Game_0. In Game_{q_k}, all keys and the challenge ciphertext are semi-functional. The last game is Game_{final}, which is the same as Game_{q_k} except that:

- the decryption oracle uses the semi-functional key to decrypt;
- the challenge ciphertext is a semi-functional encryption of a random message, instead of one of the two challenge messages.

We will prove the indistinguishability between all these games.

Lemma 9. *If there exists an adversary $\mathcal{A}$ such that $Adv_{\mathcal{A}}(Game_{real})$ - $Adv_{\mathcal{A}}(Game_0)$ $= \epsilon$, then we can construct an algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 1'.*

Proof. Given $(f_1, f_1^a, f_1^b, f_1^s, f_1^{as}, f_1^{bs}, f_1^{b^2}, f_1^{ab^2}, f_1^{b^2 s}, f_1^{b^3}, f_1^{b^3 s}, f_2, f_2^b, T)$, $\mathcal{B}$ can simulate Game_0 or Game_{real} with $\mathcal{A}$. $\mathcal{B}$ chooses random $\alpha, A, B, t_g, t_v, t_w, t_y \in \mathbb{Z}_p$, $u_2, h_2 \in \mathbb{G}_2$ sets

$\tau = b + at_y$ and calculates

$$
\begin{aligned}
g_1 &= f_1^{b^2} f_1^{t_g}, & g_1^a &= f_1^{ab^2}(f_1^a)^{t_g}, & g_1^\tau &= f_1^{b^3}(f_1^{ab^2})^{t_y}(f_1^b)^{t_g}(f_1^a)^{t_y t_g},\\
v_1 &= (f_1^{b^2})^A f_1^{t_v}, & v_1^a &= (f_1^{ab^2})^A (f_1^a)^{t_v}, & v_1^\tau &= (f_1^{b^3})^A (f_1^{ab^2})^{At_y}(f_1^b)^{t_v}(f_1^a)^{t_v t_y},\\
w_1 &= (f_1^{b^2})^B f_1^{t_w}, & w_1^a &= (f_1^{ab^2})^B (f_1^a)^{t_w}, & w_1^a &= (f_1^{b^3})^B (f_1^{ab^2})^{Bt_y}(f_1^b)^{t_w}(f_1^a)^{t_w t_y},\\
f_2 &= f_2, & x_2 &= f_2^b, & y_2 &= f_2^{t_y}.
\end{aligned}
$$

$\mathcal{B}$ sends the master public key $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, v_1, w_1, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau, \hat{e}(g_1, u_2)^\alpha, \hat{e}(g_1, h_2)^\alpha)$ to $\mathcal{A}$. The corresponding master secret key is $(\alpha, g_2, u_2, h_2, v_2, w_2, x_2, y_2, f_2)$, where $(g_2 = f_2^{b^2} f_2^{t_g}, v_2 = (f_2^{b^2})^A f_2^{t_v}, w_2 = (f_2^{b^2})^B f_2^{t_w})$ is unknown to $\mathcal{B}$.

For the i-th extraction oracle query with input ID_i, $\mathcal{B}$ chooses random $r_i, \rho_i, \mu_i, \theta_i \in \mathbb{Z}_p$ and sets

$$
\begin{aligned}
D_{1,1} &= (u_2^{\mathsf{ID}_i} h_2)^\alpha f_2^{t_w r_i}(f_2^b)^{\rho_i}, & D_{1,2} &= (f_2^b)^{-r_i B t_y} f_2^{\rho_i t_y} & D_{1,3} &= (f_2^b)^{-r_i B} f_2^{\rho_i}\\
D_{2,1} &= f_2^{t_g r_i}(f_2^b)^{\mu_i}, & D_{2,2} &= (f_2^b)^{-r_i t_y} f_2^{\mu_i t_y} & D_{2,3} &= (f_2^b)^{-r_i} f_2^{\mu_i}\\
D_{3,1} &= f_2^{t_v r_i}(f_2^b)^{\theta_i}, & D_{3,2} &= (f_2^b)^{-r_i A t_y} f_2^{\theta_i t_y} & D_{3,3} &= (f_2^b)^{-r_i A} f_2^{\theta_i}.
\end{aligned}
$$

It implicitly sets $k_1 = -br_iB + \rho_i$, $k_2 = -br_i + \mu_i$ and $k_3 = -br_iA + \theta_i$. For each decryption oracle query with input (C_i, ID_i), $\mathcal{B}$ extracts the secret key for ID_i as above and then decrypts following the **Decrypt** algorithm.

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0,1\}$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{B}$ calculates:

$$
\begin{aligned}
C_{1,1}^* &= f_1^{b^2 s}(f_1^s)^{t_g}, \quad C_{1,2}^* = T(f_1^{as})^{t_g}, \quad C_{1,3}^* = f_1^{-b^3 s} T^{-t_y}(f_1^{bs})^{-t_g}(f_1^{as})^{-t_y t_g},\\
C_{2,1}^* &= (f_1^{b^2 s})^{A\cdot\mathsf{vk}+B}(f_1^s)^{t_v\cdot\mathsf{vk}+t_w}, \quad C_{2,2}^* = T^{A\cdot\mathsf{vk}+B}(f_1^{as})^{t_v\cdot\mathsf{vk}+t_w},\\
C_{2,3}^* &= (f_1^{b^3 s})^{-A\cdot\mathsf{vk}-B} T^{-(A\cdot\mathsf{vk}+B)t_y}(f_1^{bs})^{-t_v\cdot\mathsf{vk}-t_w}(f_1^{as})^{-(t_v\cdot\mathsf{vk}+t_w)t_y},\\
C_3^* &= m\cdot\hat{e}(f_1^{b^2 s}(f_1^s)^{t_g}, u_2^{\mathsf{ID}} h_2).
\end{aligned}
$$

Denote $\hat{C} = (C_{1,1}^*, C_{1,2}^*, C_{1,3}^*, C_{2,1}^*, C_{2,2}^*, C_{2,3}^*, C_3^*)$. and $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(\hat{C})$. The challenge ciphertext is $(\hat{C}, \sigma^*, \mathsf{vk}^*)$.

If $T = f_1^{ab^2 s}$, this is a normal ciphertext and hence $\mathcal{B}$ simulates Game_{real}. If $T \in \mathbb{G}_p$, this is a semi-functional ciphertext with $z_c = A\cdot\mathsf{vk}^* + B$ and hence $\mathcal{B}$ simulates Game_0. Note that the value of $z_c \mod p$ is information theoretically hidden from the adversary $\mathcal{A}$. Therefore if $\mathcal{A}$ can distinguish between Game_{real} and Game_0, then $\mathcal{B}$ can break Assumption 1'. □

Lemma 10. *If there exists an adversary $\mathcal{A}$ such that $Adv_{\mathcal{A}}(Game_{k-1})$ - $Adv_{\mathcal{A}}(Game_k)$ = ϵ, then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2'.*

Proof. Given $(f_1, f_1^d, f_1^{d^2}, f_1^{bx}, f_1^{bdx}, f_1^{d^2x}, f_2, f_2^b, f_2^c, f_2^d, T)$, $\mathcal{B}$ can simulate Game_{k-1} or Game_k with $\mathcal{A}$. $\mathcal{B}$ chooses random $\alpha, a, A, B, t_v, t_w, t_x \in \mathbb{Z}_p$ and $u_2, h_2 \in \mathbb{G}_2$, sets $\tau = d + t_x$ and calculates

$$
\begin{aligned}
&g_1 = f_1^d, && v_1 = (f_1^d)^A f_1^{t_v}, && w_1 = (f_1^d)^B f_1^{t_w},\\
&g_1^\tau = f_1^{d^2}(f_1^d)^{t_x}, && v_1^\tau = (f_1^{d^2})^A (f_1^d)^{t_v + A} f_1^{t_v}, && w_1^\tau = (f_1^{d^2})^B (f_1^d)^{t_w + A} f_1^{t_w},\\
&g_2 = f_2^d, && v_2 = (f_2^d)^A f_2^{t_v}, && w_2 = (f_2^d)^B f_2^{t_w},\\
&x_2 = f_2^d (f_2^b)^{-a} f_2^{t_x}, && y_2 = f_2^b, && f_2 = f_2.
\end{aligned}
$$

$\mathcal{B}$ sends the master public key $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, v_1, w_1, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau, \hat{e}(g_1, u_2)^\alpha, \hat{e}(g_1, h_2)^\alpha)$ to $\mathcal{A}$. The corresponding master secret key is $(\alpha, g_2, u_2, h_2, v_2, w_2, x_2, y_2, f_2)$.

For the i-th extraction oracle query with input ID_i:

- if $i > k$, $\mathcal{B}$ chooses random $r_i, k_{i,1}, k_{i,2}, k_{i,3} \in \mathbb{Z}_p$ and calculates the normal key using the master secret key.
- if $i < k$, $\mathcal{B}$ chooses random $r_i, k_{i,1}, k_{i,2}, k_{i,3}, \gamma, z_k, z_k' \in \mathbb{Z}_p$ and calculates the semi-functional key using f_2 and a.
- if $i = k$, $\mathcal{B}$ chooses random $r', \rho, \mu, \theta \in \mathbb{Z}_p$ and returns the key:

$$
\begin{aligned}
&D_{1,1} = (u_2^{\mathsf{ID}_i} h_2)^\alpha (f_2^d)^{Br'} f_2^{t_w r'} (f_2^c)^{Bt_x - t_w} T^{-aB}, && D_{1,2} = T^B, && D_{1,3} = (f_2^c)^B,\\
&D_{2,1} = (f_2^d)^{r'} T^{-a} (f_2^c)^{t_x}, && D_{2,2} = T, && D_{2,3} = f_2^c,\\
&D_{3,1} = (f_2^d)^{Ar'} f_2^{t_v r'} (f_2^c)^{At_x - t_v} T^{-aA}, && D_{3,2} = T^A, && D_{3,3} = (f_2^c)^A.
\end{aligned}
$$

  It implicitly sets $k_1 = Bc$, $k_2 = c$, $k_3 = Ac$ and $r = r' - c$. If $T = f_2^{bc}$, then this is a normal key and hence $\mathcal{B}$ simulates Game_{k-1}. If $T \in \mathbb{G}$, then this is a semi-functional key with $z_k = B$ and $z_k' = A$. Hence $\mathcal{B}$ simulates Game_k.

For each decryption oracle query with input (C_i, ID_i), $\mathcal{B}$ extracts the normal key for ID_i as above and then decrypts following the **Decrypt** algorithm.

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0, 1\}$, a random $s' \in \mathbb{Z}_p$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{B}$ implicitly sets $s = bx + s'$, $\delta = -d^2 x$, $z_c = A \cdot \mathsf{vk}^* + B$ and calculates:

$$
\begin{aligned}
&C_{1,1}^* = f_1^{bdx}(f_1^d)^{s'}, \quad C_{1,2}^* = (f_1^{bdx})^a (f_1^d)^{as'} f_1^{-d^2 x}, \quad C_{1,3}^* = (f_1^{d^2})^{-s'} (f_1^{bdx})^{-t_x} (f_1^d)^{-t_x s'},\\
&C_{2,1}^* = (f_1^{bdx})^{A \cdot \mathsf{vk}^* + B} (f_1^{bx})^{t_v \cdot \mathsf{vk}^* + t_w} (f_1^d)^{s'(A \cdot \mathsf{vk} + B)} f_1^{s'(t_v \cdot \mathsf{vk}^* + t_w)}, \quad C_{2,2}^* = {C_{2,1}^*}^a (f_1^{d^2 x})^{-A \cdot \mathsf{vk}^* - B},\\
&C_{2,3}^* = (f_1^{bdx})^{(A - t_v) \cdot \mathsf{vk}^* + (B - t_w)} \cdot (f_1^{d^2})^{-s'(A \cdot \mathsf{vk}^* + B)}\\
&\qquad\qquad \cdot (f_1^d)^{s'((At_x - t_v) \cdot \mathsf{vk}^* + (Bt_x - t_w))} \cdot (f_1^{bx})^{t_x (t_v \cdot \mathsf{vk}^* + t_w)} \cdot f_1^{t_x s'(t_v \cdot \mathsf{vk}^* + t_w)},\\
&C_3^* = m \cdot \hat{e}(f_1^{bdx}(f_1^d)^{s'}, u_2^{\mathsf{ID}} h_2)^\alpha.
\end{aligned}
$$

Denote $\hat{C} = (C^*_{1,1}, C^*_{1,2}, C^*_{1,3}, C^*_{2,1}, C^*_{2,2}, C^*_{2,3}, C^*_3)$. and $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(\hat{C})$. The challenge ciphertext is $(\hat{C}, \sigma^*, \mathsf{vk}^*)$. It is a semi-functional ciphertext with $z_c = a \cdot \mathsf{vk}^* + b$.

If $\mathcal{B}$ attempts to test if the key k is semi-functional by creating a semi-functional ciphertext for ID_i and trying to decrypt, then decryption will work no matter key k is semi-functional or not, since $z_c = A \cdot \mathsf{vk}^* + B$, $z_k = B$ and $z'_k = A$. It means that $\mathcal{B}$ can only create a nominally semi-functional key k. Therefore if $\mathcal{A}$ can distinguish between Game_{k-1} and Game_k, then $\mathcal{B}$ can break Assumption 2'. □

Lemma 11. *If there exists an adversary $\mathcal{A}$ such that $Adv_{\mathcal{A}}(Game_{q_k})$ - $Adv_{\mathcal{A}}(Game_{final})$ $= \epsilon$, then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3'.*

Proof. Given $(f_1, f_1^a, f_1^b, f_1^c, f_2^a, f_2^b, f_2^c, T)$, $\mathcal{B}$ can simulate Game_{q_k} or Game_{final} with $\mathcal{A}$. $\mathcal{B}$ chooses random $t_g, t_v, t_w, t_x, t_y, t_u, t_h \in \mathbb{Z}_p$ and sets

$$
\begin{aligned}
&g_1 = f_1^{t_g}, \quad v_1 = f_1^{t_v}, \quad w_1 = f_1^{t_w}, \quad g_1^a = (f_1^a)^{t_g}, \quad v_1^a = (f_1^a)^{t_v}, \quad w_1^a = (f_1^a)^{t_w},\\
&g_1^\tau = f_1^{t_g t_x}(f_1^a)^{t_y t_g}, \quad v_1^\tau = f_1^{t_v t_x}(f_1^a)^{t_y t_v}, \quad w_1^\tau = f_1^{t_w t_x}(f_1^a)^{t_y t_w},\\
&x_2 = f_2^{t_x}, \quad y_2 = f_2^{t_y}, \quad g_2 = f_2^{t_g}, \quad u_2 = f_2^{t_u}, \quad h_2 = f_2^{t_h}, \quad v_2 = f_2^{t_v}, \quad w_2 = f_2^{t_w},\\
&\hat{e}(g_1, u_2)^\alpha = \hat{e}(f_1^a, f_2^b)^{t_g t_u}, \quad \hat{e}(g_1, h_2)^\alpha = \hat{e}(f_1^a, f_2^b)^{t_g t_h}.
\end{aligned}
$$

It implicitly sets $\tau = t_x + a t_y$ and $\alpha = ab$. $\mathcal{B}$ sends the master public key $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, v_1, w_1, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau, \hat{e}(g_1, u_2)^\alpha, \hat{e}(g_1, h_2)^\alpha)$ to $\mathcal{A}$. The corresponding master secret key is $(\alpha, g_2, u_2, h_2, v_2, w_2, x_2, y_2, f_2)$, where $\alpha = ab$ is not known by $\mathcal{B}$.

For the i-th extraction oracle query with input ID_i, $\mathcal{B}$ chooses random $r_i, k_{i,1}, k_{i,2}$, $k_{i,3}, \gamma_i, z'_k \in \mathbb{Z}_p$, sets $z_k = b(t_u \cdot \mathsf{ID}_i + t_h)/\gamma_i$ and returns the semi-functional key

$$
\begin{aligned}
&D_{1,1} = f_2^{t_w r_i} f_2^{t_x k_{i,1}}, && D_{1,2} = y_2^{k_{i,1}}(f_2^b)^{t_u \cdot \mathsf{ID}_i + t_h}, && D_{1,3} = f_2^{k_{i,1}},\\
&D_{2,1} = g_2^{r_i} x_2^{k_{i,2}}(f_2^a)^{-\gamma_i}, && D_{2,2} = y_2^{k_{i,2}} f_2^{\gamma_i}, && D_{2,3} = f_2^{k_{i,2}},\\
&D_{3,1} = v_2^{r_i} x_2^{k_{i,3}}(f_2^a)^{-\gamma_i z'_k}, && D_{3,2} = y_2^{k_{i,3}} f_2^{\gamma_i z'_k}, && D_{3,3} = f_2^{k_{i,3}}.
\end{aligned}
$$

For each decryption oracle query with input $((C_i, \mathsf{vk}_i, \sigma_i), \mathsf{ID}_i)$, $\mathcal{B}$ extracts the semi-functional key for ID_i as above and then decrypts following the **Decrypt** algorithm. If the ciphertext is normal, then the semi-functional key can decrypt correctly. It is indistinguishable from the decryption oracle using the normal key[14].

In the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages M_0^*, M_1^*, and an identity ID^*. $\mathcal{B}$ chooses a random bit $b' \in \{0, 1\}$, a random $\delta' \in \mathbb{Z}_p$ and runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{B}$

[14] In [139], their IND-ID-CPA proof does not have this additional restriction, since they do not have the decryption oracle.

calculates:

$$C^*_{1,1} = (f_1^c)^{t_g}, \quad C^*_{1,2} = f_1^{\delta'}, \quad C^*_{1,3} = (f_1^c)^{-t_g t_x} f_1^{-t_y \delta'},$$
$$C^*_{2,1} = (f_1^c)^{t_v \cdot \mathsf{vk}^* + t_w}, \quad C^*_{2,2} = f_1^{\delta'(t_v \cdot \mathsf{vk}^* + t_w)/t_g}, \quad C^*_{2,3} = (f_1^c)^{-\delta' t_x (t_v \cdot \mathsf{vk}^* + t_w)} f_1^{-t_y \delta'(t_v \cdot \mathsf{vk}^* + t_w)/t_g},$$
$$C^*_3 = M^*_{b'} \cdot \hat{e}(T, u_2^{\mathsf{ID}^*} h_2)$$

This sets $s = c$, $\delta = \delta' - act_g$, $z_c = (t_v \cdot \mathsf{vk}^* + t_w)/t_g$.

If $T = g^{abc}$, then this is a proper semi-functional ciphertext with message M^*_b and hence $\mathcal{B}$ simulates Game_{q_k}. If T is a random element in $\mathbb{G}_T$, then this is a semi-functional ciphertext with a random message and hence $\mathcal{B}$ simulates Game_{final}.

Notice that $\mathcal{A}$ cannot query decryption oracle query with input $((C_i, \mathsf{vk}_i, \sigma_i), \mathsf{ID}_i)$, where $\mathsf{vk}_i = \mathsf{vk}^*$ and $\mathsf{ID}_i = \mathsf{ID}^*$. Otherwise, C_i will be the forgery of the one time signature for the key vk^*. Therefore, the semi-functional challenge ciphertext cannot be queried for the identity ID^* to the decryption oracle. Therefore if $\mathcal{A}$ can distinguish between Game_{q_k} and Game_{final}, then $\mathcal{B}$ can break Assumption 3'. □

Finally in Game_{final}, the value of b is information theoretically hidden from $\mathcal{A}$. Hence $\mathcal{A}$ has no advantage in winning Game_{final}. If Assumption 1', Assumption 2' and Assumption 3' hold, then Game_{real} is indistinguishable from the Game_{final}. Hence the attacker has negligible advantage in winning Game_{real}.

For the success probability of $\mathcal{B}$, $\mathcal{B}$ has to guess from which point $\mathcal{A}$ can distinguish the games: Game_0, Game_1,...,Game_{q_k} or Game_{final}. $\mathcal{B}$ can only first take a guess and setup the game accordingly. $\mathcal{B}$ guess correctly with probability at least $1/(q_k + 2)$. □

Theorem 27. *Our IBE scheme (ϵ, t, q_k, q_d)-ANON-ID-CCA secure if (ϵ', t')-Assumption 1', (ϵ', t')-Assumption 2' and (ϵ', t')-Assumption 3' hold, where*

$$\epsilon' \geq \frac{\epsilon}{q_k + 2}, \quad t' = t + O(q_k \rho + q_d(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}_2$ and a pairing operation respectively.

Proof. The proof is almost the same as the proof of the IND-ID-CCA security. In the challenge phase of the IND-ID-CCA proof, the adversary returns a challenge identity ID^* and two challenge messages m^*_0, m^*_1. The simulator $\mathcal{B}$ computes the challenge ciphertext using ID^* and $m^*_{b'}$, where $b' \in_R \{0, 1\}$. Now in the challenge phase of the ANON-ID-CCA proof, the adversary returns a challenge message m^* and two challenge identities $\mathsf{ID}^*_0, \mathsf{ID}^*_1$. The simulator $\mathcal{B}$ and two challenge identities $\mathsf{ID}^*_0, \mathsf{ID}^*_1$. The simulator

$\mathcal{B}$ computes the challenge ciphertext using $\mathsf{ID}^*_{b'}$ and m^*, where $b' \in_R \{0,1\}$. It will not change the distribution of the challenge ciphertext in both the proof of lemma 10 and lemma 11. Therefore if Assumption 1', Assumption 2' and Assumption 3' hold, then Game_{real} is indistinguishable from the Game_{final}. Hence the attacker has negligible advantage in winning Game_{real}. □

Theorem 28. *Our IBE scheme (ϵ, t, q_e)-ACI-PKG secure if (ϵ, t')-Assumption 4' holds, where*

$$t' = t + O(q_e(\rho + \tau)),$$

and ρ and τ are the time for an exponentiation in $\mathbb{G}_1$ and an exponentiation in $\mathbb{G}_T$ respectively.

Proof. Suppose there is an adversary $\mathcal{A}$ who breaks the ACI-PKG security. We construct an algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the subgroup decisional bilinear problem with non-negligible advantage. $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ as follows:

Setup: The adversary $\mathcal{A}$ gives the master public key $\mathsf{mpk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, v_1, w_1, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau, \hat{e}(g_1, u_2)^\alpha, \hat{e}(g_1, h_2)^\alpha)$ to $\mathcal{B}$. $\mathcal{B}$ aborts if the mpk is not valid. $\mathcal{B}$ gives $((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}), g_1, v, w, g_1^a, v_1^a, w_1^a, g_1^\tau, v_1^\tau, w_1^\tau)$ to the challenger of the subgroup decisional bilinear problem instance. After that, $\mathcal{B}$ obtains $g_1^s, v^s, w^s, g_1^{as}, v_1^{as}, w_1^{as}, g_1^{\tau s}, v_1^{\tau s}, w_1^{\tau s} \in \mathbb{G}_1$ and $\hat{e}(g_1, g_2)^x, T \in \mathbb{G}_T$ from the problem instance.

Encryption Oracle: On input a message m_i in the message space $\mathcal{M}$, $\mathcal{B}$ runs $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, picks a random $s_i \in \mathbb{Z}_p$ and calculates:

$$\begin{aligned}
&C_{1,1} = g_1^{s_i}, && C_{1,2} = g_1^{as_i}, && C_{1,3} = g_1^{-\tau s_i},\\
&C_{2,1} = (v_1^{\mathsf{vk}_i} w_1)^{s_i}, && C_{2,2} = (v_1^{\mathsf{vk}_i} w_1)^{as_i}, && C_{2,3} = (v_1^{\mathsf{vk}_i} w_1)^{-\tau s_i},\\
&C_3 = m_i \cdot (\hat{e}(g_1, g_2)^x)^{s_i}.
\end{aligned}$$

Denote $\hat{C}_i = (C_{1,1}, C_{1,2}, C_{1,3}, C_{2,1}, C_{2,2}, C_{2,3}, C_3)$. $\mathcal{B}$ signs $\sigma_i \leftarrow \mathsf{Sign}_{\mathsf{sk}_i}(\hat{C}_i)$ and returns the ciphertext $(\hat{C}_i, \sigma_i, \mathsf{vk}_i)$.

Notice that this encryption oracle output implicitly sets the challenge identity ID^* such that $\hat{e}(g_1^\alpha, u_2^{\mathsf{ID}^*} h_2) = \hat{e}(g_1, g_2)^x$. It implies that ID^* is not known to both $\mathcal{A}$ and $\mathcal{B}$.

Challenge: Eventually, $\mathcal{A}$ will output two messages (m_0^*, m_1^*). $\mathcal{B}$ runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow$

Table 5.5: Review of the security of some IBE schemes according to our security model. The notation " * " means that the scheme is secure in a weaker security model only.

Schemes	IND-ID-CCA	ANON-ID-CCA	ACI-PKG	ANON-PKG	Model
BF-IBE [34]	√	√	√	√	RO
SK-IBE [184]	√	√	√	√	RO
BB_1-IBE [27]	*	×	×	×	standard
Waters-IBE [206]	√	×	×	×	standard
BW-IBE [45]	√	√	×	×	standard
Gentry-IBE [92]	√	√	*	×	standard
Waters09-IBE [208]	√	×	×	×	standard
LW-IBE [139]	√	×	×	×	standard
Our 1st construction	√	√	√	√	standard
Our 2nd construction	√	√	√	√	standard

$\mathsf{Gen}(1^\lambda)$, picks a random bit b' and uses the problem instance to calculate:

$$C^*_{1,1} = g_1^s, \qquad C^*_{1,2} = g_1^{as}, \qquad C^*_{1,3} = g_1^{-\tau s},$$
$$C^*_{2,1} = (v_1^{\mathsf{vk}^*} w_1)^s, \qquad C^*_{2,2} = (v_1^{\mathsf{vk}^*} w_1)^{as}, \qquad C^*_{2,3} = (v_1^{\mathsf{vk}^*} w_1)^{-\tau s},$$
$$C^*_3 = m^*_{b'} \cdot T.$$

Denote $\hat{C}^* = (C^*_{1,1}, C^*_{1,2}, C^*_{1,3}, C^*_{2,1}, C^*_{2,2}, C^*_{2,3}, C^*_3)$. $\mathcal{B}$ signs $\sigma^* \leftarrow \mathsf{Sign}_{\mathsf{sk}^*}(\hat{C}^*)$ and returns the challenge ciphertext $(\hat{C}^*, \sigma^*, \mathsf{vk}^*)$.

Guess: Finally $\mathcal{A}$ outputs its guess $b^* \in \{0,1\}$. If $b^* = b'$, $\mathcal{B}$ outputs $T = \hat{e}(g_1, g_2)^{xs}$; otherwise, it outputs $T \in \mathbb{G}_T$. Therefore if Assumption 4' holds, then the adversary has negligible advantage in winning the ACI-PKG game. □

Theorem 29. *Our IBE scheme is information theoretically secure against the ANON-PKG attack.*

Proof. In the challenge ciphertext $(\hat{C}^* = (C^*_{1,1}, C^*_{1,2}, C^*_{1,3}, C^*_{2,1}, C^*_{2,2}, C^*_{2,3}, C^*_3), \sigma^*, \mathsf{vk}^*)$, $C^*_{1,1}, C^*_{1,2}, C^*_{1,3}, C^*_{2,1}, C^*_{2,2}, C^*_{2,3}$, σ^* and vk^* does not contain any information about the user identity. On the other hand, C^*_3 appears random to the adversary $\mathcal{A}$ since it contains the challenge message m^* which is unknown to $\mathcal{A}$. □

5.4.5 Comparison

We first compare our IBE schemes with the existing IBE schemes in Table 5.5, according to the security model for fully anonymous IBE. Most of the existing practical IBE schemes are classified into three frameworks [44]: Full-domain-hash IBE, exponent-inversion IBE and commutative-blinding IBE.

Full-domain-hash IBE. Boneh and Franklin [34] proposed the first practical IBE scheme (BF-IBE). For full-domain-hash IBE, the identity is hashed directly into a bilinear group.

- **BF-IBE.** Boneh and Franklin proved the IND-ID-CCA security of the BF-IBE under the bilinear Diffie-Hellman (BDH) assumption in the random oracle model (ROM). The CCA security proof is further revised by Galindo [88] and then by Nishioka [164]. Abdalla *et al.* [1] proved that the simplified version (BasicIdent) of the BF-IBE scheme is ANON-ID-CPA secure under the BDH assumption in the ROM. It is easy to extend the proof to CCA security for the (full version) BF-IBE scheme. Chow [65] proved that the BF-IBE is ACI-PKG secure under the computational bilinear assumption in the ROM. We are able to prove that BF-IBE is ANON-PKG information theoretically secure.

Exponent-inversion IBE. Sakai and Kasahara [184] gives the first exponent-inversion IBE scheme (SK-IBE). For exponent-inversion IBE, the master public key is g and g^{α}. The identity-based secret key of a user ID is $sk_{\mathsf{ID}} = g^{1/f(\alpha,\mathsf{ID})}$, where f is a function such that $g^{f(\alpha,\mathsf{ID})}$ is easy to compute without knowing α, but sk_{ID} is not.

- **SK-IBE.** The SK-IBE sets $f(\alpha, \mathsf{ID}) = \alpha + \mathsf{ID}$. The SK-IBE satisfies all indistinguishability and anonymity model due to its simplicity. The IND-ID-CCA security of the SK-IBE is proved by Chen and Cheng [61], under the q-BDH Inversion (q-BDHI) assumption in the ROM. Paterson and Srinivasan [172] proved the ANON-ID-CCA of the SK-IBE in the ROM, contradicting a claim of [43]. We are able to prove that SK-IBE is ANON-PKG information theoretically secure.

- **Gentry-IBE.** Gentry [92] extends the key system of [184] to achieve a tight security proof without random oracles. Gentry-IBE is IND-ID-CCA secure under the "decisional truncated augmented BDH exponent" assumption. Chow [65] proved that the Gentry-IBE is ACI-PKG secure under the modified decisional BDH assumption. However, we can show that Gentry-IBE is not ANON-PKG secure. In Gentry-IBE, the master secret key is α and the ciphertext includes $C_1 = g^{s(\alpha+H(\mathsf{ID}))}$ and $C_2 = \hat{e}(g, g_1)^s$. Therefore, the malicious PKG can check if $\hat{e}(C_1, g_1) = C_2^{\alpha+\mathsf{ID}}$.

Commutative-blinding IBE. The first IBE system proposed in [27] by Boneh and Boyen (BB_1-IBE) has a different key structure than the previous IBE schemes. The basic idea is to create, from two or more secret coefficients, two blinding factors that commute with each other under the pairing.

- **BB_1-IBE.** BB_1-IBE is secure under the decisional BDH assumption in the IND-sID-CCA model. Waters [206] proposed a IND-ID-CCA secure construction under the same assumption. Most of the schemes in this family does not achieve the ANON-ID-CCA, ACI-PKG and ANON-PKG security.

- **BW-IBE.** Boyen and Waters [45] proposed the first anonymous IBE without random oracles (BW-IBE). It is IND-ID-CCA secure under the decisional BDH assumption. It satisfies the ANON-ID-CCA model under the decision linear assumption. It is the first recipient anonymous IBE in the commutative-blinding IBE framework.

Recently, Waters [208] used dual system encryption to construct the Waters09-IBE. Lewko and Waters [139] also used dual system encryption to construct the LW-IBE. However, they do not provide recipient anonymity, anonymous ciphertext indistinguishability and anonymity against the PKG. We observe that all existing schemes that have anonymity against the PKG are only provably secure in the random oracle model.

Detailed Comparison with the Waters-IBE and the Gentry-IBE

We give a more detailed comparison for our IBE schemes, the Waters-IBE [206] and the Gentry-IBE [92]. Denote q as the number of key extraction oracle query.

- Waters-IBE:
  - Pros: The security is reduced to the standard DBDH problem.
  - Cons: The public parameters size is proportional to n, where n is inversely proportional to the probability of solving the DBDH problem. The security reduction is loose[15].

- Gentry-IBE:
  - Pros: It has short public parameters. The security reduction is tight.
  - Cons: The security relies on the $q+1$-ABDHE problem. Cheon [64] analysised the security of intractability problems with input size q. Cheon suggested to use a prime p with a larger bit-size in order to achieve the same symmetric security level.

[15] Waters' original proof has a reduction loss of $\frac{\epsilon}{32q(n+1)}$. Recently, Bellare and Ristenpart [20] gives a simplified proof which has a reduction loss of $\frac{\epsilon^2}{27qn+3\epsilon}$.

Table 5.6: Comparison of the loss due to the number of key extraction oracle query q. The reduction loss is the probability of solving the intractability problem if there exists an IND-ID-CPA adversary with advantage ϵ.

	Public Parameters Size	Assumption	Reduction Loss
Waters-IBE	$O(n)$	DBDH	$\frac{\epsilon}{32q(n+1)}$
Gentry-IBE	$O(1)$	$q+1$-ABDHE	$\epsilon - \frac{2}{p}$
Our first IBE	$O(q)$	decisional BDHI	ϵ
Our second IBE	$O(1)$	Assumption 1, 2, 3	$\frac{\epsilon}{q+3}$

- Our first IBE:
  - Pros: The security is reduced to the decisional BDHI problem, which is equivalent to the DBDH problem. The security reduction is tight. It satisfies all security models of fully anonymous IBE.
  - Cons: The public parameters size are proportional to n, where $q \leq (n+1)/2$. The number of multiplication in extraction and encryption are also proportional to q.
- Our second IBE:
  - Pros: It has $O(1)$ public parameters and the security is reduced to static intractability assumption (which is independent of q).
  - Cons: The security relies on new intractability assumptions. The ciphertext size and the key size are large no matter we use composite order groups or prime order groups. The security reduction is loose.

Notice that both the Waters-IBE, the Gentry-IBE and our IBE schemes have some loss related to the number of key extraction oracle query q. In the Waters-IBE and our second IBE, the reduction loss in the security proof is proportional to q. The intractability assumption of the Gentry-IBE is the $q+1$-ABDHE assumption, which has a problem instance of size $q+1$. In our first IBE, the public parameters size is proportional to q. We summarise the loss related to q in Table 5.4.5.

The main disadvantage of our first IBE scheme is that the size of the public parameters is proportional to the number of identity-based secret keys issued by the PKG. Nevertheless, it does not prohibit the practicality of the scheme as justified as follows. We observe that in broadcast encryption, the size of the public parameters is proportional to the maximum number of user secret key (N) issued by the PKG

[35, 40, 41], or the maximum size of a broadcast recipient group (ℓ) [94]. The state of the art broadcast encryption schemes in terms of the size of the public parameters are [40, 41], having $O(\sqrt{N})$ size. If we only consider broadcast encryption schemes with $O(1)$ size ciphertext (as our IBE scheme), their size of the public parameters are either $O(N)$ [35] or $O(\ell)$ [94]. In practical applications, Boneh *et al.* [35] suggested that the public parameters are stored on a file system. Even for a large organisation of 100,000 users this file is only 4MB long (using a standard security parameter where each group element is 20 bytes). The same argument can be applied to our first IBE scheme.

In this Section, we give two IBE schemes which are fully anonymous even against the PKG in the standard model. Each of them has its own advantage to use. Our proof technique in the first IBE scheme gives an interesting alternative to the existing IBE schemes. The existing IBE schemes either have a security reduction loss of q (e.g. BF-IBE, Waters-IBE, Waters09-IBE[16]) or rely on an intractability assumption of size q (e.g. SK-IBE, Gentry-IBE). We show that we can have an IBE scheme which has a tight security reduction and relies on a standard intractability assumption, with the price of having the public parameters size proportional to q.

5.5 Black-Box Accountable Authority IBE

After introducing the notion of *fully anonymous IBE* as a preventive measure to the key escrow problem, we review and improve the existing *Black-Box Accountable Authority IBE*, which belongs to the category of the blaming mechanism of the key escrow problem defined in §5.1.

In 2007, Goyal [101] proposed a new cryptosystem called the Accountable Authority IBE (A-IBE). In A-IBE, the identity-based secret key, which is generated by the PKG interacting with the user, is different every time with a high probability. Therefore, the PKG can be caught when there exist two different identity-based secret keys for the same identity. This is called as the *white-box* A-IBE model. Goyal [101] gave an efficient white-box A-IBE construction. Au *et al.* [8] further proposed an extension that the secret key of the PKG can even be extracted.

Goyal [101] also proposed the *black-box* A-IBE model, which aims to 'blame' a malicious PKG selling a decoder box. In a black-box A-IBE scheme, every pirate decoder box can be traced back to its source. However, Goyal [101] can only construct a *weak black-box* A-IBE scheme, such that traceability is only guaranteed against malicious

[16]The Waters09-IBE has O(1) size public parameters. The IND-ID-CPA security relies on the decisional linear assumption and the DBDH assumption, with a reduction loss of ϵ/q.

PKGs that have no decryption oracle in the attack game. Furthermore, Goyal's weak black-box A-IBE is inefficient as the decryption cost and the ciphertext size are linear in the security parameter. In 2008, Goyal *et al.* [102] proposed the first black-box A-IBE scheme, which allows decryption oracle in the attack game. However, this scheme is also inefficient as the decryption cost and the ciphertext size are linear in the security parameter. In 2009, Libert and Vergnaud [143] proposed a weak black-box A-IBE with short ciphertexts and private keys.

In this section, we first strengthen the security model of the black-box A-IBE, namely the DishonestUser game and the DishonestPKG game. Secondly, we point out a drawback of the weak black-box A-IBE scheme proposed by Libert and Vergnaud [143]. We propose a new security model called "tracing ciphertext indistinguishability" to capture this attack. Furthermore, we construct a new black-box A-IBE scheme which is proven secure in our new model.

Goyal *et al.* [102] used attribute-based encryption (ABE) and oblivious transfer to construct a black-box A-IBE scheme. We observe that the black-box A-IBE scheme proposed by Goyal *et al.* [102] does not achieve perfect correctness. It has infrequent decryption error. Dwork *et al.* [76] provided several methods for transforming an encryption scheme with decryption error into one that is immune to these errors. It can be applied to encryption schemes with infrequent decryption error or more frequent decryption error. If we relax the requirement of the probability of decryption correctness, we can use hierarchical identity-based encryption (HIBE) and oblivious transfer to construct a black-box A-IBE scheme. The advantage of using HIBE over ABE to construct black-box A-IBE is that there are more HIBE schemes proven secure in stronger models (e.g. adaptive-ID model) or having nice properties (e.g. constant size ciphertext). To the best of the authors' knowledge, most ABE schemes in the literature are only selective-set secure (except the recent scheme by Lewko *et al.* [138]) and have the decryption cost and the ciphertext size linear in the security parameter. In this section, we use the HIBE scheme by Boneh *et al.* [30] to construct a black-box A-IBE scheme. It is the *first* black-box A-IBE scheme with constant size ciphertext, when we relax the requirement of decryption correctness.

5.5.1 Security Model for Black-Box Accountable Authority IBE

A Black-box Accountable Authority IBE (A-IBE) scheme has five polynomial-time algorithms, namely **Setup**, **Extract**, **Enc**, **Dec**, **Trace**. These definitions are adapted

from Goyal *et al.* [102] with improvements to the black-box tracing.

- **Setup**(1^λ): On input a security parameter 1^λ, it generates a master secret key msk and some public parameters mpk.
- **Extract**: This is an efficient two-party protocol $(\mathbf{Extract}_p, \mathbf{Extract}_u)$ between the PKG and the user. The common input are mpk and an identity ID. The PKG's algorithm $\mathbf{Extract}_p$ private input is msk. Additionally, the PKG and the user may use a sequence of random coin tosses as private inputs. At the end of the protocol, the user obtains the identity-based secret key sk_{ID} or outputs $\perp$ if the secret key cannot pass a subroutine called **key sanity check**$(\mathsf{mpk}, sk_{\mathsf{ID}})$.
- **Enc**$(\mathsf{mpk}, \mathsf{ID}, m)$: On input mpk, ID and a message m, it outputs a ciphertext C.
- **Dec**$(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, C)$: On input mpk, ID, sk_{ID}, and C, it outputs a message m or outputs $\perp$ if it is invalid.
- **Trace**(ID, sk_{ID}, D): On input an identity ID, a well-formed identity-based secret key sk_{ID} (where "well-formed" means that it can pass the $\mathsf{key\ sanity\ check}$), and a black-box access to a ϵ_D-useful decoder box D, i.e.

  $$\Pr[\mathsf{D}(\mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) = m] > \epsilon_D,$$

  it outputs:

  - PKG if it thinks that the decoder box D is made by the PKG;
  - User if it thinks that the decoder box D is made by the user ID;
  - Fail if it cannot determine who makes the decoder box D.

  The algorithm runs in time polynomial in k and $1/\epsilon$.

Correctness. We define the *correctness* as:

$$\mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) = m,$$

where sk_{ID} is the output of $\mathbf{Extract}_u(\mathsf{mpk}, \mathsf{ID})$ interacting with $\mathbf{Extract}_p(\mathsf{mpk}, \mathsf{ID}, \mathsf{msk})$, and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$.

Security Model for Confidentiality

The security model for confidentiality captures the attack that intends to distinguish which message is encrypted in the ciphertext. We have the following IND-ID-CCA game for confidentiality:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives mpk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to query the oracle adaptively:
   - Key Extraction Oracle $\mathcal{KEO}(\mathsf{ID})$: On input an identity ID, $\mathcal{A}$ runs the $\mathbf{Extract}_u$ protocol to query the oracle. Finally the oracle returns the output of the $\mathbf{Extract}_p$ protocol.
   - Decryption Oracle $\mathcal{DO}(C, \mathsf{ID})$: On input a ciphertext C and an identity ID, it returns a message $m/\perp \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, C)$.
3. $\mathcal{A}$ sends two messages m_0, m_1 and an identity ID^* to the challenger. The challenger picks a random bit b' and computes $C^* \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m_{b'})$. The challenge ciphertext C^* is sent to $\mathcal{A}$.
4. $\mathcal{A}$ is allowed to query the above oracles adaptively.
5. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. We require that there was no query that $\mathcal{KEO}(\mathsf{ID}^*)$ and $\mathcal{DO}(C^*, \mathsf{ID}^*)$. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game minus 1/2. An A-IBE scheme is (ϵ, t, q_k, q_d)-IND-ID-CCA secure if there is no t time adversary has advantage at least ϵ with q_k and q_d queries to $\mathcal{KEO}$ and $\mathcal{DO}$, respectively.

Security Model for Dishonest PKG

The following security model captures the attack that a *malicious PKG* intends to output a ϵ_D-useful decoder box which cannot be traced to itself. We have the following $\mathsf{DishonestPKG}$ game:

1. The adversary $\mathcal{A}$ gives mpk to the challenger. The challenger aborts if mpk is not valid. Otherwise, the challenger maintains two lists L_1 and L_2, both of them are initialised to null.
2. $\mathcal{A}$ is allowed to query the oracles adaptively:

- Key Extraction Oracle $\mathcal{KEO}_u(\mathsf{ID})$: $\mathcal{A}$ runs the $\mathbf{Extract}_p$ protocol and the challenger acts as the $\mathbf{Extract}_u$ protocol. Every input ID to the $\mathcal{KEO}_u$ must be distinct. Finally the challenger gets an identity-based secret key sk_{ID} (or if $\mathbf{Extract}_u$ outputs $\perp$, then the challenger aborts). The challenger stores $(\mathsf{ID}, sk_{\mathsf{ID}})$ in the list L_1.
- Corruption Oracle $\mathcal{CO}(\mathsf{ID})$: On input an identity ID, the challenger checks if $(\mathsf{ID}, sk_{\mathsf{ID}}) \in L_1$. If it is true, the challenger returns sk_{ID}. The challenger puts ID in the list $\mathcal{L}_2$. Otherwise, the challenger returns $\perp$.
- Decryption Oracle $\mathcal{DO}(C, \mathsf{ID})$: On input a ciphertext C and an identity ID, where $(\mathsf{ID}, \cdot) \in L_1$, it returns the decrypted message $m/\perp \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, C)$.

3. $\mathcal{A}$ returns an identity ID^* and a decoder box D^*.

$\mathcal{A}$ wins the game if $(\mathsf{ID}^*, sk_{\mathsf{ID}^*}) \in L_1$ and $\mathsf{ID}^* \notin L_2$, the decoder box D^* is ϵ_D-useful for ID^*, and $\mathbf{Trace}(\mathsf{ID}^*, sk_{\mathsf{ID}^*}, \mathsf{D}^*, \epsilon_D) = \mathsf{User}$. A black-box A-IBE scheme is (ϵ, t, q_k, q_c, q_d)-DishonestPKG-ID-CCA secure if there is no t time adversary winning the above game with probability at least ϵ with q_k, q_c and q_d queries to $\mathcal{KEO}_u$, $\mathcal{CO}$ and $\mathcal{DO}$, respectively.

Enhancement to the previous model. In Goyal *et al.*'s [102] model, the adversary $\mathcal{A}$ gives the challenge identity ID^* to the challenger during the first step of the DishonestPKG game. $\mathcal{A}$ is only allowed to query $\mathcal{KEO}_u$ once with input ID^*, and there is no corruption oracle. The $\mathcal{KEO}_u$ and the $\mathcal{CO}$ are proposed by Au *et al.* [8] for white-box A-IBE [17]. Therefore, our current model is the strongest black-box A-IBE model in the literature for the DishonestPKG game.

Security Model for Dishonest User

The following security model captures the attack that a *malicious user* intends to output a ϵ_D-useful decoder box which can be used to frame an honest PKG. We have the following DishonestUser game:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives mpk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to query the key extraction oracle and the decryption oracle (as defined in the IND-ID-CCA game) adaptively. However every input ID to the key extraction oracle must be distinct.

[17] White-box A-IBE will be discussed in section 5.5.1

3. $\mathcal{A}$ returns a decoder box D^*, an identity-based secret keys sk^*_{ID} and an identity ID^*.

$\mathcal{A}$ wins the game if the decoder box is ϵ_D-useful for ID^*, the **key sanity check**(mpk, sk^*_{ID}) does not output $\perp$, and **Trace**(ID^*, sk^*_{ID}, D^*, ϵ_D) = PKG. The advantage of $\mathcal{A}$ is the probability of $\mathcal{A}$ winning the above game. A black-box A-IBE scheme is (ϵ, t, q_k, q_d)-DishonestUser-ID-CCA secure if there is no t time adversary winning the above game with probability at least ϵ with q_k and q_d queries to $\mathcal{KEO}$ and $\mathcal{DO}$, respectively.

Enhancement to the previous model. Goyal *et al.* [102] only defined the DishonestUser game without the decryption oracle. We emphasise that this oracle potentially helps the adversary to construct a valid decoder box. Notice that the functionality of a decryption oracle is very similar to a black-box access to a decoder box. Therefore, the $\mathcal{DO}(\cdot, \mathsf{ID}^*)$ query can be viewed as a *sample decoder box* (which gives q_d decryption only) to $\mathcal{A}$. Finally $\mathcal{A}$ outputs a decoder box D^* that can decrypt with probability greater than ϵ_D. Therefore, our current model is the strongest black-box accountable authority IBE model in the literature for the DishonestUser game.

Variations of Security Models

The security models for different A-IBE schemes may have some differences with each others. We list some of the commonly used models below.

CPA Model. The chosen ciphertext attack (CCA) model can be changed to a weaker chosen plaintext attack (CPA) model by forbidding the decryption oracle queries. It applies to the security models with decryption oracle queries.

Selective Identity Model. The selective identity (sID) model is a weaker model than the adaptive identity (ID) model. The challenge identity ID^* is given to the challenger during step 1 of the game. Examples can be found in the IND-sID-CCA model of [27]. It applies to the security models where the challenge identity is chosen by the adversary.

Weak Black-box and White-box Accountability Model. The weak black-box accountability model [101] does not allow decryption oracle queries in the DishonestPKG and DishonestUser game. We can view this as the CPA model for accountability. For white-box accountability [101], the **Trace** algorithm will take an identity and an identity-based secret key as inputs. It outputs a decryption key family number n_F or outputs $\perp$ if sk_{ID} is not a valid key. The output of the DishonestPKG game is

changed to an identity-based secret key which is different from the previous key extraction oracle output, but both of them are traced to the same decryption key family number. The outputs of the DishonestUser game are two identity-based secret keys for the same identity, but with different decryption key family numbers.

5.5.2 Indistinguishability for Tracing Ciphertext

In the **Trace** algorithm of the existing A-IBE schemes, we feed a malformed ciphertext (with respect to the identity-based secret key of an honest user) to a decoder box. This (malformed) tracing ciphertext must be indistinguishable from the valid, normal ciphertext, with respect to all users with identity-based secret keys. Otherwise, the user-generated decoder box may just simply outputs $\perp$ to all malformed tracing ciphertexts and causes the **Trace** algorithm to fail.

However, should the malformed ciphertext be indistinguishable from the valid ciphertext, with respect to a dishonest PKG? This question may have different answers for different situations. Suppose only the dishonest PKG can distinguish the malformed ciphertext but not the dishonest user.

- From the security model point of view, the decoder box constructed by the dishonest PKG outputs $\perp$ to all malformed ciphertexts, while the decoder box constructed by any dishonest user outputs some message m'. Therefore, the **Trace** algorithm can still determine whether to blame the PKG or the user.

- In the real world, the dishonest PKG may formulate a decoder box such that it will erase all internal data and will stop further functioning when it is fed with a malformed ciphertext. Therefore, it is not clear if this decoder box can blame the PKG of being dishonest in a court of law. The PKG can defense himself by arguing that the decoder box may have some hardware failure when it is fed with a malformed ciphertext. Unless there are lots of decoder boxes which are available for testing by the **Trace** algorithm, the jury is not satisfied beyond a "reasonable doubt".

A Drawback of Libert and Vergnaud's Weak Black-Box A-IBE

The weak black-box A-IBE scheme proposed by Libert and Vergnaud [143] does not provide indistinguishability for the malformed ciphertexts. We review the related part of their scheme as follows (using our notations):

- **Setup**(1^λ): On input the security parameter 1^λ, it randomly picks generators $g, h, Y,\ Z \in \mathbb{G}$ as part of the mpk. It chooses random $x \in \mathbb{Z}_p$ as msk and calculate $X = g^x$ as part of the mpk.

- **Enc**$(\mathsf{mpk}, m, \mathsf{ID})$: To encrypt a message m given mpk and ID, choose $s \in \mathbb{Z}_p$ and compute the ciphertext $C = (C_1, C_2, C_3, C_4)$, where

$$C_1 = X^s, \quad C_2 = (g^{\mathsf{ID}} Z)^s, \quad C_3 = \hat{e}(g,h)^s, \quad C_4 = m \cdot \hat{e}(g,Y)^s.$$

- **Trace**$(\mathsf{param}, \mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, \mathsf{D}, \epsilon)$: Suppose $sk_{\mathsf{ID}} = (d_1,\ d_2, d_3)$. Conduct the following steps:

  1. Initialise a counter $ctr \rightarrow 0$ and repeat the next steps $L = 16k/\epsilon$ times:
     (a) Choose distinct exponents $s, s' \in \mathbb{Z}_p$ at random, compute $C_1 = X^s$, $C_2 = (g^{\mathsf{ID}} Z)^s$ and $C_3 = \hat{e}(g,h)^{s'}$. Calculate:

$$C_4 = m \cdot \hat{e}(C_1, d_1)/(\hat{e}(C_2, d_2) \cdot C_3^{d_3}),$$

     for a random message $m \in \mathbb{G}_T$.
     (b) Feed the decryption box D with $(C_1, C_2, C_3,\ C_4)$. If D outputs m' such that $m' = m$, increment ctr.
  2. If $ctr < 4k$, incriminate the PKG. Otherwise, incriminate the user.

In the proof of theorem 5 of [143], Libert and Vergnaud proved that even $\Pr[ctr \geq 1]$ is negligible for D generated by the dishonest PKG. By theorem 4 of [143], a decryption box D generated by a dishonest user has negligible $\Pr[ctr < 4k]$. However, we observe that a dishonest PKG can distinguish a malformed tracing ciphertext by checking if $\hat{e}(C_1, h) \stackrel{?}{=} C_3^x$. If it is not equal, the dishonest PKG can formulate a decryption box D' that it will erase all internal data and will stop further functioning. Therefore, the **Trace** algorithm will not run for $L = 16k/\epsilon$ iterations for a single decoder box D. Neither PKG nor User can be outputted by the **Trace** algorithm. According to the security notion, the **Trace** algorithm can only output Fail in this case. As a result, we cannot get the guarantee from the theorem 4 of [143].

We note that this attack is ruled out by the security model of [143]. They assume that "pirate devices are stateless, or resettable, and do not retain information from prior queries: each query is answered as if it were the first one". However, this attack is not ruled out by the original security model of [102]. We think that this attack is

realistic and is likely to occur in the real world. Therefore we propose a new security model for the indistinguishability for tracing ciphertext.

In short, the **Trace** algorithm of the weak black-box A-IBE in [143] works only when there are $L = 16k/\epsilon$ identical decoder boxes. However, we do not know any method to check if two decoder boxes are identical when we only have black-box access to them.

Security Model of Indistinguishability for Tracing Ciphertext

In order to avoid the possible dispute in the court of law (as demonstrated in the previous real world example) and the flaw similar to the weak black-box A-IBE schemes of [143], we propose a new indistinguishability model for tracing ciphertext. We define a IND-Trace game as follows:

1. The adversary $\mathcal{A}$ gives mpk to the challenger. The challenger aborts if mpk is not valid.

2. $\mathcal{A}$ gives an identity ID^* to the challenger. $\mathcal{A}$ runs the **Extract** protocol with the challenger, acting as the PKG. Finally the challenger obtains the secret key sk_{ID^*}.

3. $\mathcal{A}$ gives a message m^* to the challenger. The challenger runs $C_{Enc} \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m^*)$. The challenger also formulates a (malformed) tracing ciphertext C_{Tr} used in **Trace** for sk_{ID^*} and m^*. The challenger randomly picks a bit $b' \in \{0, 1\}$. The challenger returns $C_{b'} = C_{Enc}$ and $C_{1-b'} = C_{Tr}$ to $\mathcal{A}$.

4. $\mathcal{A}$ returns a guess b^* of b'.

$\mathcal{A}$ wins the game if $b' = b^*$. The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game minus half. We say that a black-box A-IBE scheme is (t, ϵ)-IND-Trace secure if no t time adversary wins the above game with non-negligible advantage ϵ.

5.5.3 Another Black-box Accountable Authority IBE

Goyal *et al.* [102] gave the first black-box A-IBE scheme. However, this scheme is far less efficient in terms of public and private key size, decryption cost and ciphertext size, when compared to the white-box constructions ([101, 143]). Goyal *et al.*'s construction used t-out-of-n oblivious transfer during **Extract** and attribute-based encryption during **Enc**. Since the existing attribute-based encryption are only selective-set secure, the resulting black-box A-IBE scheme is only selective-ID secure for the DishonestUser game.

Decryption Correctness

The black-box A-IBE by Goyal *et al.* [102] has a slightly different definition for decryption correctness. The standard definition for decryption correctness is

$$\Pr[\mathbf{Dec}(\mathsf{mpk}, \mathsf{ID}, sk_{\mathsf{ID}}, \mathbf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) = m] = 1.$$

We denote this as *perfect correctness*, similar to the definition for public key encryption in [76].

However for the black-box A-IBE by Goyal *et al.* [102], the probability for decryption correctness is greater than $1 - e^{-2n/225}$ (using the figure in appendix A of [102]), where $|n| = poly(k)$ and **Setup** is given 1^{λ} as input. Furthermore, this probability only applies to the ciphertexts honestly generated from the **Enc** algorithm. It is because during the **Trace** algorithm, a decoder box is fed with some carefully designed ciphertext such that it cannot decrypt correctly with non-negligible probability. Goyal *et al.* [102] did not give the details for eliminating the decryption errors. They only mentioned that their scheme can achieve perfect correctness by running a "complementary scheme" in parallel.

Handling Decryption Error. Some cryptosystems do not achieve perfect correctness as well, such as the Ajtai-Dwork cryptosystem [3] and NTRU [114]. Dwork *et al.* [76] defined an encryption scheme to be α-correct, if the probability of decryption error is at most $1 - \alpha$. Dwork *et al.* provided several methods for transforming an encryption scheme with decryption error into one that is immune to these errors. They can be applied to the black-box A-IBE by Goyal *et al.* [102], which only has infrequent decryption error. Furthermore, Dwork *et al.* also provided a transformation for more frequent decryption error. To improve an α-correct encryption scheme, where α may be as small as $1/poly(k)$, Dwork *et al.* used parallel repetition, hard core bit and direct product for the transformation. It will produce schemes that has perfect correctness for almost all keys in the standard model.

Our Black-box A-IBE Construction

Intuition. If we use the relaxed α-correct definition for black-box A-IBE, we can construct a black-box A-IBE scheme with constant size of ciphertext and provably secure in a stronger security model. We can then use the transformation of Dwork *et al.* [76] to achieve perfect correctness. In contrast with the black-box A-IBE scheme in [102], we propose the use of hierarchical identity-based encryption (HIBE) instead

of attribute-based encryption in [102]. The first level of HIBE is the user identity, while the lower levels are the dummy identities (cf. dummy attributes in [102]). The advantage of our approach is that there are some HIBE schemes which are secure in the adaptive-ID model, with constant size of ciphertext or constant decryption cost.

We now give a black-box A-IBE construction based on the HIBE scheme in [30]:

- **Setup:** On input 1^λ, it generates a λ-bit prime p. Let $\mathbb{G}$ be a cyclic group of order p and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a pairings. It selects the length ℓ_e, ℓ_t, ℓ_n as the system parameters. It picks some generators $g, g_2, g_3, h_0, \ldots, h_{\ell_n} \in \mathbb{G}$. It selects some dummy identities $\mathsf{id}_1, \ldots, \mathsf{id}_{\ell_n} \in \mathbb{Z}_p$. Denote a set S_u as $\{1, \ldots, \ell_n\}$.

  The PKG chooses a random $\alpha \in \mathbb{Z}_p$ and calculates $g_1 = g^\alpha$. The master public key is $\mathsf{mpk} = (g, g_1, g_2, g_3, h_0, \ldots, h_{\ell_n}, \mathsf{id}_1, \ldots, \mathsf{id}_{\ell_n})$ and the master secret key is $\mathsf{msk} = g_2^\alpha$.

- **Extract:** To generate an identity-based secret key for an identity ID, the PKG computes:

$$a_0 = g_2^\alpha (g_3 h_0^{\mathsf{ID}} h_1^{\mathsf{id}_1} \cdots h_{\ell_n}^{\mathsf{id}_{\ell_n}})^r, \quad a_1 = g^r,$$
$$b_1 = h_1^r, \quad \ldots, \quad b_{\ell_n} = h_{\ell_n}^r$$

  The PKG and the user runs a ℓ_t-out-of-ℓ_n oblivious transfer (refer to §3.4.1 for details) for π, where π is a random permutation of the set $\{b_1, \ldots, b_{\ell_n}\}$. Denote the set S_{ID} be the set of indices i such that the user obtains b_i after the oblivious transfer. The PKG also sends (a_0, a_1) to the user.

  Key Sanity Check: The user accepts the identity-based secret key if

$$\hat{e}(g, a_0) = \hat{e}(g_1, g_2) \cdot \hat{e}(a_1, g_3 h_0^{\mathsf{ID}} h_1^{\mathsf{id}_1} \cdots h_{\ell_n}^{\mathsf{id}_{\ell_n}}),$$
$$\hat{e}(a_1, h_i) = \hat{e}(g, b_i) \quad \forall i \in S_{\mathsf{ID}}.$$

  Otherwise, the user outputs $\perp$.

- **Enc:** To encrypt a message $m \in \mathbb{G}_T$ under the public key ID, it randomly picks ℓ_e integers from $\{1, \ldots, \ell_n\}$. Denote the set of integers as S_e. It also selects a random $t \in \mathbb{Z}_p$ and outputs

$$C = \big(m \cdot \hat{e}(g_1, g_2)^t, \quad g^t, \quad (g_3 h_0^{\mathsf{ID}} \prod_{j \in S_e} h_j^{\mathsf{id}_j})^t, \quad S_e\big).$$

- **Dec:** To decrypt a ciphertext $C = (c_1, c_2, c_3, S_e)$, the user ID with secret key $(a_0, a_1, \{b_i\}_{i \in S_{\mathsf{ID}}})$ checks if $S_{\mathsf{ID}} \supseteq S_u \backslash S_e$. He outputs $\perp$ if it is not true. Otherwise, he outputs the message m, where:

$$a'_0 = a_0 \prod_{i \in S_u \backslash S_e} b_i^{-\mathsf{id}_i}, \quad m = c_1 \cdot \hat{e}(a_1, c_3)/\hat{e}(c_2, a'_0).$$

- **Trace:** It takes an identity ID, a well-formed secret key $sk_{\mathsf{ID}} = (a_0, a_1, \{b_i\}_{i \in S_{\mathsf{ID}}})$ and a decoder box D which is ϵ_D-useful. It conducts the following steps:

  1. Initialise a counter $ctr \rightarrow 0$ and repeat the next steps $L = 8k/\epsilon_D$ times:
     (a) Choose a set S_e where $\exists i \in S_u \setminus S_e$ and $i \notin S_{\mathsf{ID}}$.
     (b) Encrypt a random message m using the set S_e as dummy identities and obtain a ciphertext C.
     (c) Feed the decryption box D with the ciphertext C. If D outputs m' such that $m' = m$, increment ctr.
  2. If $ctr < 4k$, incriminate the user. Otherwise, incriminate the PKG.

Instantiation of Parameters. If sk_{ID} uses the set S_t of size ℓ_t as dummy identities, it can decrypt a ciphertext if $\forall i \in S_u \setminus S_e, i \in S_t$. It happens with probability $\epsilon' = \frac{\ell_t}{\ell_n} \cdot \frac{\ell_t - 1}{\ell_n - 1} \cdots \frac{\ell_t - \ell_n + \ell_e + 1}{\ell_e + 1}$ if $\ell_t \geq \ell_n - \ell_e$. For simplicity, we set $\ell_t = \ell_e = \ell_n - 1$. Then we have $\epsilon' = \frac{\ell_n - 1}{\ell_n}$. If ℓ_n is large, sk_{ID} cannot decrypt a valid ciphertext with probability $\frac{1}{\ell_n}$.

Theorem 30. *Our black-box A-IBE scheme is (t', ϵ, q_k)-IND-sID-CPA secure if the (ϵ, t)-$(\ell_n + 1)$-decisional wBDHI assumption holds in $\mathbb{G}$, where $t' < t - \Theta(\tau q_k)$ and τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose the adversary $\mathcal{A}$ has advantage ϵ in attacking the black-box A-IBE scheme. We construct an algorithm B that solves the $(\ell_n + 1)$-Decisional wBDHI problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, let $y_i = g^{\alpha^i} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random instance of the $(\ell_n + 1)$-Decisional wBDHI problem: $(g, h, y_1, \ldots, y_{\ell_n + 1}, T)$. Algorithm $\mathcal{B}$'s goal is to output 1 if $T = \hat{e}(g, h)^{\alpha^{\ell_n + 2}}$ and 0 otherwise. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup. The selective identity game begins with $\mathcal{A}$ first outputting an identity ID^* that it intends to attack. After that, the algorithm $\mathcal{B}$ picks a random $\gamma \in \mathbb{Z}_p$ and sets

$g_1 = y_1 = g^\alpha$ and $g_2 = y_{\ell_n+1} \cdot g^\gamma = g^{\alpha^{\ell_n+1}+\gamma}$. Next, B picks random $\gamma_0, \ldots, \gamma_{\ell_n} \in \mathbb{Z}_p$ and sets $h_i = g^{\gamma_i}/y_{\ell_n-i+1}$ for $i = 0, \ldots, \ell_n$.

$\mathcal{B}$ picks a set of random dummy identities $\{\mathsf{id}_1, \ldots, \mathsf{id}_{\ell_n}\}$. $\mathcal{B}$ randomly picks ℓ_e integers from $S_u = \{1, \ldots, \ell_n\}$. Denote the set of integers as S_e^*. $\mathcal{B}$ also picks a random $\delta \in \mathbb{Z}_p$ and sets $g_3 = g^\delta \cdot y_{\ell_n+1}^{\mathsf{ID}^*} \cdot \prod_{i \in S_e^*} y_{\ell_n-i+1}^{\mathsf{id}_i}$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $\mathsf{mpk} = (g, g_1, g_2, g_3, h_0, \ldots, h_{\ell_n}, \mathsf{id}_1, \ldots, \mathsf{id}_{\ell_n})$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$. The corresponding master secret key is $g_2^\alpha = g^{\alpha(\alpha^{\ell_n+1}+\gamma)} = y_{\ell_n+2} \cdot y_1^\gamma$, which is unknown to $\mathcal{B}$.

Oracles Simulation. $\mathcal{B}$ simulates the key extraction oracle as follows. On input an identity ID, $\mathcal{B}$ first picks an integer $\tilde{r} \in \mathbb{Z}_p$. Denote $r = \frac{\alpha}{\mathsf{ID}-\mathsf{ID}^*} + \tilde{r}$. $\mathcal{B}$ calculates:

$$a_0 = y_{\ell_n+1}^{\tilde{r}(\mathsf{ID}^*-\mathsf{ID})} y_1^\gamma \big(g^{\delta+\gamma_0\mathsf{ID}+\sum_{j=1}^{\ell_n}\gamma_j\mathsf{id}_j} \prod_{i \in S_u \setminus S_e^*} y_{\ell_n-i+1}^{-\mathsf{id}_i}\big)^r, \quad a_1 = g^{\tilde{r}} y_1^{\frac{1}{\mathsf{ID}-\mathsf{ID}^*}},$$

$$b_1 = g^{\tilde{r}\gamma_1} y_1^{\frac{\gamma_1}{\mathsf{ID}-\mathsf{ID}^*}} y_{\ell_n}^{-\tilde{r}} y_{\ell_n+1}^{\frac{-1}{\mathsf{ID}-\mathsf{ID}^*}}, \quad \ldots, \quad b_{\ell_n} = g^{\tilde{r}\gamma_{\ell_n}} y_1^{\frac{\gamma_{\ell_n}}{\mathsf{ID}-\mathsf{ID}^*}} y_1^{-\tilde{r}} y_2^{\frac{-1}{\mathsf{ID}-\mathsf{ID}^*}}.$$

Notice that the secret key $(a_0, a_1, b_1, \ldots, b_{\ell_n})$ can be calculated using y_1 to y_{ℓ_n+1}.

Observe that a_0 is a valid component of the secret key:

$$\begin{aligned}
a_0 &= g_2^\alpha (g_3 h_0^{\mathsf{ID}} h_1^{\mathsf{id}_1} \cdots h_{\ell_n}^{\mathsf{id}_{\ell_n}})^r \\
&= y_{\ell_n+2} \cdot y_1^\gamma \cdot \big(g^\delta \cdot y_{\ell_n+1}^{\mathsf{ID}^*} \cdot \prod_{i \in S_e^*} y_{\ell_n-i+1}^{\mathsf{id}_i} \cdot (\frac{g^{\gamma_0}}{y_{\ell_n+1}})^{\mathsf{ID}} \cdot \prod_{j=1}^{\ell_n} (\frac{g^{\gamma_j}}{y_{\ell_n-j+1}})^{\mathsf{id}_j}\big)^r \\
&= y_{\ell_n+2} \cdot y_1^\gamma \cdot y_{\ell_n+1}^{(\mathsf{ID}^*-\mathsf{ID})r} \cdot \big(g^\delta \cdot \prod_{i \in S_e^*} y_{\ell_n-i+1}^{\mathsf{id}_i} \cdot g^{\gamma_0\mathsf{ID}} \cdot \prod_{j=1}^{\ell_n} (\frac{g^{\gamma_j}}{y_{\ell_n-j+1}})^{\mathsf{id}_j}\big)^r \\
&= y_{\ell_n+1}^{\tilde{r}(\mathsf{ID}^*-\mathsf{ID})} y_1^\gamma \big(g^{\delta+\gamma_0\mathsf{ID}+\sum_{j=1}^{\ell_n}\gamma_j\mathsf{id}_j} \prod_{i \in S_u \setminus S_e^*} y_{\ell_n-i+1}^{-\mathsf{id}_i}\big)^r.
\end{aligned}$$

$\mathcal{B}$ uses this secret key to run the oblivious transfer protocol with the adversary $\mathcal{A}$.

Challenge. At some point $\mathcal{A}$ sends two challenge messages m_0^*, m_1^* to $\mathcal{B}$. $\mathcal{B}$ picks a random bit $b' \in \{0, 1\}$. $\mathcal{B}$ calculates:

$$C^* = \big(m_{b'}^* \cdot T \cdot \hat{e}(y_1, h)^\gamma, \quad h, \quad h^{\delta+\mathsf{ID}^*\gamma_0+\sum_{i \in S_e^*}\mathsf{id}_i\gamma_i}, \quad S_e^*\big),$$

where h and T are from the problem instance given to $\mathcal{B}$. First note that if $h = g^c$ (for some unknown $c \in \mathbb{Z}_p$) then:

$$\begin{aligned}
h^{\delta+\mathsf{ID}^*\gamma_0+\sum_{i \in S_e^*}\mathsf{id}_i\gamma_i} &= \big(g^\delta \cdot (\frac{g^{\gamma_0}}{y_{\ell_n+1}} \cdot y_{\ell_n+1})^{\mathsf{ID}^*} \cdot \prod_{i \in S_e^*} (\frac{g^{\gamma_i}}{y_{\ell_n-i+1}} \cdot y_{\ell_n-i+1})^{\mathsf{id}_i}\big)^c \\
&= (g_3 h_0^{\mathsf{ID}^*} \prod_{i \in S_e^*} h_i^{\mathsf{id}_i})^c,
\end{aligned}$$

$$\begin{aligned}\hat{e}(g,h)^{\alpha^{\ell_n+2}} \cdot \hat{e}(y_1,h)^{\gamma} &= \left(\hat{e}(y_1, y_{\ell_n+1}) \cdot \hat{e}(y_1, g^{\gamma})\right)^{c} \\ &= \hat{e}(y_1, y_{\ell_n+1} g^{\gamma})^{c} \\ &= \hat{e}(g_1, g_2)^{c}.\end{aligned}$$

Therefore, if $T = \hat{e}(g,h)^{\alpha^{\ell_n+1}}$, then C^* is a valid encryption of the message $m^*_{b'}$ under the identity ID^*.

Output. Finally $\mathcal{A}$ outputs a bit b^*. If $b^* = b'$, then $\mathcal{B}$ outputs 1 meaning $T = \hat{e}(g,h)^{\alpha^{\ell_n+1}}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}$. □

Theorem 31. *Our black-box A-IBE scheme is DishonestUser-sID-CPA secure if the decisional wBDHI assumption holds.*

Proof. (Sketch) The proof is similar to the IND-sID-CPA proof. We denote the dummy identities set of chosen by the adversary in sk^*_{ID} as S^*_t. The main difference with the IND-sID-CPA game is that after the last step of the DishonestUser game, the algorithm $\mathcal{B}$ generates a challenge ciphertext C^* using a dummy identities set S^*_e. $\mathcal{B}$ sets S^*_e such that $\exists i \in S_u \setminus S^*_e$ and $i \notin S^*_t$. Then C^* is a valid ciphertext which cannot be decrypted by sk^*_{ID}. If the decoder box D^* can decrypt the challenge ciphertext C^* correctly with probability ϵ_D, then $\mathcal{B}$ can answer the decisional wBDHI problem. □

Theorem 32. *Our black-box A-IBE scheme is DishonestPKG-ID-CPA secure if the underlying ℓ_t-out-of-ℓ_n oblivious transfer protocol is secure.*

Proof. The algorithm $\mathcal{B}$ honestly runs the key extraction oracle with the adversary $\mathcal{A}$. If the underlying ℓ_t-out-of-ℓ_n oblivious transfer protocol is secure, then the dishonest PKG $\mathcal{A}$ has negligible probability of correctly guessing the dummy identities chosen by $\mathcal{B}$.

In an iteration of the **Trace** algorithm, the decoder box D^* is given a ciphertext $C = (c_1, c_2, c_3, c_4, S_e)$, which cannot be decrypted by the specific secret key sk^*_{ID}. Notice that the dummy identity set S_{ID^*} corresponding to sk^*_{ID} is unknown to $\mathcal{A}$. By definition, if the decoder box is ϵ_D-useful, the ciphertext C can be decrypted with probability ϵ_D. Now ctr can be seen as the sum of $L = 8k/\epsilon_D$ independent random variables $X_i \in \{0,1\}$ having the same expected value ϵ_D. We have $\mu = \mathrm{E}[ctr] = L\epsilon_D$. By the Chernoff bound, for any real number ω such that $0 \leq \omega \leq 1$, $\Pr[ctr < (1-\omega)\mu] < e^{-\mu\omega^2/2}$. We take $\omega = 1/2$. The Chernoff bound guarantees that $\Pr[ctr < 4k] = \Pr[ctr < \mu/2] < e^{-\mu/8} = e^{-k}$. Therefore, $\mathcal{A}$ incriminates the user with negligible probability. □

Theorem 33. *Our black-box A-IBE scheme is IND-Trace secure if the underlying ℓ_t-out-of-ℓ_n oblivious transfer protocol is secure.*

Proof. (Sketch) Our tracing ciphertext and the ciphertext computed from **Enc** are the same, except that the set S_e is chosen according to the sk_{ID^*} for the former case, while S_e is randomly chosen from the latter case. By the security of the ℓ_t-out-of-ℓ_n oblivious transfer, $\mathcal{A}$ cannot know the sk_{ID^*}. Therefore, the set S_e appears to be random to $\mathcal{A}$. □

Extensions

IND-CCA Security. Boneh *et al.* [32] showed how to build an IND-sID-CCA-secure ℓ-HIBE scheme from an IND-sID-CPA-secure $\ell + 1$-HIBE scheme. Applying this method to our construction will give us an IND-sID-CCA-secure black-box A-IBE scheme.

Adaptive ID Security. Boneh and Boyen [27] observed that IBE systems that are selective-ID secure are also secure against adversaries who adaptively select the identity to attack, as long as one hashes the identity prior to using it. Assume $H : \{0,1\}^* \rightarrow \{0,1\}^d$ be a collision resistant hash function (where $d \approx 160$ bits). The reduction introduces a 2^d multiplicative security loss factor in the standard model.

DishonestPKG-CCA Security. Our scheme is not DishonestPKG-ID-CCA secure since the decryption oracle will leak information about the dummy identity set S_{ID^*} when it outputs $\perp$. However, if we use the Dwork *et al.*'s transformation [76], then the decryption oracle only outputs $\perp$ with negligible probability. Therefore, our scheme can be extended to achieve DishonestPKG-ID-CCA security.

Shorter Private Keys. We can modify our scheme such that the private key size grows only sublinearly with ℓ_t (the size of the set S_{ID}), following the idea in [30].

Perfect Correctness. If we allow certain degree of decryption error, our black-box A-IBE will have a constant size of ciphertext. If we want to have a black-box A-IBE with perfect correctness, we can apply the transformation of Dwork *et al.* [76] to our scheme. However, the resulting ciphertext size is proportional to the decryption error.

Comparison

We compare our A-IBE scheme with the existing black-box A-IBE schemes in the literature. We denote scheme 1 as our basic scheme in section 5.5.3. We obtain scheme 2 by applying the transformation of Dwork *et al.* [76] to scheme 1. The results are in Table 5.7.

Table 5.7: Comparison of black-box A-IBE schemes. "Dis." stands for dishonest. * means that perfect correctness can be achieved by applying the transformation of Dwork *et al.* [76]. However, it will increase the size of the ciphertext.

	Confidentiality	Dishonest User	Dishonest PKG
Goyal [101]	IND-ID-CCA	Dis.User-ID-CPA	Dis.PKG-ID-CPA
Goyal *et al.* [102]	IND-ID-CCA	Dis.User-sID-CCA	Dis.PKG-sID-CPA
Libert-Vergnaud [143]	IND-sID-CCA	Dis.User-sID-CPA	Dis.PKG-sID-CPA
Our scheme 1	IND-sID-CCA	Dis.User-sID-CCA	Dis.PKG-sID-CPA
Our scheme 2	IND-sID-CCA	Dis.User-sID-CCA	Dis.PKG-sID-CCA

	Perfect Correctness	IND-Trace	Constant Size Ciphertext
Goyal [101]	$\surd$	$\surd$	$\times$
Goyal *et al.* [102]	*	$\surd$	$\times$
Libert-Vergnaud [143]	$\surd$	$\times$	$\surd$
Our scheme 1	*	$\surd$	$\surd$
Our scheme 2	$\surd$	$\surd$	$\times$

Notice that the scheme by Goyal [101] and the scheme by Libert and Vergnaud [143] are only weak black-box A-IBE only. We give the first black-box A-IBE (scheme 1) with constant size of ciphertext, if we allow certain degree of decryption error. We can remove the decryption error, at the cost of having longer ciphertext (scheme 2).

In this section, we proposed some improvements to the existing security model of the black-box A-IBE. We showed a drawback of the A-IBE scheme in [143]. We proposed a new model "IND-Trace" to capture this attack. We also proposed a new black-box A-IBE scheme. It is the first black-box A-IBE with constant size of ciphertext, if we allow certain degree of decryption error. We suggested a method to remove the decryption error, at the cost of having longer ciphertext.

5.5.4 Summary

In this Chapter, we analyse the key escrow problem in identity-based cryptography. We propose a solution of escrow-free identity-based signatures. We show the impossibility of ideal escrow-free identity-based encryption. To deal with the key escrow problem in identity-based encryption, we propose a new notion of fully anonymous identity-based encryption as a preventive measure, and give a new construction of accountable authority identity-based encryption as a blaming mechanism.

Pairings can be used to construct various identity-based cryptosystems. There are

some "generalised" version of identity-based cryptosystems which can be constructed from pairings, such as the attribute-based encryption [103] and the predicate encryption [127]. It is interesting to construct an even more "generalised" version of identity-based encryption, in which the secret key of the user is generated by a trused third party. In the next Chapter, we will show that such generalisation is possible if we instantiate the new cryptographic primitive *lossy trapdoor functions* in pairings.

Chapter 6

New Cryptographic Primitives and Protocols

Pairings can be used to construct new cryptographic primitives and cryptosystems. In this Chapter, we introduce some new cryptographic protocols and applications that can be constructed using pairings.

Firstly, pairings can be used to give new instantiation of important general primitives in public key cryptography. For example, Nguyen [162] constructed a new accumulator from pairings and proposed some applications for accumulator. Groth and Sahai [106] gave a new non-interactive zero-knowledge proof system using pairings. In §6.1, we propose two new instantiations of lossy trapdoor function using pairings and suggest some new applications for them.

Secondly, pairings can be used to construct new cryptographic primitives, which may be difficult to give an efficient construction without using pairings. For example, Chase and Lysyanskaya [58] introduced the notion of "simulatable verifiable random functions" and gave an efficient construction using pairings. Sahai and Waters [183] proposed a fuzzy identity-based scheme, which can be used for a new application called "attribute-based encryption". Katz *et al.* [127] introduced the notion of "predicate encryption" and gave an efficient construction using pairings. In particular, attribute-based encryption and predicate encryption can be viewed as the generalisation of identity-based encryption, which replace the identity by the attribute or the predicate function respectively. In §6.2, we propose a more generalised version of identity-based encryption, which is called *two-tier encryption*. Two-tier encryption can be constructed from the extension of lossy trapdoor function in §6.1, which we called *two-tier trapdoor function*. We give a concrete construction using pairings and prove its security.

Finally, pairings can be used to construct secure and efficient practical applications. For example, Camenisch *et al.* [49] proposed a new compact e-cash system which uses pairings. It satisfies more security requirements of e-cash system than previous

schemes. In §6.3, we present a cryptographic treatment for publish/subscribe systems and give a concrete construction. Our construction satisfies more security requirements of publish/subscribe system than previous schemes.

6.1 Lossy Trapdoor Functions

Trapdoor functions (TDFs) are an important general primitive in public key cryptography proposed by Diffie and Hellman [71]. The well-known realisations of TDFs are all related to factoring: RSA [180], Rabin [178] and Paillier [171]. In 2008, Peikert and Waters [174] proposed a new primitive called lossy trapdoor functions. Their contribution is twofold. Firstly, the lossy TDFs imply the injective TDFs. They realise lossy TDFs in two ways, one depends on the hardness of the decisional Diffie-Hellman (DDH) problem and the other depends on the worst-case hardness of lattice problems (LWE). This yields the first known TDFs based on number theoretic problems that are not directly related to integer factorisation. They also suggested the use of Paillier's techniques to construct a more efficient variant. Rosen and Segev [182] proposed a similar result by using the Damgård-Jurik encryption.

Secondly, Peikert and Waters [174] presented black-box constructions for different cryptography primitives using lossy TDFs. They constructed collision-resistant hash functions (which are also universal one-way hash functions) from a collection of lossy functions, a weaker primitive whose injective functions do not require to have trapdoors. They also proposed chosen-ciphertext (CCA) secure encryption and oblivious transfer using lossy TDFs and all-but-one (ABO) trapdoor functions, which is a richer abstraction of lossy TDFs.

Our Results In this section, we make the following contributions:

- We realise the lossy TDFs in the DDH-easy groups, including the pairings. We propose two schemes based on the subgroup decision problem and the decision linear problem, respectively. These yield *the first* TDFs based on the pairings. We compare our work and other constructions of lossy TDFs in Table 6.1.

- We present a black-box construction of a tag-based encryption and a universal re-encryption based on lossy TDFs. We compare our work and other applications of lossy TDFs in Table 6.2.

- When our tag-based encryption scheme is instantiated by the LWE-based lossy

Table 6.1: Summary of different constructions of lossy TDFs. * denotes the work in this section.

Lossy TDF	constructed from ⟵	DDH [174] LWE [174] Strong RSA [182] Subgroup Decision * Decision Linear *

Table 6.2: Comparison of the work from Peikert and Waters [174] and this section. * denotes the work in this section.

Lossy TDF	Public Key Encryption [174] Collision-Resistant Hash Functions [174] Oblivious Transfer [174] Tag-Based Encryption * Universal Re-encryption *

TDF in [174], we obtain the first tag-based encryption based on the complexity of a lattice problem.

6.1.1 Lossy Trapdoor Functions from DDH-Easy Groups

Peikert and Waters [174] proposed the realisation of lossy TDFs in DDH-hard groups. However, there exist DDH-easy groups in many cryptographic primitives, like the pairings in identity-based cryptography. In this section, we propose two different constructions which are secure in the DDH-easy groups.

Subgroup Decision

We first review the notion related to the subgroup decision problem introduced by Boneh *et al.* [36]. Consider a generator algorithm $\mathcal{G}$ that, on input a security parameter 1^λ, outputs a tuple $(p, q, \mathbb{G}, g)$, in which p and q are independent uniform random λ-bit primes, $\mathbb{G}$ is a cyclic group of order $N = pq$ and g is a generator of $\mathbb{G}$. Let $\mathbb{G}_q \subset \mathbb{G}$ denotes the subgroup of $\mathbb{G}$ of order q.

Encryption from Subgroup Decision. First we review the BGN encryption scheme from the subgroup decision problem in [36]. The user runs $\mathcal{G}(1^\lambda)$ to generate his key pairs. He picks a generator $h \in_R \mathbb{G}_q$. The secret key is q and the public key is ($N = pq$, g, h, $\mathbb{G}$). To encrypt an $m \in \{0, \ldots, T\}$ with $T < p$, choose an $r \in_R \mathbb{Z}_N$ and create the ciphertext $E_h(m; r) = h^r g^m \in \mathbb{G}$. To decrypt a ciphertext c, output $D_q(c) = \log_g(c^q)$. Since c encrypts a small set of m, the discrete logarithm can be computed by

enumeration. Boneh *et al.* [36] proved that this cryptosystem is semantically secure under the subgroup decision assumption.

Note that the cryptosystem is additively homomorphic in the following way:

$$E_h(m;r) \odot E_h(m';r') = E_h(m+m';r+r'),$$

where $\odot$ denotes coordinate-wise multiplication of ciphertexts. Therefore we also have $E_h(m;r)^x = E_h(mx;\, rx)$ for any $x \in \mathbb{Z}_p$, where exponentiation of a ciphertext is also coordinate-wise. Finally, the cryptosystem also has the property that

$$c \boxplus v := c \cdot g^v = E_h(m+v;r).$$

Encrypting matrices. We now describe the encryption for a matrix $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n\times n}$. First it chooses the secret keys p, q and constructs the cyclic group $\mathbb{G}$ of order $N = pq$. Then it picks generators $h_j \in_R \mathbb{G}_q$ for $j \in [n]$ according to the BGN encryption. The ciphertext matrix $\mathbf{C}$ consists of $(c_{i,j})$ where $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$ for all $i, j \in [n]$. The decryption key is q for all rows. We can also represent $\mathbf{C}$ as the matrices:

$$\mathbf{C} = \begin{pmatrix} h_1^{r_1} \cdot g^{m_{1,1}} & h_2^{r_1} \cdot g^{m_{1,2}} & \cdots & h_n^{r_1} \cdot g^{m_{1,n}} \\ \vdots & & \ddots & \vdots \\ h_1^{r_n} \cdot g^{m_{n,1}} & h_2^{r_n} \cdot g^{m_{n,2}} & \cdots & h_n^{r_n} \cdot g^{m_{n,n}} \end{pmatrix}.$$

Lemma 12. *The subgroup decision matrix encryption scheme described above produces indistinguishable ciphertexts under the subgroup decision assumption.*

Proof. Let $\mathbf{L} = (\ell_{i,j})$, $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n\times n}$ be any two arbitrary matrices. We first define a set of experiments $H_0, \ldots, H_{n^2}$. In experiment H_k, the output is a matrix $\mathbf{C} = (c_{i,j})$ chosen in the following way: choose secret/public key pairs q, h_j for $j \in [n]$ and exponents $r_i \in \mathbb{Z}_p$ for $i \in [n]$ as above. Then for the first k pairs $(i,j) \in [n]^2$ (where we order the pairs lexicographically), let $c_{i,j} = E_{h_j}(\ell_{i,j}; r_i)$. For the remaining $n^2 - k$ pairs (i,j), let $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$.

Observe that experiment H_0 produces an encryption of the matrix $\mathbf{L}$ and H_{n^2} produces an encryption of the matrix $\mathbf{M}$. Now we show that for every $k \in [n]^2$, experiments H_{k-1} and H_k are computationally indistinguishable.

For any $k \in [n]^2$, let (i^*, j^*) be the lexicographically k-th pair in $[n]^2$. Consider the following simulator algorithm S: the input is the subgroup decision problem instance $(N, \mathbb{G}, w)$. Set the public key $h^* = w$. Pick a random $r^* \in \mathbb{Z}_p$ and define a ciphertext $c^* = w^{r^*} g^{?}$. Notice that if $w \in \mathbb{G}_q$, then $c^* = E_{h^*}(?; r^*)$, which is an encryption of either $\ell_{i,j}$ or $m_{i,j}$.

S produces an encrypted matrix $\mathbf{C} = (c_{i,j})$ in the following way. First, for every $j \neq j^*$ it chooses the public keys $h_j \in_R \mathbb{G}_q$, and for every $i \neq i^*$ it chooses random exponents $r_i \in \mathbb{Z}_p$.

- For rows $i \neq i^*$, S encrypts normally: for $i < i^*$ and all $j \in [n]$, let $c_{i,j} = E_{h_j}(\ell_{i,j}; r_i)$; similarly for $i > i^*$ and all $j \in [n]$, let $c_{i,j} = E_{h_j}(m_{i,j}; r_i)$.
- For row $i = i^*$, S encrypts using the public key h_j. That is, for column $j < j^*$, let
$$c_{i,j} = {h_j}^{r^*} \cdot g^{\ell_{i,j}} = E_{h_j}(\ell_{i,j}; r^*),$$
and similarly for $j > j^*$ (encrypting $m_{i,j}$).
- Finally, for $i = i^*$ and $j = j^*$, let $c_{i,j} = c^*$.

One can see that S's output is distributed according to either H_{k-1} or H_k, depending on whether c^* was an encryption of $\ell_{i,j}$ or $m_{i,j}$ (respectively). If the adversary can distinguish between H_{k-1} and H_k, S can solve the subgroup decision problem. □

Lossy TDF. We formally describe the function generation, evaluation and inversion algorithm as follows:

- *Sampling an injective function.* The injective function generator $S_{\text{inj}}(1^\lambda)$ first selects $(p, q, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the matrix encryption (as described above) of the identity matrix $\mathbf{I} \in \mathbb{Z}_p^{n \times n}$. The function index is $(\mathbf{C}, N = pq, \mathbb{G}, g)$ and the trapdoor is q.
- *Sampling a lossy function.* The lossy function generator $S_{\text{loss}}(1^\lambda)$ first selects $(p, q, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the matrix encryption of the matrix $\mathbf{0} \in \mathbb{Z}_p^{n \times n}$. The function index is $(\mathbf{C}, N = pq, \mathbb{G}, g)$.
- *Evaluation.* F_{ltdf} takes as input $(s, \mathbf{x})$, where $s = (\mathbf{C}, N, \mathbb{G}, g)$ is the function index and $\mathbf{x} = \{x_1, \ldots, x_n\} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. Denote $\mathbf{C}$ as the matrix encryption of some matrix $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$. The output is $\mathbf{y} = \{y_1, \ldots y_n\}$ where:
$$y_j = \bigodot_{i \in [n]} c_{i,j}^{x_i} = E_{h_j}((\mathbf{xM})_j; R := \sum_{i \in [n]} x_i r_i).$$
- *Inversion.* F_{ltdf}^{-1} takes as input $(q, \mathbf{y})$, where q is the trapdoor and $\mathbf{y} = \{y_1, \ldots, y_n\}$. The output is $x = \{x_1, \ldots, x_n\} \in \{0,1\}^n$, where $x_j = D_q(y_j)$ for $j \in [n]$.

Theorem 34. *The algorithms described above give a collection of* $(n, n - \lg q)$*-lossy TDFs under the subgroup decision assumption.*

Proof. The invertibility for the injective function via the trapdoor information is shown in the algorithm. The indistinguishability between the injective and the lossy functions follows from the semantic security of the BGN encryption. We now show the lossiness property.

Note that for a function generated by $S_{\text{loss}}(1^\lambda)$, for any input x_j for $j \in [n]$, the output is $y_j = E_{h_j}(0; R)$ for some fixed R (dependent on $\mathbf{x}$). Since h_j has order q, the number of possible function outputs is at most q. Therefore the lossiness is $n - \lg q$. □

Note that even using the same security parameter 1^λ, the sampled λ-bit modulus N is different in each function. However it has an exponentially high probability that $N > 2^{\lambda/2}$. We can also approximate $q \approx \sqrt{N}$. The DDH construction in [174] has lossiness of $n - \lg p$ where p is a λ-bit prime.

All-But-One TDF. Let the set of branches $B = \{1, \ldots, \hat{N}\}$, where $\hat{N}$ is at most the smallest value of N produced by $\mathcal{G}(1^\lambda)$. We describe the function generation, evaluation and inversion algorithm as follows:

- *Sampling an ABO function.* The ABO function generator $S_{\text{abo}}(1^\lambda, b^* \in B)$ first selects $(p, q, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the matrix encryption of the matrix $-(b^*\mathbf{I}) \in \mathbb{Z}_p^{n \times n}$. The function index is $(\mathbf{C}, N = pq, \mathbb{G}, g)$ and the trapdoor is q with the lossy branch value b^*.

- *Evaluation.* G_{abo} takes as input $(s, b, \mathbf{x})$, where $s = (\mathbf{C}, N, \mathbb{G}, g)$ is the function index, $b \in B$ is the desired branch and $\mathbf{x}$ is an n-bit input interpreted as a vector. The output is a vector $\mathbf{y} = \mathbf{x}(\mathbf{C} \boxplus b\mathbf{I})$. By the homomorphic property of the matrix encryption, we can also write the j-th coordinate of $\mathbf{y}$ as:

$$y_j = E_{h_j}((b - b^*)x_j; R := \sum_{i \in [n]} x_i r_i).$$

- *Inversion.* G_{abo}^{-1} takes as input $(q, b, \mathbf{y})$, where q is the trapdoor and b is the evaluated branch. When $b \neq b^*$, the output is $\mathbf{x} = \{x_1, \ldots, x_n\}$ where $x_j = D_q(y_j)/(b - b^*)$ for $j \in [n]$.

Theorem 35. *The algorithms described above give a collection of* $(n, n - \lg q)$*-all-but-one TDFs under the subgroup decision assumption.*

Proof. The invertibility for the injective function via the trapdoor information is shown in the algorithm. The hidden lossy branch property follows from the semantic security of the BGN encryption. We now show the lossiness property.

Note that when $b = b^*$, for any input x_j for $j \in [n]$, the output is $y_j = E_{h_j}(0; R)$ for some fixed R (dependent on $\mathbf{x}$). Since h_j has order q, the number of possible function outputs is at most q. Therefore the lossiness is $n - \lg q$. □

Decision Linear

Linear encryption is a natural extension of the ElGamal encryption in DDH-easy groups such as the pairings. We can construct lossy TDFs in a way similar to Peikert and Waters's approach [174] to construct lossy TDFs from ElGamal encryption. Now, consider a generator algorithm $\mathcal{G}$ that, on input a security parameter 1^λ, outputs a tuple $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$, in which p is a uniform random λ-bit prime and $\mathbb{G}$, $\mathbb{G}_T$ are cyclic groups of order p, $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a pairing and g is a generator of $\mathbb{G}$.

Encryption from Decision Linear. Linear encryption is a natural extension of the ElGamal encryption in DDH-easy groups such as the pairings. The security of the linear encryption is based on the decision linear assumption. We modify the linear encryption scheme in [31] as follows. The user runs $\mathcal{G}(1^\lambda)$ to generate the public parameters. He picks a generator $u, v \in_R \mathbb{G}$ and random integers $\alpha, \beta \in_R \mathbb{Z}_p^*$. He computes w, h such that $u^\alpha = w$ and $v^\beta = h$. He also sets $g_t = \hat{e}(g, g)$. The secret key is (α, β) and the public key is $(u, v, h, w, g_t, p, \mathbb{G}, \mathbb{G}_T, \hat{e})$. To encrypt an $m \in \mathbb{Z}_p$, choose an $r_u, r_v \in_R \mathbb{Z}_p$ and create the ciphertext

$$E_{(w,h)}(m; r_u, r_v) = (u^{r_u}, v^{r_v}, \hat{e}(w, h)^{r_u + r_v} g_t^m).$$

To decrypt a ciphertext $c = (c_1, c_2, c_3)$, output

$$D_{\alpha,\beta}(c) = \log_{g_t} \frac{c_3}{\hat{e}(c_1^\alpha, h) \cdot \hat{e}(w, c_2^\beta)}.$$

When c encrypts a bit $m \in \{0, 1\}$, the discrete logarithm can be computed by enumeration.

Lemma 13. *The encryption scheme described above is semantically secure under the decision linear assumption.*

Proof. Suppose the algorithm $\mathcal{B}$ is given the decision linear problem instance $(u^*, u^{*a}, v^*, v^{*b}, h^*, h^{*c})$. If there is an adversary $\mathcal{A}$ who can break the semantic security of the

above encryption scheme, then $\mathcal{B}$ can use $\mathcal{A}$ to help him to solve the decision linear problem.

Firstly, $\mathcal{B}$ runs $\mathcal{G}(1^\lambda)$ to generate the public parameters $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$. $\mathcal{B}$ picks a random $w \in \mathbb{G}$ and calculates $g_t = \hat{e}(g, g)$. $\mathcal{B}$ sets the public key as $(u^*, v^*, h^*, w, g_t, p, \mathbb{G}, \mathbb{G}_T, \hat{e})$ and gives it to $\mathcal{A}$.

$\mathcal{B}$ receives the challenge message m_0 and m_1 from $\mathcal{A}$. $\mathcal{B}$ flips a coin $b' \in \{0, 1\}$ and returns the challenge ciphertext as:

$$C^* = (u^{*a}, v^{*b}, \hat{e}(w, h^{*c}) \cdot g_t^{m_{b'}}).$$

Finally $\mathcal{A}$ returns his guess $\bar{b}$. If $b' = \bar{b}$, then $\mathcal{B}$ answers $c = a + b$ to the decision linear problem. Otherwise, $\mathcal{B}$ answers $c \neq a + b$. □

The cryptosystem is additively homomorphic:

$$E_{(w,h)}(m; r_u, r_v) \odot E_{(w,h)}(m'; r'_u, r'_v) = E_{(w,h)}(m + m'; r_u + r'_u, r_v + r'_v),$$

where $\odot$ denotes coordinate-wise multiplication of ciphertexts. The cryptosystem also has the property that

$$c \boxplus m' := c \cdot g_t^{m'} = E_{w,h}(m + m'; r).$$

Encrypting matrices. We now describe the encryption for a matrix $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$. First choose some random generator $u, v \in_R \mathbb{G}$ and n independent secret keys (α_j, β_j) for $j \in [n]$. Then compute $w_j = u^{\alpha_j}, h_j = v^{\beta_j}$ for $j \in [n]$ according to the above linear encryption, and choose $2n$ independent exponents $r_{u,i}, r_{v,i} \in_R \mathbb{Z}_p$ for $i \in [n]$. Denote the encryption of $\mathbf{M}$ as $E_{w_j,h_j}(m_{i,j}; r_{u,i}, r_{v,i})$ for all $i, j \in [n]$. Then the ciphertext matrix $\mathbf{C}$ consists of $(c_{i,j})$ where $c_{i,j} = E_{w_j,h_j}(m_{i,j}; r_{u,i}, r_{v,i})$ for all $i, j \in [n]$. We can also represent $\mathbf{C}$ as:

$$\mathbf{C}_1 = \begin{pmatrix} u^{r_{u,1}} \\ \vdots \\ u^{r_{u,n}} \end{pmatrix}, \quad \mathbf{C}_2 = \begin{pmatrix} v^{r_{v,1}} \\ \vdots \\ v^{r_{v,n}} \end{pmatrix},$$

$$\mathbf{C}_3 = \begin{pmatrix} \hat{e}(w_1, h_1)^{r_{u,1}+r_{v,1}} \cdot g_t^{m_{1,1}} & \cdots & \hat{e}(w_n, h_n)^{r_{u,1}+r_{v,1}} \cdot g_t^{m_{1,n}} \\ \vdots & \ddots & \vdots \\ \hat{e}(w_1, h_1)^{r_{u,n}+r_{v,n}} \cdot g_t^{m_{n,1}} & \cdots & \hat{e}(w_n, h_n)^{r_{u,n}+r_{v,n}} \cdot g_t^{m_{n,n}} \end{pmatrix}.$$

Lemma 14. *The matrix encryption scheme described above produces indistinguishable ciphertexts under the decision linear assumption.*

Proof. Let $\mathbf{L} = (\ell_{i,j})$, $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$ be any two arbitrary matrices. We first define a set of experiments $H_0, \ldots, H_{n^2}$. In experiment H_k, the output is a matrix $\mathbf{C} = (c_{i,j})$ chosen in the following way: choose secret/public key pairs $(\alpha_j, \beta_j), (w_j, h_j)$ for $j \in [n]$ and exponents $r_{u,i}, r_{v,i} \in \mathbb{Z}_p$ for $i \in [n]$ as above. Then for the first k pairs $(i,j) \in [n]^2$ (where we order the pairs lexicographically), let $c_{i,j} = E_{w_j,h_j}(\ell_{i,j}; r_{u,i}, r_{v,i})$. For the remaining $n^2 - k$ pairs (i,j), let $c_{i,j} = E_{w_j,h_j}(m_{i,j}; r_{u,i}, r_{v,i})$.

Observe that experiment H_0 produces an encryption of the matrix $\mathbf{L}$ and H_{n^2} produces an encryption of the matrix $\mathbf{M}$. Now we show that for every $k \in [n]^2$, experiments H_{k-1} and H_k are computationally indistinguishable.

For any $k \in [n]^2$, let (i^*, j^*) be the lexicographically k-th pair in $[n]^2$. Consider the following simulator algorithm S: the input is the decision linear problem instance $(u^*, u^{*a}, v^*, v^{*b}, h^*, h^{*c})$. For the public key (u^*, v^*, h^*, w^*), define a ciphertext $c^* = (c_1^*, c_2^*, c_3^*) = (u^{*a}, v^{*b}, \hat{e}(w^*, h^*)^c g_t^?)$. Notice that if $c = a + b$, then $c^* = E_{w^*,h^*}(?; a, b)$, which is an encryption of either $\ell_{i,j}$ or $m_{i,j}$.

S produces an encrypted matrix $\mathbf{C} = (c_{i,j})$ in the following way. First, it randomly chooses $w^* \in \mathbb{G}$ and sets $u = u^*$ and $v = v^*$. Then for every $j \neq j^*$ it randomly chooses $\alpha_j, \beta_j \in \mathbb{Z}_p^*$ and calculates $w_j = u^{*\alpha_j}$ and $h_j = v^{*\beta_j}$. For every $i \neq i^*$ it chooses random exponents $r_{u,i}, r_{v,i} \in \mathbb{Z}_p$.

- For rows $i \neq i^*$, S encrypts normally: for $i < i^*$ and all $j \in [n]$, let $c_{i,j} = E_{w_j,h_j}(\ell_{i,j}; r_{u,i}, r_{v,i})$; similarly for $i > i^*$ and all $j \in [n]$, let $c_{i,j} = E_{w_j,h_j}(m_{i,j}; r_{u,i}, r_{v,i})$.

- For row $i = i^*$, S encrypts using (α_j, β_j). That is, for column $j < j^*$, let

$$\begin{aligned} c_{i,j} &= (u^{*a}, v^{*b}, \hat{e}(u^{*a\alpha_j}, h_j) \cdot \hat{e}(w_j, v^{*b\beta_j}) \cdot g_t^{\ell_{i,j}}) \\ &= (u^{*a}, v^{*b}, \hat{e}(w_j, h_j)^{a+b} \cdot g_t^{\ell_{i,j}}) \\ &= E_{w_j,h_j}(\ell_{i,j}; a, b), \end{aligned}$$

  and similarly for $j > j^*$ (encrypting $m_{i,j}$).

- Finally, for $i = i^*$ and $j = j^*$, let $c_{i,j} = c^*$.

One can see that S's output is distributed according to either H_{k-1} or H_k, depending on whether c^* was an encryption of $\ell_{i,j}$ or $m_{i,j}$ (respectively). If the adversary can distinguish between H_{k-1} and H_k, S can solve the decision linear problem. □

Lossy TDF. We describe the function generation, evaluation and inversion algorithm as follows:

- *Sampling an injective function.* The injective function generator $S_{\text{inj}}(1^\lambda)$ first selects $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g) \leftarrow \mathcal{G}(1^\lambda)$. The function indices are $(\mathbf{C}, p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$, where $\mathbf{C}$ is a matrix encryption of the identity $\mathbf{I} \in \mathbb{Z}_p^{n \times n}$, and the trapdoor t is (α_j, β_j) for $j \in [n]$.

- *Sampling a lossy function.* The lossy function generator $S_{\text{loss}}(1^\lambda)$ first selects $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g) \leftarrow \mathcal{G}(1^\lambda)$. The function indices are $(\mathbf{C}, p, \mathbb{G}, g)$, where $\mathbf{C}$ is a matrix encryption of $\mathbf{0} \in \mathbb{Z}_p^{n \times n}$.

- *Evaluation.* F_{ltdf} takes as input $(s, \mathbf{x})$, where $s = (\mathbf{C}, p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$ is the function index and $\mathbf{x} \in \{0,1\}^n$ is interpreted as a vector. The output is a vector $\mathbf{y} = \mathbf{xC}$, where:

$$\begin{aligned} y_j &:= \bigodot_{i \in [n]} c_{i,j}^{x_i} \\ &= E_{w_j,h_j}((\mathbf{xM})_j; R_u = \langle \mathbf{x}, \mathbf{r_u} \rangle, R_v = \langle \mathbf{x}, \mathbf{r_v} \rangle), \end{aligned}$$

  where $\mathbf{M}$ is the identity $\mathbf{I}$ or $\mathbf{0}$ and $\mathbf{r_u} = (r_{u,1}, \ldots, r_{u,n})$ and $\mathbf{r_v} = (r_{v,1}, \ldots, r_{v,n})$ is the vector of random numbers used to construct $\mathbf{C}$. Note that the randomness R_u and R_v in y_j are the same for all $j \in [n]$. Therefore we can represent $\mathbf{xC}_1$ as $\prod_{i \in [n]} u^{x_i r_{u,i}}$ and represent $\mathbf{xC}_2$ using $\prod_{i \in [n]} v^{x_i r_{v,i}}$. As a result, we may represent $\mathbf{y}$ more compactly using $n + 2$ group elements.

- *Inversion.* F_{ltdf}^{-1} takes as input $(t, \mathbf{y})$, where $t = (\alpha_j, \beta_j)$ for $j \in [n]$ is the trapdoor. The output is $\mathbf{x} \in \{0,1\}^n$ where $x_j = D_{\alpha_j,\beta_j}(y_j)$.

Theorem 36. *The algorithm described above gives a collection of $(n, n - \lg p)$-lossy TDFs under the decision linear assumption.*

Proof. The invertibility for the injective function via the trapdoor information is shown above. The indistinguishability between the injective and the lossy functions follows by Lemma 14. We now show the lossiness property.

Note that for a function generated by $S_{\text{loss}}(1^\lambda)$, for any input x, the output is $\mathbf{y}$ such that $y_j = E_{w_j,h_j}(0; R_u, R_v)$ for some fixed $R_u, R_v \in \mathbb{Z}_p$ and fixed w_j, h_j. The number of possible function outputs is at most p. Therefore the lossiness is $n - \lg p$. □

All-But-One TDF. Let the set of branches $B = [q]$, where q is at most the smallest value of p produced by $\mathcal{G}(1^\lambda)$. We formally describe the function generation, evaluation and inversion algorithm as follows:

- *Sampling an ABO function.* The ABO function generator $S_{\text{abo}}(1^\lambda, b^* \in B)$ first selects $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g) \leftarrow \mathcal{G}(1^\lambda)$. The function indices are a matrix encryption $\mathbf{C}$ of the matrix $-(b^*\mathbf{I}) \in \mathbb{Z}_p^{n\times n}$, and the group description $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$. The trapdoor is (α_j, β_j) for $j \in [n]$ together with the lossy branch value b^*.

- *Evaluation.* G_{abo} takes as input $(s, b, \mathbf{x})$, where $s = (\mathbf{C}, p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$ is the function index, $b \in B$ is the desired branch and $\mathbf{x}$ is an n-bit input interpreted as a vector. The output is $\mathbf{y} = \mathbf{x}(\mathbf{C} \boxplus b\mathbf{I})$, where $\boxplus$ is the homomorphic scalar addition operation applies entry-wise to the matrices and where:

$$y_j = E_{w_j,h_j}((b - b^*)x_j; R_u = \langle \mathbf{x}, \mathbf{r_u} \rangle, R_v = \langle \mathbf{x}, \mathbf{r_v} \rangle),$$

  where $\mathbf{r_u} = (r_{u,1}, \ldots, r_{u,n})$ and $\mathbf{r_v} = (r_{v,1}, \ldots, r_{v,n})$ are the vectors of random numbers used to construct $\mathbf{C}$. Note that the randomness R_u and R_v in y_j is the same for all $j \in [n]$. As before, we may represent $\mathbf{y}$ more compactly using $n + 2$ group elements.

- *Inversion.* G^{-1}_{abo} takes as input $(t, b, \mathbf{y})$, where $t = (\alpha_j, \beta_j)$ for $j \in [n]$ is the trapdoor and b is the evaluated branch. When $b \neq b^*$, the output is $\mathbf{x} \in \{0, 1\}^n$, where $x_j = D_{\alpha_j,\beta_j}(y_j)/(b - b^*)$.

Theorem 37. *The algorithm described above gives a collection of $(n, n - \lg p)$-all-but-one TDFs under the decision linear assumption.*

Proof. The invertibility for the injective function via the trapdoor information is shown above. The indistinguishability between the injective and the lossy functions follows by Lemma 14. We now show the lossiness property.

Note that for a function generated by $S_{\text{abo}}(1^\lambda, b^*)$, for any input b and $\mathbf{x} \in \{0, 1\}^n$ with $b = b^*$, the output is $\mathbf{y}$ where $y_j = E_{w_j,h_j}(0; R_u, R_v)$ for some fixed $R_u, R_v \in \mathbb{Z}_p$ and fixed w_j, h_j. The number of possible function outputs is at most p. Therefore the lossiness is $n - \lg p$. □

6.1.2 More Applications using Lossy and ABO Trapdoor Functions

Peikert and Waters [174] showed how to construct collision-resistant hash functions, oblivious transfer and CCA-secure encryption from lossy and ABO trapdoor functions. In this section, we will show how to construct other cryptographic primitives from these functions.

Tag-Based Encryption

In tag-based encryption, the tag in the ciphertext provides the context within which the ciphertext is to be decrypted [192]. The applications of tag-based encryption include onion routing [50]. We only consider the selective-tag weak CCA security model by Kiltz [130].

Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ be a collection of (n,k)-lossy TDFs. Let $(S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ be a collection of (n,k')-ABO TDFs. We require that the total lossiness $k+k' \geq n+\kappa$ for some $\kappa = \omega(n)$. Let $\mathcal{H}$ be a family of pairwise independent collision-resistant hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq \kappa - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. The construction is as follows:

- **TBEkg.** The algorithm takes as input the security parameter 1^λ and generates a lossy TDF: $(s,t) \leftarrow S_{\text{ltdf}}(1^\lambda)$. It then generates an ABO TDF having lossy branch 0^v: $(s',t') \leftarrow S_{\text{abo}}(1^\lambda, 0^v)$. It also chooses a hash function $h \leftarrow \mathcal{H}$. The public key is $pk = (s, s', h)$ and the secret key is $sk = (t, t', pk)$.

- **TBEenc.** The algorithm takes as input (pk, tag, m), where $pk = (s, s', h)$ is the public key and $m \in \{0,1\}^\ell$ is the message. It chooses a random $x \leftarrow \{0,1\}^n$ and computes the ciphertext $C = (c_1, c_2, c_3)$, where:

$$c_1 = F_{\text{ltdf}}(s, x), \quad c_2 = G_{\text{abo}}(s', tag, x), \quad c_3 = m \oplus h(x).$$

- **TBEdec.** The algorithm takes as input (sk, tag, C), where $sk = (t, t', (s, s', h))$ is the secret key and $C = (c_1, c_2, c_3)$ is the ciphertext. It computes $x = F_{\text{ltdf}}^{-1}(t, c_1)$. It checks that $c_1 = F_{\text{ltdf}}(s, x)$ and $c_2 = G_{\text{abo}}(s', tag, x)$; if not, it outputs $\perp$. Finally it outputs $m = c_3 \oplus h(x)$.

Theorem 38. *The algorithms above are tag-based encryption which is selective-tag secure against weak chosen-ciphertext attack.*

Proof. The correctness of the above algorithms is straightforward. We prove the security by first describing a sequence of games $Game_1, \ldots, Game_4$, where $Game_1$ is the selective-tag, weak CCA game. Then we show that $Game_i$ and $Game_{i+1}$ are indistinguishable for $i = 1, \ldots, 3$. Finally we will show that the adversary must have negligible advantage in $Game_4$.

We review the definition of $Game_1$ (selective-tag, weak CCA game) here:

- Setup. On input 1^λ, the adversary $\mathcal{A}$ gives tag^* to the challenger $\mathcal{C}$. Then $\mathcal{C}$ runs $(pk, sk) \leftarrow \mathbf{TBEkg}(1^\lambda)$ and gives pk to the adversary.

- Decrypt. The adversary $\mathcal{A}$ queries the oracle with input the ciphertext C and tag where $tag \neq tag^*$. $\mathcal{C}$ returns the decrypted message $m/\perp \leftarrow \mathbf{TBEenc}(sk, tag, C)$.

- Challenge. $\mathcal{A}$ gives two messages m_0 and m_1 in the message space to $\mathcal{C}$. $\mathcal{C}$ picks a bit $b = 0/1$ and returns $C^* = \mathbf{TBEenc}(pk, tag^*, m_b)$ to $\mathcal{A}$.

$\mathcal{A}$ can query the decryption oracle again before he finally outputs his guess b'. When referring to an implementation of these algorithms in a specific game i, we use a subscript i, e.g. Setup_1.

- $Game_1$: It is the selective-tag, weak CCA game. In particular, he chooses the ABO function lossy branch to be 0^v and decrypts using the trapdoor t.

- $Game_2$: In this game, the only change is in Setup_2, in which the ABO function lossy branch is chosen to be tag^*.

- $Game_3$: In this game, $\mathsf{Setup}_2 = \mathsf{Setup}_3$. The only change is in $\mathsf{Decrypt}_3$, in which the decryption is now done by the ABO trapdoor t'. During **TBEdec**, he replaces $x = F^{-1}_{\text{ltdf}}(t, c_1)$ with $x = G^{-1}_{\text{abo}}(t', tag^*, c_2)$.

- $Game_4$: In this game, $\mathsf{Decrypt}_3 = \mathsf{Decrypt}_4$. The only change is in Setup_4, in which we replace the injective function $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$ with the lossy one $(s, \perp) \leftarrow S_{\text{loss}}(1^\lambda)$.

We now prove a sequence of claims which lead to the main theorem.

Claim 7. *$Game_1$ and $Game_2$ are computationally indistinguishable, given the hidden lossy branch property of the ABO TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, br)$ where $br \in B_\lambda$. During Setup, $\mathcal{S}$ runs $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, br)$. $\mathcal{S}$ also runs $(s, t) \leftarrow S_{\text{inj}}(1^\lambda)$, $h \leftarrow \mathcal{H}$. The public key is $pk = (s, s', h)$. For Setup_1, we have $br = 0^v$; for Setup_2, we have $br = tag^*$. By the hidden lossy branch property, the public keys are computationally indistinguishable from Setup_1 and Setup_2.

Notice that $\mathsf{Decrypt}$ is simulated using t in both games and $\mathsf{Challenge}$ is simulated as in the algorithm description. Therefore $\mathcal{S}$ perfectly simulates $Game_1$ and $Game_2$. □

Claim 8. *$Game_2$ and $Game_3$ are perfectly equivalent (if the lossy and ABO TDF collections are both perfect) or statistically close (if either the lossy or ABO TDF collections is almost-always).*

Proof. In both $Game_2$ and $Game_3$, (s, t) is generated by $S_{\text{inj}}(1^\lambda)$ and (s', t') is generated by $S_{\text{abo}}(1^\lambda, tag^*)$. During the $\mathsf{Decrypt}_2(C, tag)$ (or $\mathsf{Decrypt}_3(C, tag)$) query, $\mathcal{S}$ checks if $c_1 = F_{\text{ltdf}}(s, x) = f_s(x)$ and $c_2 = G_{\text{abo}}(s', tag, x) = g_{s',tag}(x)$; and output $\perp$ if not. Now f_s and $g_{s',tag}$ are both injective (since $tag \neq tag^*$ by the selective-tag weak CCA model). Therefore there is a unique x such that $(c_1, c_2) = (f_s(x), g_{s'tag}(x))$. $\mathsf{Decrypt}_2$ finds x by computing $F^{-1}_{\text{ltdf}}(t, c_1)$ and $\mathsf{Decrypt}_3$ finds x by computing $G^{-1}_{\text{abo}}(t', tag, c_2)$. Therefore $\mathsf{Decrypt}$ is perfectly equivalent in the two games (or with overwhelming probability, if the trapdoor systems are almost-always). □

Claim 9. *$Game_3$ and $Game_4$ are computationally indistinguishable, given the indistinguishability of the injective and lossy functions of the lossy TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, s)$. During Setup, $\mathcal{S}$ runs $(s', t') \leftarrow S_{\text{abo}}(1^\lambda, tag^*)$ and $h \leftarrow \mathcal{H}$. For Setup_3, s was generated by $S_{\text{inj}}(1^\lambda)$; for Setup_4, s was generated by $S_{\text{loss}}(1^\lambda)$. The public key is $pk = (s, s', h)$. By the indistinguishability of the injective and lossy functions of the lossy TDF collection, the public keys are computationally indistinguishable from Setup_3 and Setup_4.

Notice that $\mathsf{Decrypt}$ is simulated using t' in both games and $\mathsf{Challenge}$ is simulated as in the algorithm description. Therefore $\mathcal{S}$ perfectly simulates $Game_3$ and $Game_4$. □

Claim 10. *No adversary has more than a negligible advantage in $Game_4$.*

Proof. Fix all the randomness in $Game_4$, except the choice of the hash function h and the randomness x^* used to produce the challenge ciphertext C^*. We observe that $F_{\text{ltdf}}(s, \cdot) = f_s(\cdot)$ and $G_{\text{abo}}(s', tag, \cdot) = g_{s',tag}(\cdot)$ are lossy functions with image size at most 2^{n-k} and $2^{n-k'}$ respectively. Therefore the random variable $(c_1^*, c_2^*) = (f_s(x^*), g_{s',tag}(x^*))$ can take at most $2^{2n-(k+k')} \leq 2^{n-\kappa}$ values by the requirement that $k + k' \geq n + \kappa$. By lemma 2 and the hypothesis that $\ell \leq \kappa - 2\lg(1/\epsilon)$, we have that $(c_1^*, c_2^*, h, h(x^*))$ and $(c_1^*, c_2^*, h, U_\ell)$ are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance. □

The main theorem follows the above claims. □

Note that when we combine the proposed tag-based encryption scheme with the transformation from tag-based encryption to public key encryption in [130], we obtain a CCA secure public key encryption scheme which is identical to the scheme in [174]. The relationships between these schemes are shown in Table 6.3.

Kiltz [130] proposed the first tag-based encryption scheme without pairing operations under the decision linear assumption. However their security proofs have to be carried out in gap-groups, such that an DDH oracle exists. By using our tag-based

Table 6.3: Black-box constructions between lossy TDFs, tag-based encryption and public key encryption.

	This section	
Lossy TDFs	$\longrightarrow$	Tag-Based Enc.
[174] $\searrow$		$\swarrow$ [130]
	Public Key Enc.	

encryption construction, we can obtain the first generic tag-based encryption scheme from lossy TDFs. The lossy TDFs can be instantiated by the DDH assumption only or the worst-case complexity of lattice problem [174]; or by some pairing-based assumptions appear in later sections. Furthermore, this is the first tag-based encryption scheme whose security is based on the complexity of lattice problems.

Universal Re-encryption

Universal re-encryption [100] allows a third party to re-randomise the ciphertext, without the knowledge of the plaintext nor the corresponding public key. Its applications include different types of mixnets and RFID tags.

To construct universal re-encryption from a lossy TDF, we require that the lossy TDF itself has an additional property: the evaluation function $F_{\mathrm{ltdf}}(\cdot, x)$ is of the form $E_y(x; r)$, where E is additively homomorphic encryption with respect to the public key y. Let $(S_{\mathrm{ltdf}}, F_{\mathrm{ltdf}}, F_{\mathrm{ltdf}}^{-1})$ give such a collection of (n, k)-lossy TDFs. Let $\mathcal{H}$ be a family of pairwise independent collision-resistant hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq k - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$.

- **UKg**(1^λ): It first generates a lossy TDF: $(s, t) \leftarrow S_{\mathrm{ltdf}}(1^\lambda)$. It also chooses a hash function $h \leftarrow \mathcal{H}$. The public key is $pk = (s, h)$ and the secret key is $sk = t$.

- **UE**(pk, m): It chooses $x \leftarrow \{0,1\}^n$ uniformly at random and some random numbers r, r' in the corresponding domain. It computes:

$$c_1 = F_{\mathrm{ltdf}}(s, x) = E_y(x; r), \quad c_2 = E_y(0; r'), \quad c_3 = E_y(1, r'), \quad c_4 = m \oplus h(x).$$

  The ciphertext is $C = (c_1, c_2, c_3, c_4)$.

- **URe**(C): For the i-th re-encryption, it parses $C = (c_1, c_2, c_3, c_4, \mathbf{b})$ where $\mathbf{b} = \{b_1, \ldots, b_{i-1}\}$ is the set of re-encryption tag ($\mathbf{b}$ is empty if C has never been re-encrypted before). It picks random numbers $\bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{r}_{b_1}, \ldots, \bar{r}_{b_{i-1}}$ and a random

$x_i \leftarrow \{0,1\}^n$ to compute:

$$c'_1 = c_1 \odot c_2^{\bar{r}_1}, \quad c'_2 = c_2^{\bar{r}_2}, \quad c'_3 = c_3 \odot c_2^{\bar{r}_2 - 1},$$
$$c'_4 = c_4 \oplus h(x_i), \quad b'_i = {c_3}^{x_i} \odot c_2^{\bar{r}_3}, \quad b'_j = b_j \odot c_2^{\bar{r}_{b_j}},$$

for $j = 1, \ldots, i-1$. It sets $\mathbf{b}' = \{b'_1, \ldots, b'_i\}$. The ciphertext is $C' = (c'_1, c'_2, c'_3, c'_4, \mathbf{b}')$. We can see that it is equivalent to:

$$c'_1 = E_y(x; r + r'\bar{r}_1), \quad c'_2 = E_y(0; r'\bar{r}_2), \quad c'_3 = E_y(1; r'\bar{r}_2),$$
$$b'_i = E_y(x_i; r'(x_i + \bar{r}_3)), \quad b'_j = E_y(x_i; r_{b_j} + \bar{r}_{b_j}).$$

- $\mathbf{UD}(sk, C)$: It parses $C = (c_1, c_2, c_3, c_4, (b_1, \ldots, b_j))$. It computes $x = F^{-1}_{\mathrm{ltdf}}(sk, c_1)$ and $x_i = F^{-1}_{\mathrm{ltdf}}(sk, b_i)$ for $1 \leq i \leq j$ (if exists). Finally it outputs $m = c_3 \oplus h(x) \oplus h(x_1) \oplus \cdots \oplus h(x_j)$.

Theorem 39. *The algorithms above are universal re-encryption which is semantically secure under encryption.*

Proof. The correctness of the scheme is straightforward. We prove the security by first describing two games $Game_1$ and $Game_2$. We review the definition of $Game_1$ (semantically secure encryption game [100]) here:

- Setup. On input 1^λ, the challenger $\mathcal{C}$ runs $(pk, sk) \leftarrow \mathbf{UKg}(1^\lambda)$ and gives pk to the adversary $\mathcal{A}$.
- $\mathsf{Challenge}$. $\mathcal{A}$ gives $\mathcal{C}$ two messages m^*_0 and m^*_1. $\mathcal{C}$ picks a bit $b = 0/1$ and encrypts $C^* \leftarrow \mathbf{UE}(pk, m^*_b)$. $\mathcal{C}$ returns C^* to $\mathcal{A}$.

$\mathcal{A}$ finally outputs his guess b'. When referring to an implementation of these algorithms in a specific game i, we use a subscript i, e.g. Setup_1. We now describe the two games:

$Game_1$: $\mathcal{S}$ runs Setup_1 by using $(s, t) \leftarrow S_{\mathrm{inj}}(1^\lambda)$ in $\mathbf{UKg}(1^\lambda)$.
$Game_2$: In Setup_2, we replace the injective function $(s, t) \leftarrow S_{\mathrm{inj}}(1^\lambda)$ with the lossy one $(s, \perp) \leftarrow S_{\mathrm{loss}}(1^\lambda)$.

Claim 11. *$Game_1$ and $Game_2$ are computationally indistinguishable, given the indistinguishability of the injective and lossy functions of the lossy TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, s)$. During Setup, $\mathcal{S}$ runs $h \leftarrow \mathcal{H}$. For Setup_1, s was generated by $S_{\mathrm{inj}}(1^\lambda)$; for Setup_2, s was generated by $S_{\mathrm{loss}}(1^\lambda)$. The

public key is $pk = (s, h)$. By the indistinguishability of the injective and lossy functions of the lossy TDF collection, the public keys are computationally indistinguishable from Setup_1 and Setup_2. Notice that $\mathsf{Challenge}$ is simulated as in the algorithm description. Therefore $\mathcal{S}$ perfectly simulates $Game_1$ and $Game_2$. □

Claim 12. *No adversary has more than a negligible advantage in $Game_2$.*

Proof. Fix all the randomness in $Game_2$, except the choice of the hash function h and the randomness x^* used to produce the challenge ciphertext C^*. We observe that $F_{\mathrm{ltdf}}(s, \cdot) = f_s(\cdot)$ is a lossy function with image size at most 2^{n-k}. Therefore the random variable $c_1^* = f_s(x^*)$ can take at most 2^{n-k} values. Notice that c_2^* and c_3^* are independent of c_1^* and c_4^* (and hence m_b^*). By lemma 2 and the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, we have that $(c_1^*, h, h(x^*))$ and (c_1^*, h, U_ℓ) are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance. □

The main theorem follows from the above claims. □

Theorem 40. *The algorithms above are an universal re-encryption which is universally semantically secure under re-encryption.*

Proof. We prove the security by first describing two games $Game_1$ and $Game_2$, where $Game_1$ is the standard semantic security game. Then we show that $Game_1$ and $Game_2$ are indistinguishable. Finally we will show that the adversary must have negligible advantage in $Game_2$. We review the definition of $Game_1$ (semantically secure re-encryption game [100]) here:

- Setup. On input 1^λ, the challenger $\mathcal{C}$ runs $\mathbf{UKg}(1^\lambda)$ twice to generate two public keys pk_0 and pk_1. $\mathcal{C}$ gives pk_0 and pk_1 to the adversary $\mathcal{A}$.
- $\mathsf{Challenge}$. $\mathcal{A}$ gives $\mathcal{C}$ two messages m_0^* and m_1^*; and two sets of randomness r_0 and r_1. $\mathcal{C}$ encrypts $C_0^* \leftarrow \mathbf{UE}(pk_0, m_0^*)$ using randomness r_0 and $C_1^* \leftarrow \mathbf{UE}(pk_1, m_1^*)$ using randomness r_1. $\mathcal{S}$ picks a bit $b = 0/1$ and computes $C'^*_0 = \mathbf{URe}(C_b^*)$ and $C'^*_1 = \mathbf{URe}(C_{1-b}^*)$. $\mathcal{S}$ returns C'^*_0 and C'^*_1 to $\mathcal{A}$.

$\mathcal{A}$ finally outputs his guess b'. When referring to an implementation of these algorithms in a specific game i, we use a subscript i, e.g. Setup_1. We describe the games as follows:

$Game_1$: $\mathcal{S}$ runs Setup_1 by using $(s, t) \leftarrow S_{\mathrm{inj}}(1^\lambda)$ in $\mathbf{UKg}(1^\lambda)$.
$Game_2$: In Setup_2, we replace the injective function $(s, t) \leftarrow S_{\mathrm{inj}}(1^\lambda)$ with the lossy one $(s, \perp) \leftarrow S_{\mathrm{loss}}(1^\lambda)$.

Same as claim 11, $Game_1$ and $Game_2$ are computationally indistinguishable, given the indistinguishability of the injective and lossy functions of the lossy TDF collection. Therefore we can prove the theorem by showing that no adversary has more than a negligible advantage in $Game_2$.

Fix all the randomness in $Game_2$, except the choice of the hash function h and the randomness x_1^* used to produce the challenge ciphertexts $C'^*_0 = (c'^*_{0,1}, \ldots, c'^*_{0,4}, b'^*_{0,1})$ and $C'^*_1 = (c'^*_{1,1}, \ldots, c'^*_{1,4}, b'^*_{1,1})$. We observe that $F_{\text{ltdf}}(s, \cdot) = f_s(\cdot)$ is a lossy function with image size at most 2^{n-k}. Therefore the random variable $b'^*_{\cdot,1} = f_s(x^*_{\cdot,1})$ can take at most 2^{n-k} values. By lemma 2 and the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, we have that $(b'^*_{\cdot,1}, h, h(x^*_{\cdot,1}))$ and $(b'^*_{\cdot,1}, h, U_\ell)$ are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance.

WLOG, assume C_0^* is re-encrypted to C'^*_0. For $\rho = 1, \ldots, 4$, $c'^*_{0,\rho}$ and $c^*_{0,\rho}$ are encryption of the same message with different randomness. By the semantic security of the encryption from theorem 39, they are indistinguishable. (If not, the adversary can make encryptions of the challenge messages by himself, and try to distinguish between them and the challenge ciphertext. He breaks the semantic security of the encryption if he succeeds.) □

Notice that our universal re-encryption scheme has a ciphertext which grows linearly to the number of re-encryption performed. Therefore the number of re-encryption performed is known. Nevertheless, our universal re-encryption can be built based on the generic homomorphic additive lossy TDFs, which includes the one based on the DDH problem or the LWE problem from lattice in [174], or the subgroup decision problem or the decision linear problem in this section.

6.2 Two-Tier Encryption and Two-Tier Trapdoor Functions

In the public key cryptography, cryptosystems are usually far more complicated than merely a simple encryption or signature scheme involving only Alice and Bob. Traditional public key encryption or signatures are only useful for point-to-point communication. Many cryptographic primitives have to introduce a key generating party which is different from Alice and Bob, like the group manager in group signatures, or the private key generator in identity-based cryptography. Therefore, it is more natural to consider the existence of a third party in the real world applications. For example in the case of subscription of Internet Protocol Television (IPTV) service, the broadcaster

does not encrypt the TV programmes using each subscriber's public key. Instead, the subscribers have to obtain their decryption keys from the IPTV company and the broadcaster encrypts using the IPTV company's public key. Another motivating scenario is in the case of e-voting. In an e-voting scenario, a voter may obtain a signing key from an election authority and signs his vote. Anyone can verify that the vote is signed by one of the voters approved by the election authority, without knowing the identity of the voter.

Fine-grained access control systems grant differential access rights to a set of users and provide delicate control over access to sensitive data. There are several known techniques to implement fine-grained access control. Software-based access control systems checks if a user is authorised to access a piece of data. However, this method is not desirable from security point of view. If the database is compromised (for example, due to a software vulnerability exploit or an insider attack), the loss will be disastrous. Recently, some encryption schemes are proposed to provide fine-grained access control. A common point in these schemes is that there is a key generating party to generate keys to users. These access control schemes can be classified into two types:

- The ciphertexts are associated with attributes [103, 127]. The key generating party is responsible to decide who can decrypt. For example, a ciphertext is associated with the attributes "database" and "security". A staff with the access right of "database AND security AND web" is able to decrypt the ciphertext.

- The user secret keys are associated with attributes [24]. The encryptor decides who can decrypt. For example, Alice is associated with the attributes "Manager" and "IT department". Bob wants to encrypt a message to all staffs in the IT department with at least manager position. Then Alice can decrypt the ciphertext from Bob.

A more fine-grained access control would be a combination of the above two types. The ciphertexts and the users' secret keys are both associated with attributes. The key generating party and the encryptor are both responsible to decide who can decrypt. For example, Bob can encrypt a message, with keywords "database" and "security", to all staffs in the IT department with at least manager position. The IT department manager Alice, with the access right of "database AND security", is able to decrypt the ciphertext.

The applications described above require a new cryptographic primitive which is

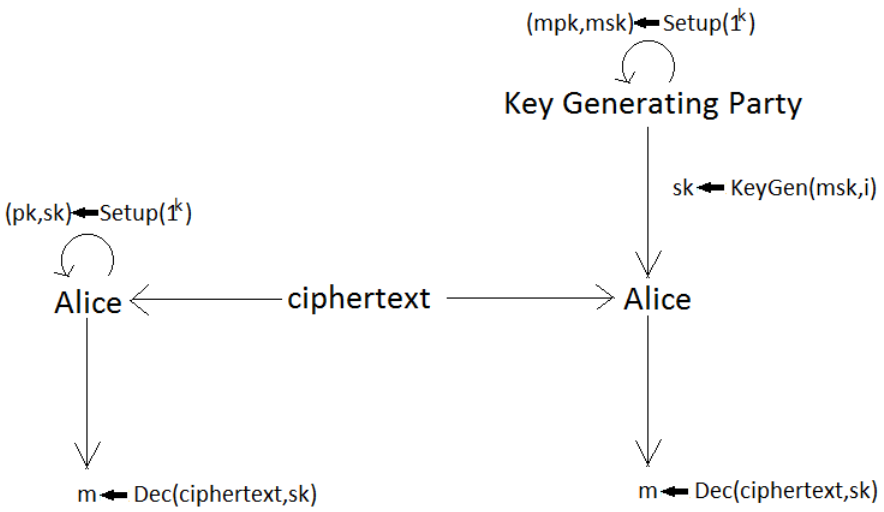


Figure 6.1: Comparison of the public key encryption (left) and the two-tier encryption (right).

different from the traditional public key encryption or signatures. Two-tier cryptosystem is a new general primitive to capture the system that involves a key generating party. A key generating party interacts with a user to generate a user secret key usk associated with some information i. For a two-tier encryption, a sender encrypts a message M corresponds to some information i'. The user can use usk to decrypt and obtain M if $(i, i') \in \mathcal{R}$ for some relation $\mathcal{R}$ defined in the setup process. Different instantiation of the relation $\mathcal{R}$ gives different cryptographic primitives.

New Framework. We introduce a new encryption framework called *two-tier encryption*. It is a framework to capture encryption schemes which have a party responsible for generating and distributing secret keys to different users. The secret key sk corresponds to some user information i. The user can use the secret key sk to decrypt the ciphertext. During the encryption algorithm, a message is encrypted with respect to some information i' and the algorithm outputs the ciphertext C. The decryption of C is correct if the user has the secret key sk such that $(\mathsf{i}, \mathsf{i}') \in \mathcal{R}$, where $\mathcal{R}$ is some relation. We compare the public key encryption and the two-tier encryption in Figure 6.1.

Different instantiations of the key generation algorithms, and the definition of i, i' and $\mathcal{R}$ result in different cryptographic primitives. For example, in the identity-based

Table 6.4: Primitives from two-tier encryption.

Two-tier Encryption	Identity-based Encryption
	Attribute-based Encryption
	Broadcast Encryption
	Certificate-based Encryption
	Group Encryption
	Predicate Encryption

encryption, the information i and i' are both the user identity and

$$(\mathsf{i}, \mathsf{i}') \in \mathcal{R} \qquad \text{if} \qquad \mathsf{i} = \mathsf{i}'.$$

On the other hand, in the broadcast encryption, the information i is the user index, the information i' is the set of legitimate users and

$$(\mathsf{i}, \mathsf{i}') \in \mathcal{R} \qquad \text{if} \qquad \mathsf{i} \in \mathsf{i}'.$$

In this section, we will also demonstrate how the attribute-based encryption, the certificate-based encryption, the group encryption, and the predicate encryption are related to the framework of *two-tier encryption*. It is summarised in Table 6.4.

Our Approaches. We observe that the security of encryption schemes having a two-tier structure are often reduced to some complexity assumptions which also have a two-tier structure. For example, the decisional bilinear Diffie-Hellman (DBDH) assumption used in the Boneh-Franklin identity-based encryption (IBE) [34] implies the computational Diffie-Hellman (CDH) assumption. The CDH assumption protects the user secret key calculated by the key generating party against other users in [34]. Therefore, it seems that complexity assumptions having a two-tier structure are necessary to provide a secure key generating algorithm.

In this section, we combine the idea of constructing CCA secure encryption using the lossy TDFs [174] (also discussed in §6.1) and the new idea of TDFs with a two-tier structure. We firstly introduce a new general primitive called *two-tier trapdoor functions* (two-tier TDFs). It is injective TDFs with an additional key generation algorithm. Similar to [174], we extend the two-tier TDFs and define the *lossy two-tier trapdoor functions*. We prove that the two-tier TDFs implies the injective TDFs. We present a black-box construction of a *two-tier encryption* for equivalence relation based on lossy two-tier TDFs.

Finally, we realise the lossy two-tier TDFs based on the hardness of the decisional bilinear Diffie-Hellman (DBDH) problem. We then demonstrate how to construct an

identity-based encryption scheme, an attribute-based encryption scheme and a broadcast encryption scheme based on the new lossy two-tier TDFs.

Related Works. Predicate encryption is a new paradigm to generalise a family of public key encryption. In a predicate encryption scheme, a secret key SK_f corresponds to a predicate $f(\cdot)$ and a ciphertext associates with an attribute I. The secret key SK_f can be used to decrypt σ if and only if $f(I) = 1$. Sahai and Waters [183] proposed the concept of attribute-based encryption (ABE) and presented a particular scheme called the fuzzy IBE. In the fuzzy IBE scheme, the attribute is an identity I and the predicate function f is a set overlap distance with the user identity I'. Goyal *et al.* [103] constructed a key-policy ABE scheme where the predicate function is a monotone access structure. Recently, Katz *et al.* [127] presented a predicate encryption scheme where the predicate function can be disjunctions, polynomial equations or inner products.

The definition of the predicate encryption and our two-tier encryption are similar. However the starting point of these cryptosystems are quite different. For the predicate encryption, the sender may want to define a policy determining who is allowed to recover the encrypted data. For the two-tier encryption, the key generating party wants to distribute keys to different users such that they can decrypt correctly. As a result, the approach to instantiate these cryptosystems are different. For the predicate encryption, different predicate functions are proposed in the literature [183, 103, 127]. In this section, we first consider how the key generation algorithm are protected by complexity assumptions.

The two-tier encryption is more generalised than the predicate encryption in two ways. Firstly, we allow the key generation algorithm to be an interactive protocol between the key generating party and the user. It allows the user to involve in the key generation algorithm, like the certificate-based encryption. Secondly, in the two-tier encryption, the relation R is more general than the predicate function f in the secret key and the attribute I in the ciphertext. In the predicate encryption, the key generating party defines the predicate function f and restricts who can decrypt. Predicate encryption can be captured in the framework of the two-tier encryption, by defining $(f, I) \in R$ if and only if $f(I) = 1$. On the other hand, cryptosystems that are not captured by the predicate encryption can still be captured by two-tier encryption. For example in the ciphertext-policy ABE [24], a secret key corresponds to an attribute I' and a ciphertext associates with a predicate function f'. It is opposite to the setting of the predicate encryption. It can be represented by the two-tier encryption when we define $(I', f') \in R$ if and only if $f'(I') = 1$.

The two-tier encryption can be used for a more fine-grained access control as follows. The encryptor encrypts a message with the attribute I and defines the predicate function f' to restrict who can decrypt. The key generating party gives a secret key to an user with the attribute I' and the predicate function f. We define $((I, f'), (I', f)) \in R$ if and only if $f(I) = 1$ and $f'(I') = 1$.

6.2.1 Two-Tier Trapdoor Functions

In a traditional public key cryptosystem, we observe that the underlying complexity assumptions for the encryption or signatures usually imply the hardness to compute the private key from the public key. For example, the ElGamal encryption is secure if the decisional Diffie-Hellman (DDH) assumption holds. The DDH assumption implies the discrete logarithm assumption, which prevents the adversary to compute the private key x from the public key $y = g^x$. The RSA encryption is secure if the strong RSA assumption holds. The strong RSA assumption implies that factorisation is hard, which prevents the adversary to compute the private key e (or p, q) from the public key ($d = e^{-1} \bmod \phi(n), n = pq$).

For encryption schemes that involve a key generating party, we observe that the underlying complexity assumptions usually imply the hardness to compute the user private key from the user public key, and they further imply the hardness to compute the master private key from the master public key. For example, the Boneh-Franklin IBE [34] (BF-IBE) is secure if the decisional bilinear Diffie-Hellman (DBDH) assumption holds. The DBDH assumption implies the computational Diffie-Hellman (CDH) assumption, which prevents the adversary to compute the user secret key $H(ID)^x$ from the user public key ID and the master public key g^x. Obviously the CDH assumption implies the discrete logarithm assumption, which prevents the adversary to compute the master secret key x from the master public key $y = g^x$. Another example is that Boneh-Boyen IBE in [27] (BB-IBE) is secure if the decisional q-bilinear Diffie-Hellman inversion (decisional q-BDHI) assumption holds. The decisional q-BDHI assumption implies the Diffie-Hellman inversion (DHI) assumption, which prevents the adversary to compute the user secret key $(r, g^{1/ID+x+ry})$ from the user public key ID and the master public key (g^x, g^y). Obviously the DHI assumption implies the discrete logarithm assumption, which prevents the adversary to compute the master secret key (x, y) from the master public key (g^x, g^y). We summarise the above observation in Table 6.5.

In this section we introduce a new class of trapdoor functions called "Two-Tier Trapdoor Functions". These TDFs have a two-tier structure where the second tier

Table 6.5: Intractability assumptions required to satisfy requirements in different encryption schemes. m stands for the message. C stands for the ciphertext. usk and upk stand for the user secret key and the user public key respectively. msk and mpk stand for the master secret key and the master public key respectively.

	ElGamal	RSA	BF-IBE	BB-IBE
Hide m from C	CDH	RSA	DBDH	decisional q-BDHI
Hide usk from upk	DL	Factorisation	CDH	DHI
Hide msk from mpk	-	-	DL	DL

trapdoor can be derived from the first tier trapdoor. The inversion protocol is computed via the second tier trapdoor. The relationship between the two-tier trapdoor function and the traditional trapdoor function is an analogue to the relationship between the complexity assumptions in the previous paragraphs. We then define the lossy and all-but-one two-tier trapdoor functions, which is similar to the definition of lossy and all-but-one trapdoor functions [174] introduced in §3.1.1.

Two-Tier Trapdoor Functions Notations

For generality, let $n_k = \text{poly}(\lambda)$, $n_f = \text{poly}(\lambda)$ denote the input length of the two-tier TDFs in terms of security parameter. A collection of two-tier TDFs is given by a tuple of polynomial time algorithms $(S, K, K^{-1}, F, F^{-1})$:

- $S(1^\lambda)$: The setup algorithm S takes as input the security parameter 1^λ and outputs (s, t_1) where s is a function index and t_1 is the first tier trapdoor.
- $K^{-1}(t_1, \mathsf{i})$: The key generation algorithm K^{-1} takes as input (t_1, i), where t_1 is the first tier trapdoor, i is the information over the domain $\{0,1\}^{n_k}$; and outputs the second tier trapdoor t_2.
- $K(s, t_2, \mathsf{i})$: The key verification algorithm K takes as input (s, t_2, i), where s is the function index, t_2 is the second tier trapdoor and i is the information; and outputs $\top$ for valid or $\bot$ for invalid.
- $F(s, \mathsf{i}, x)$: The function evaluation algorithm F takes as input (s, i, x), where s is the function index, i is the information and x is the input over the domain $\{0,1\}^{n_f}$; and outputs y.
- $F^{-1}(t_2, y)$: The function inversion algorithm F^{-1} takes as input (t_2, y), where t_2 is the second tier trapdoor; and outputs x.

It has the following properties:

1. *Easy to sample, compute and invert with trapdoor:* $S(1^\lambda)$ outputs (s, t_1) where s is a function index and t_1 is the first tier trapdoor. $K^{-1}(t_1, \cdot)$ computes a function $k_{t_1}^{-1}(\cdot)$ over the domain $\{0,1\}^{n_k}$ and $K(s, \cdot, \mathsf{i})$ computes $\top/\bot \leftarrow k_{s,\mathsf{i}}(\cdot)$. $F(s, \mathsf{i}, \cdot)$ computes an injective function $f_{s,\mathsf{i}}(\cdot)$ over the domain $\{0,1\}^{n_f}$ and $F^{-1}(t_2, \cdot)$ computes $f_{t_2}^{-1}(\cdot)$.

2. *Hard to invert k without first tier trapdoor:* for any PPT inverter $\mathcal{I}_k$, the probability that $\mathcal{I}_k(1^\lambda, s)$ outputs (t_2, i) such that $k_{s,\mathsf{i}}(t_2) = \top$ is at most negligible, where the probability is taken over the choice of $(s, t_1) \leftarrow S(1^\lambda)$ and $\mathcal{I}_k$'s randomness.

3. *Hard to invert f without second tier trapdoor:* for any PPT inverter $\mathcal{I}_f$, the probability that $\mathcal{I}_f(1^\lambda, s, \mathsf{i}, f_{s,\mathsf{i}}(x))$ outputs x is at most negligible, where the probability is taken over the choice of $(s, t_1) \leftarrow S(1^\lambda)$, $\mathsf{i} \leftarrow \{0,1\}^{n_k}$, $x \leftarrow \{0,1\}^{n_f}$ and $\mathcal{I}_f$'s randomness.

Lemma 15. *Two-tier trapdoor functions imply (traditional) injective trapdoor functions.*

Proof. Assume we have a two-tier TDF $(S, K, K^{-1}, F, F^{-1})$. We construct an injective TDF (S', F', F'^{-1}) such that:

- $S'(1^\lambda)$: The setup algorithm runs $(s, t_1) \leftarrow S(1^\lambda)$. It picks a random $\mathsf{i} \in \{0,1\}^{n_k}$ and computes $t_2 \leftarrow K^{-1}(t_1, \mathsf{i})$. It outputs $s' = (s, \mathsf{i})$ and $t' = t_2$.
- $F'(s', x')$: The evaluation algorithm parses $s' = (s, \mathsf{i})$ and computes $y' \leftarrow F(s, \mathsf{i}, x')$. It outputs y'.
- $F'^{-1}(t', y')$: The inversion algorithm computes $x' \leftarrow F^{-1}(t', y')$. It outputs x'.

Hence (S', F', F'^{-1}) is easy to sample, compute and invert with trapdoor; and hard to invert without trapdoor. □

Notice that it is possible for the function evaluation $F(s, \mathsf{i}, x)$ to be inverted by a second tier trapdoor t_2 for the information i', where $\mathsf{i}' \subseteq \mathsf{i}$.

Lossy Two-Tier Trapdoor Functions

In this section we define lossy two-tier TDFs. Notice that the first tier function does not need to be lossy.

Define $n_f(\lambda) = \text{poly}(\lambda)$ as the input length of the function F, and $k(\lambda) \leq n_f(\lambda)$ represents the lossiness of the collection. We omit the dependence on λ for convenience.

A collection of (n_f, k)-lossy two-tier TDFs is given by a tuple of polynomial time algorithms $(S_{\text{l2tdf}}, K_{\text{l2tdf}}, K^{-1}_{\text{l2tdf}}, F_{\text{l2tdf}}, F^{-1}_{\text{l2tdf}})$ having the properties below. For notational convenience, define $S_{\text{inj}}(\cdot) = S_{\text{l2tdf}}(\cdot, 1)$ and $S_{\text{loss}}(\cdot) = S_{\text{l2tdf}}(\cdot, 0)$.

1. *Easy to sample an injective function with trapdoor:* $S_{\text{inj}}(1^\lambda)$ outputs (s, t_1) where s is the function index and t_1 is the first tier trapdoor. $F_{\text{l2tdf}}(s, \mathsf{i}, \cdot)$ computes an injective function $f_{s,\mathsf{i}}(\cdot)$ over the domain $\{0,1\}^{n_f}$, and $F^{-1}_{\text{l2tdf}}(t_2, \cdot)$ computes $f^{-1}_{t_2}(\cdot)$ where $t_2 = K^{-1}_{\text{l2tdf}}(t_1, \mathsf{i})$.

2. *Easy to sample a lossy function:* $S_{\text{loss}}(1^\lambda)$ outputs (s', t'_1) where s' is the function index. $K^{-1}_{\text{l2tdf}}(t'_1, \mathsf{i})$ outputs t'_2 such that $K_{\text{l2tdf}}(s', t'_2, \mathsf{i}) = \top$. The behavior of $F^{-1}_{\text{l2tdf}}(t'_2, \cdot)$ is unspecified. $F_{\text{l2tdf}}(s', \mathsf{i}, \cdot)$ computes a function $y = f_{s',\mathsf{i}}(\cdot)$ over the domain $\{0,1\}^{n_f}$, whose image has size at most 2^{n-k}.

3. *Hard to distinguish injective from lossy:* the outputs $(s, t_1) \leftarrow S_{\text{inj}}(1^\lambda)$ and $(s', t'_1) \leftarrow S_{\text{loss}}(1^\lambda)$ are computationally indistinguishable. The outputs from $K^{-1}_{\text{l2tdf}}(t_1, \cdot)$ and $K^{-1}_{\text{l2tdf}}(t'_1, \cdot)$ are also computationally indistinguishable.

All-But-One Two-Tier Trapdoor Functions

Similar to the all-but-one (ABO) trapdoor functions (§3.1.1), we define the ABO version for our two-tier TDFs. In an ABO two-tier collection, each second tier function has several branches. Almost all the branches are injective TDFs (with the same trapdoor value), except for one branch which is lossy. The lossy branch is specified as a parameter to the function sampler, and its value is hidden.

We retain the same notation for n_f, k as above, and also let $\mathcal{B} = \{B_\lambda\}_{\lambda \in \mathbb{N}}$ be a collection of sets whose elements represent the branches. A collection of (n_f, k)-ABO two-tier TDFs with branch collection $\mathcal{B}$ is given by a tuple of polynomial time algorithms $(S_{\text{abo2}}, K_{\text{abo2}}, K^{-1}_{\text{abo2}}, G_{\text{abo2}}, G^{-1}_{\text{abo2}})$ having the properties below:

1. *Easy to sample a trapdoor function with given lossy branch:* for any $b^* \in B_\lambda$, $S_{\text{abo2}}(1^\lambda, b^*)$ outputs (s, t_1), where s is a function index and t_1 is the first tier trapdoor.

   For any $b \in B_\lambda$ distinct from b^*, $G_{\text{abo2}}(s, \mathsf{i}, b, \cdot)$ computes an injective function $g_{s,\mathsf{i},b}(\cdot)$ over the domain $\{0,1\}^{n_f}$, and $G^{-1}_{\text{abo2}}(t_2, b, \cdot)$ computes $g^{-1}_{t_2,b}(\cdot)$.

   Furthermore, $G_{\text{abo2}}(s, \mathsf{i}, b^*, \cdot)$ computes an injective function $g_{s,\mathsf{i},b^*}(\cdot)$ over the domain $\{0,1\}^{n_f}$, whose image has size at most 2^{n-k}.

2. *Lossy branch is hidden:* There are two levels of secrecy for the lossy branch:

   - Level 1: The lossy branch is hidden to the outsider only. For any $b_0^*, b_1^* \in B_\lambda$, the first output s_0^* of $S_{\text{abo2}}(1^\lambda, b_0^*)$ and the first output s_1^* of $S_{\text{abo2}}(1^\lambda, b_1^*)$ are computationally indistinguishable.
   - Level 2: The lossy branch is hidden to the outsider and the insider. For any $b_0^*, b_1^* \in B_\lambda$, the output $(s_0^*, t_{2,0}^*)$ and the output $(s_1^*, t_{2,1}^*)$ are computationally indistinguishable, where for $c = 0/1$:

     $$(s_c^*, t_{1,c}^*) \leftarrow S_{\text{abo2}}(1^\lambda, b_c^*), \quad t_{2,c}^* \leftarrow K_{\text{abo2}}^{-1}(t_{1,c}^*, \mathsf{i}),$$

     and i is any information from the corresponding domain.

Level 2 secrecy of the lossy branch provides a stronger security. However our instantiation only provides level 1 secrecy of the lossy branch.

6.2.2 Realisation of Two-Tier TDF from DBDH

Consider a generator algorithm $\mathcal{G}$ that, on input a security parameter 1^λ, outputs a tuple $(p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g)$, in which p is an independent uniform random λ-bit prime, $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the pairings for cyclic groups of order p and g is a generator of $\mathbb{G}$. Also denote $g_t = \hat{e}(g, g)$ as the generator of $\mathbb{G}_T$.

DBDH Encryption. First we present an additively homomorphic encryption based on the DBDH problem. A secret key is chosen as $(\alpha, \beta) \in_R \mathbb{Z}_p^2$, and the public key is $(u = g^\alpha, v = g^\beta) \in \mathbb{G}^2$. To encrypt an $m \in \mathbb{Z}_p$, choose a $\gamma \in_R \mathbb{Z}_p$ and create the ciphertext $E_{u,v}(m; \gamma) = (g^\gamma, g_t^m \cdot \hat{e}(u, v)^\gamma)$. To decrypt a ciphertext $c = (c_1, c_2)$, we need the key $d = g^{\alpha\beta}$ (not necessarily α and β) such that we can output $D_d(c) = log_{g_t}(c_2/\hat{e}(d, c_1))$; when c encrypts a bit $m \in \{0, 1\}$ (or any small value of m), the discrete logarithm can be computed by enumeration. It is straightforward to prove that this cryptosystem is semantically secure under the DBDH assumption. Note that the encryption is additively homomorphic:

$$E_{u,v}(m; \gamma) \odot E_{u,v}(m'; \gamma') = E_{u,v}(m + m'; \gamma + \gamma'),$$

where $\odot$ denotes coordinate-wise multiplication of ciphertexts. Similarly, for $x \in \mathbb{Z}_p$, $E_{u,v}(m; \gamma)^x = E_{u,v}(mx; \gamma x)$.

Furthermore, the encryption also has the homomorphic scalar addition property, where one can add any scalar value $v \in \mathbb{Z}_p$ to the underlying plaintext without knowing

the public key. Let $c = (c_1, c_2) = E_{u,v}(m; \gamma)$. Then we define $c \boxplus v := (c_1, c_2 \cdot g_t^v) = E_{u,v}(m + v; \gamma)$.

DBDH Encrypting Matrices. For simplicity, denote $n = n_f$. We also define the method to encrypt a matrix $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$. First choose n independent secret/public key pairs $(\alpha_j, \beta_j), (u_j = g^{\alpha_j}, v_j = g^{\beta_j})$ for $j \in [n]$, and n independent exponents $\gamma_i \in_R \mathbb{Z}_p$ for $i \in [n]$. The encryption of $\mathbf{M}$ consists of the matrix $\mathbf{C} = (c_{i,j})$ of ciphertexts $c_{i,j} = E_{u_j,v_j}(m_{i,j}; \gamma_i)$ for all $i, j \in [n]$. The decryption key is the collection of secret keys $d_j = g^{\alpha_j \beta_j}$ for $j \in [n]$. The encrypted matrix can also be expressed as:

$$\mathbf{C}_1 = \begin{pmatrix} g^{\gamma_1} \\ \vdots \\ g^{\gamma_n} \end{pmatrix}, \qquad \mathbf{C}_2 = \begin{pmatrix} \hat{e}(u_1, v_1)^{\gamma_1} \cdot g_t^{m_{1,1}} & \cdots & \hat{e}(u_n, v_n)^{\gamma_1} \cdot g_t^{m_{1,n}} \\ \vdots & \ddots & \vdots \\ \hat{e}(u_1, v_1)^{\gamma_n} \cdot g_t^{m_{n,1}} & \cdots & \hat{e}(u_n, v_n)^{\gamma_n} \cdot g_t^{m_{n,n}} \end{pmatrix}.$$

Lemma 16. *The DBDH matrix encryption scheme above produces indistinguishable ciphertexts under the DBDH assumption.*

Proof. Let $\mathbf{L} = (\ell_{i,j})$, $\mathbf{M} = (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$ be any two arbitrary matrices. We first define a set of experiments $H_0, \ldots, H_{n^2}$. In experiment H_k, the output is a matrix $\mathbf{C} = (c_{i,j})$ chosen in the following way: choose secret/public key pairs $(\alpha_j, \beta_j), (u_j, v_j)$ for $j \in [n]$ and exponents $\gamma_i \in \mathbb{Z}_p$ for $i \in [n]$ as above. Then for the first k pairs $(i, j) \in [n]^2$ (where we order the pairs lexicographically), let $c_{i,j} = E_{u_j,v_j}(\ell_{i,j}; \gamma_i)$. For the remaining $n^2 - k$ pairs (i, j), let $c_{i,j} = E_{u_j,v_j}(m_{i,j}; \gamma_i)$.

Observe that experiment H_0 produces an encryption of the matrix $\mathbf{L}$ and H_{n^2} produces an encryption of the matrix $\mathbf{M}$. Now we show that for every $k \in [n]^2$, experiments H_{k-1} and H_k are computationally indistinguishable.

For any $k \in [n]^2$, let (i^*, j^*) be the lexicographically k-th pair in $[n]^2$. Consider the following simulator algorithm S: the input is a public key $(u^* = g^{\alpha^*}, v^* = g^{\beta^*})$ from the DBDH problem instance $(g^{\alpha^*}, g^{\beta^*}, g^{\gamma^*}, T^*)$ and a ciphertext $c^* = (c_1^*, c_2^*) = (g^{\gamma^*}, T^* g^?)$. Notice that if $T^* = \hat{e}(g, g)^{\alpha^* \beta^* \gamma^*}$, then $c^* = E_{u^*,v^*}(?; \gamma^*)$, which is an encryption of either $\ell_{i,j}$ or $m_{i,j}$.

S produces an encrypted matrix $\mathbf{C} = (c_{i,j})$ in the following way. First, for every $j \neq j^*$ it chooses secret/public keys $(\alpha_j, \beta_j), (u_j, v_j)$ as above, and for every $i \neq i^*$ it chooses random exponents $\gamma_i \in \mathbb{Z}_p$.

- For rows $i \neq i^*$, S encrypts normally: for $i < i^*$ and all $j \in [n]$, let $c_{i,j} = E_{u_j,v_j}(\ell_{i,j}; \gamma_i)$; similarly for $i > i^*$ and all $j \in [n]$, let $c_{i,j} = E_{u_j,v_j}(m_{i,j}; \gamma_i)$.

- For row $i = i^*$, S encrypts using the secret key (α_j, β_j). That is, for column $j < j^*$, let

$$\begin{aligned} c_{i,j} &= (c_1^*, \hat{e}(g, c_1^*)^{\alpha_j \beta_j} \cdot g^{\ell_{i,j}}) \\ &= (g^{\gamma^*}, \hat{e}(g, g)^{\alpha_j \beta_j \gamma^*} \cdot g^{\ell_{i,j}}) \\ &= E_{u_j, v_j}(\ell_{i,j}; \gamma^*), \end{aligned}$$

  and similarly for $j > j^*$ (encrypting $m_{i,j}$).

- Finally, for $i = i^*$ and $j = j^*$, let $c_{i,j} = c^*$.

One can see that S's output is distributed according to either H_{k-1} or H_k, depending on whether c^* was an encryption of $\ell_{i,j}$ or $m_{i,j}$ (respectively). If the adversary can distinguish between H_{k-1} and H_k, S can solve the DBDH problem. □

Lossy Two-Tier Trapdoor Function. Before we describe the algorithms, we first define $\mathsf{H} : \{0,1\}^{n_k} \to \mathbb{G}$ as a family of publicly computable function with input i such that for all $\mathsf{h} \in \mathsf{H}$:

- $\log_g \mathsf{h}(\mathsf{i})$ is not known to the public.

- $\mathsf{h}(\mathsf{i})^{\gamma_\ell}$ is publicly computable, where γ_ℓ is the randomness used to compute the encrypted matrix $\mathbf{C}$ for $\ell \in [n_k]$. (For example, let $\mathsf{i} = \{\mathsf{i}_1, \ldots, \mathsf{i}_{n_k}\} \in \{0,1\}^{n_k}$. We review Waters' hash function

$$\mathsf{h}(\mathsf{i}) = u' \prod_{\ell \in [n_k]} u_\ell^{\mathsf{i}_\ell},$$

  where $u', u_1, \ldots, u_{n_k}$ are in the function index. It also includes $u'^{\gamma_\ell}, u_1^{\gamma_\ell}, \ldots, u_n^{\gamma_\ell}$ in the function index. Therefore $\mathsf{h}(\mathsf{i})^{\gamma_\ell}$ is publicly computable.)

We formally describe the function algorithms as follows:

- *Sampling an injective function.* The injective function generator $S_{\text{inj}}(1^\lambda)$ first selects $(p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the DBDH matrix encryption of the identity $\mathbf{I} \in \mathbb{Z}_p^{n \times n}$. It picks pairwise independent $\mathsf{h}_j \in \mathsf{H}$ for $j \in [n]$. The function index is $s = (p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{C}, \{\mathsf{h}_j\}_{j \in [n]})$. The first tier trapdoor t_1 consists of the decryption keys d_j for $j \in [n]$.

- *Sampling a lossy function.* The lossy function generator $S_{\text{loss}}(1^\lambda)$ first selects $(p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the DBDH matrix encryption of $\mathbf{0} \in$

$\mathbb{Z}_p^{n\times n}$. It picks pairwise independent $\mathsf{h}_j \in \mathsf{H}$ for $j \in [n]$. The function index is $s = (p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{C}, \{\mathsf{h}_j\}_{j\in[n]})$. The first tier trapdoor t_1 consists of d_j for $j \in [n]$.

- *Key generation.* K_{l2tdf}^{-1} takes as input (t_1, i), where $t_1 = \{d_j\}$ for $j \in [n]$ is the first tier trapdoor and $\mathsf{i} \in \{0,1\}^{n_k}$ is the information. It picks a random $r \in \mathbb{Z}_p$ and computes:

$$z_j = d_j \cdot \mathsf{h}_j(\mathsf{i})^r, \qquad \tilde{z} = g^r,$$

for $j \in [n]$. The output is the second tier trapdoor $t_2 = (\{z_j\}_{j\in[n]}, \tilde{z})$.

- *Key verification.* K_{l2tdf} takes as input (s, t_2, i), where s is the function index, $t_2 = (\{z_j\}_{j\in[n]}, \tilde{z})$ is the second tier trapdoor and i is the information. It outputs $\top$ if $\hat{e}(z_j, g) = \hat{e}(u_j, v_j) \cdot \hat{e}(\mathsf{h}_j(\mathsf{i}), \tilde{z})$ for all $j \in [n]$. Otherwise it outputs $\perp$.

- *Evaluation.* F_{l2tdr} takes as input $(s, \mathsf{i}, \mathbf{x})$, where s is the function index, i is the information, and $\mathbf{x} = \{x_1, \ldots, x_n\} \in \{0,1\}^n$ is an n-bit input. It outputs $y = (\{y_j, \tilde{y}_j\}_{j\in[n]})$ where:

$$y_j = \bigodot_{i\in[n]} c_{i,j}^{x_i} = E_{u_j,v_j}((\mathbf{xM})_j; \sum_{i\in[n]} x_i\gamma_i), \quad \tilde{y}_j = \prod_{i\in[n]} (\mathsf{h}_j(\mathsf{i})^{\gamma_i})^{x_i},$$

with $\mathbf{M}$ is either $\mathbf{I}$ from the injective generator or $\mathbf{0}$ from the lossy generator.

- *Inversion.* F_{l2tdf}^{-1} takes as input $y = (\{y_j, \tilde{y}_j\}_{j\in[n]})$ and the trapdoor $t_2 = (\{z_j\}_{j\in[n]}, \tilde{z})$. Denote $y_j = (y_{1,j}, y_{2,j})$. Notice that $y_{1,j}$ are the same for all $j \in [n]$ in our construction. It computes

$$w_j = \frac{y_{2,j} \cdot \hat{e}(\tilde{y}_j, \tilde{z})}{\hat{e}(z_j, y_{1,j})}.$$

It outputs $x_j = \log_{g_t} w_j$ for $j \in [n]$. If x_j is a bit (or any small value) this discrete logarithm may be computed by enumeration.

Lemma 17. *The algorithm described above give a collection of $(n, n - \lg p)$-lossy two-tier TDFs under the DBDH assumption for $(\mathbb{G}, \mathbb{G}_T)$.*

Proof. The invertibility for the injective function via the trapdoor information is shown in the algorithm. The indistinguishability follows by Lemma 16. For lossiness, notice that in F_{loss} for any input $\mathbf{x}$ and i, the output includes $y_j = E_{u_j,v_j}(0; R)$ for some fixed $R = \sum_{i\in[n]} x_i\gamma_i \in \mathbb{Z}_p$ and fixed (u_j, v_j). The output also includes $\tilde{y}_j = \mathsf{h}_j(\mathsf{i})^R$. The number of possible outputs is at most p. Therefore the lossiness is $k \geq n - \lg p$. □

ABO Two-Tier Trapdoor Functions. Our instantiation only achieves level 1 secrecy of the lossy branch. In particular, each user obtains the value of the lossy branch in order to decrypt. Therefore the lossy branch is only hidden to the outsider.

Let the set of branches $B = [q]$, where q is at most the smallest value of p produced by $\mathcal{G}(1^\lambda)$. We formally describe the algorithms as follows:

- *Sampling an ABO function.* The function generator $S_{\text{abo2}}(1^\lambda, b^* \in B)$ first selects $(p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g) \leftarrow \mathcal{G}(1^\lambda)$. Denote $\mathbf{C}$ as the DBDH matrix encryption of the matrix $-b^*\mathbf{I} \in \mathbb{Z}_p^{n\times n}$. It picks pairwise independent $\mathsf{h}_j \in \mathsf{H}$ for $j \in [n]$. The function index is $s = (p, \hat{e}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{C}, \{\mathsf{h}_j\}_{j\in[n]})$. The first tier trapdoor t_1 consists of the decryption keys d_j for $j \in [n]$, along with the lossy branch b^*.

- *Key generation.* K_{abo2}^{-1} takes as input (t_1, i), where $t_1 = (\{d_j\}_{j\in[n]}, b^*)$ is the first tier trapdoor and $\mathsf{i} \in \{0,1\}^{n_k}$ is the information. It picks a random $r \in \mathbb{Z}_p$ and computes:

$$z_j = d_j \cdot \mathsf{h}_j(\mathsf{i})^r, \qquad \tilde{z} = g^r,$$

  for $j \in [n]$. The output is the second tier trapdoor $t_2 = (\{z_j\}_{j\in[n]}, \tilde{z}, b^*)$.

- *Key verification.* K_{abo2} takes as input (s, t_2, i), where s is the function index, $t_2 = (\{z_j\}_{j\in[n]}, \tilde{z})$ is the second tier trapdoor and i is the information. It checks if $\hat{e}(z_j, g) = \hat{e}(u_j, v_j) \cdot \hat{e}(\mathsf{h}_j(\mathsf{i}), \tilde{z})$ for all $j \in [n]$; if not, it outputs $\perp$. It then uses t_2 to check if $\mathbf{C}$ is an matrix encryption of the matrix $-b^*\mathbf{I}$; if not, it outputs $\perp$. Finally, it outputs $\top$.

- *Evaluation.* G_{abo2} takes as input $(s, \mathsf{i}, b, \mathbf{x})$, where s is the function index, i is the information, $b \in B$ is the desired branch, and $\mathbf{x} = \{x_1, \ldots, x_n\} \in \{0,1\}^n$ is an n-bit input. It computes:

$$y_j = E_{u_j,v_j}(((b - b^*)\mathbf{x}\mathbf{I})_j; \sum_{i\in[n]} x_i\gamma_i), \quad \tilde{y}_j = \prod_{i\in[n]} (\mathsf{h}_j(\mathsf{i})^{\gamma_i})^{x_i}.$$

  Notice that y_j can be computed by $\mathbf{x}(\mathbf{C} \boxplus b\mathbf{I})$, where $\boxplus$ is the homomorphic scalar addition. It outputs $y = (\{y_j, \tilde{y}_j\}_{j\in[n]})$.

- *Inversion.* G_{abo2}^{-1} takes as input $(t_2, b, \mathbf{y})$, where the trapdoor $t_2 = (\{z_{i,j}\}_{j\in[n]}, b^*)$, $b \neq b^*$ is the evaluated branch, and $\mathbf{y} = (\{y_j, \tilde{y}_j\}_{j\in[n]})$. Denote $y_j = (y_{1,j}, y_{2,j})$. It computes

$$w_j = \frac{y_{2,j} \cdot \hat{e}(\tilde{y}_j, \tilde{z})}{\hat{e}(z_j, y_{1,j})}.$$

It outputs $x_j = (\log_{g_t} w_j)/(b - b^*)$ for $j \in [n]$. If x_j is a bit (or any small value) this discrete logarithm may be computed by enumeration.

Lemma 18. *The algorithm described above give a collection of $(n, n - \lg p)$-ABO two-tier TDFs with level 1 hidden lossy branch under the DBDH assumption.*

Proof. The invertibility is shown in the algorithm. The level 1 hidden lossy branch property follows by Lemma 16. For lossiness, notice that when $b = b^*$, for any input $\mathbf{x}$ and i, the output includes $y_j = E_{u_j,v_j}(0; R)$ for some fixed $R = \sum_{i\in[n]} x_i\gamma_i \in \mathbb{Z}_p$ and fixed (u_j, v_j). The output also includes $\tilde{y}_j = \mathsf{h}_j(\mathsf{i})^R$. The number of possible outputs is at most p. Therefore the lossiness is $k \geq n - \lg p$. □

6.2.3 Two-Tier Encryption

Here we describe our generic two-tier encryption. It does not explicitly define how the key generating party and the user interact with each other nor how the information is used to generate the keys. It acts as a framework to capture encryption schemes with a two-tier structure. Our aim is to show how a CPA-secure or CCA-secure encryption can be constructed regardless of the underlying K^{-1} construction.

Security Notions

We define the security notion for the two-tier encryption below:

- **Setup:** On input the security parameter 1^λ, output the master public key mpk, the master secret key msk and the relation $\mathcal{R}$.
- **KeyGen**: This is an efficient two-party protocol ($\mathbf{KeyGen}_m$, $\mathbf{KeyGen}_u$). $\mathbf{KeyGen}_m$ takes as input the master secret key msk, and $\mathbf{KeyGen}_u$ takes as input (i, mpk, ask), where i is the information, mpk is the master public key and ask is the auxiliary secret key. $\mathbf{KeyGen}_u$ finally outputs usk if usk is a valid key for i; or outputs $\perp$ otherwise.
- **Encrypt**: The algorithm takes as input (mpk, i', m), where mpk is the master public key, i' is the information and m is the message. It outputs the ciphertext σ.
- **Decrypt**: The algorithm takes as input (usk, σ), where usk is the user secret key and σ is the ciphertext. The user decrypts and obtains the message m or $\perp$ indicates decryption failure.

Correctness. We say that a two-tier encryption is correct if

$$\mathbf{Decrypt}(usk, \mathbf{Encrypt}(mpk, \mathsf{i}', m)) = m,$$

where $usk \leftarrow (\mathbf{KeyGen}_u(\mathsf{i}, mpk, ask) \leftrightarrow \mathbf{KeyGen}_m(msk))$ and $(\mathsf{i}, \mathsf{i}') \in \mathcal{R}$.

Confidentiality. We define the chosen ciphertext security (CCA) for two-tier encryption as in the following game:

1. Setup. The challenger runs $(mpk, msk, \mathcal{R}) \leftarrow \mathbf{Setup}(1^\lambda)$ and gives the master public key mpk and the relation $\mathcal{R}$ to the adversary $\mathcal{A}$.

2. The adversary $\mathcal{A}$ is allowed to query the following oracles adaptively:

   - Key Oracle: on input the information i, the oracle runs the $\mathbf{KeyGen}_m$ protocol with the adversary interactively. Finally the adversary gets a user secret key usk.
   - Decryption Oracle: on input the ciphertext σ and the information i, return the message $m/\perp \leftarrow \mathbf{Decrypt}(usk, \sigma)$, where $usk \leftarrow (\mathbf{KeyGen}_u(\mathsf{i}, mpk, ask) \leftrightarrow \mathbf{KeyGen}_m(msk))$.

3. Challenge. The adversary $\mathcal{A}$ sends two messages m_0^* and m_1^* in the message space and the information i^* to the challenger, with the restriction that $(\mathsf{i}, \mathsf{i}^*) \notin \mathcal{R}$ for all i queried in the Key Oracle. The challenger randomly picks a bit $b \in \{0, 1\}$ and computes $\sigma^* = \mathbf{Encrypt}(mpk, \mathsf{i}^*, m_b^*)$. $\mathcal{S}$ returns σ^* to $\mathcal{A}$.

4. The adversary $\mathcal{A}$ is allowed to query the oracles adaptively, with the restriction that any information i is not allowed to query in the Key Oracle if $(\mathsf{i}, \mathsf{i}^*) \in \mathcal{R}$.

5. Output. Finally $\mathcal{A}$ outputs his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$. A two-tier encryption is (ϵ, t, q_k, q_d)-secure against chosen ciphertext attack if there is no t-time adversary with q_k queries to the key oracle and q_d queries to the decryption oracle has an advantage over ϵ in the game.

We also would like to define two weaker security notions. The first one is the standard "chosen plaintext attack" (CPA) model, where we restrict $q_d = 0$ in our security game. A two-tier encryption is (ϵ, t, q_k)-secure against chosen plaintext attack if there is no t-time adversary with q_k queries to the key oracle has an advantage over ϵ in the game.

The second one is the "outsider security against chosen ciphertext attack", where we restrict $q_k = 0$ in our security game. A two-tier encryption is (ϵ, t, q_d)-secure against outsider, chosen ciphertext attack if there is no t-time adversary with q_d queries to the decryption oracle has an advantage over ϵ in the game.

Primitives from Two-Tier Encryption

We show that for different realisations of the information i in KeyGen, the information i' in $\mathsf{Encrypt}$, the relation $\mathcal{R}$ and the KeyGen function in the two-tier encryption, we can obtain many cryptographic primitives.

Identity-based Encryption [34]. We can obtain identity-based encryption by realising:

- i, i': they are the user identity $\mathsf{id} \in \{0,1\}^{n_f}$.
- $\mathcal{R}$: $(\mathsf{i}, \mathsf{i}') \in \mathcal{R}$ if $\mathsf{i} = \mathsf{i}'$.
- **KeyGen**: $\mathbf{KeyGen}_m$ is the key extraction protocol run by the PKG (private key generator) and the user verifies his identity-based secret key using $\mathbf{KeyGen}_u$. There is no auxiliary secret key in identity-based encryption.

In the security model, the simulation of the key extraction oracle is needed. In some variant, the challenge identity id^* is given by the adversary in advance. This is known as the "selective identity" model.

Certificate-Based Encryption [91]. We can obtain certificate-based encryption by realising:

- i, i': they are (id, upk), where id is the user identity and upk is the user public key.
- $\mathcal{R}$: $(\mathsf{i}, \mathsf{i}') \in \mathcal{R}$ if $\mathsf{i} = \mathsf{i}'$.
- **KeyGen**: The auxiliary secret key ask is the secret key corresponding to upk. In the $\mathbf{KeyGen}_u$ algorithm, the user sends the proof of knowledge of ask with respect to upk to the $\mathbf{KeyGen}_m$ algorithm. $\mathbf{KeyGen}_m$ computes the certificate corresponds to (id, upk) and sends it to $\mathbf{KeyGen}_u$. $\mathbf{KeyGen}_u$ verifies the certificate.

Group Encryption [129]. We can obtain group encryption by realising:

- i, i': they are the user public key upk.
- $\mathcal{R}$: $(\mathsf{i}, \mathsf{i}') \in \mathcal{R}$ if $\mathsf{i} = \mathsf{i}'$.
- **KeyGen**: The auxiliary secret key ask is the secret key corresponding to upk. In the $\mathbf{KeyGen}_u$ algorithm, the user sends the proof of knowledge of ask with respect to upk to the $\mathbf{KeyGen}_m$ algorithm. $\mathbf{KeyGen}_m$ computes the certificate corresponds to upk and publishes it.

For group encryption, it also has an extra **Open** algorithm for the open authority to decrypt; and an interactive proof of knowledge protocol $(\mathcal{P}, \mathcal{V})$ to prove that the encryption uses the public key $(upk, cert)$.

Key-Policy Attribute-Based Encryption [103]. We can obtain key-policy attribute-based encryption by realising:

- i: it is the access structure $\mathcal{T}$.
- i': it is the set of attributes γ.
- $\mathcal{R}$: $(\mathcal{T}, \gamma) \in \mathcal{R}$ if γ satisfies $\mathcal{T}$.
- **KeyGen**: $\mathbf{KeyGen}_m$ is the key extraction protocol run by the PKG (private key generator) and the user verifies his attribute-based secret key using $\mathbf{KeyGen}_u$. There is no auxiliary secret key in key-policy attribute-based encryption.

To the best of the authors' knowledge, the current constructions of key-policy attribute-based encryption in the literature are all in the "selective-set model", where the challenge set of attributes γ^* is given by the adversary in advance.

Ciphertext-Policy Attribute-Based Encryption [24]. We can obtain ciphertext-policy attribute-based encryption by realising:

- i: it is the set of attributes γ.
- i': it is the access structure $\mathcal{T}$.
- $\mathcal{R}$: $(\gamma, \mathcal{T}) \in \mathcal{R}$ if γ satisfies $\mathcal{T}$.

- **KeyGen**: $\mathbf{KeyGen}_m$ is the key extraction protocol run by the PKG (private key generator) and the user verifies his attribute-based secret key using $\mathbf{KeyGen}_u$. There is no auxiliary secret key in ciphertext-policy attribute-based encryption.

To the best of the authors' knowledge, the current constructions of key-policy attribute-based encryption in the literature are all in the "selective-access structure model", where the challenge set of access structure $\mathcal{T}^*$ is given by the adversary in advance.

Predicate Encryption [127]. We can obtain predicate encryption by realising:

- i: it is the predicate function f.
- i': it is the information I.
- $\mathcal{R}$: $(f, I) \in \mathcal{R}$ if and only if $f(I) = 1$.

Broadcast Encryption [77]. We can obtain broadcast encryption by realising:

- i: it is an arbitrary string i.
- i': it is a set of broadcast recipients S.
- $\mathcal{R}$: $(\mathsf{i}, S) \in \mathcal{R}$ if $\mathsf{i} \in S$.
- **KeyGen**: $\mathbf{KeyGen}_m$ is the key extraction protocol run by the PKG (private key generator) and the user verifies his broadcast decryption key using $\mathbf{KeyGen}_u$. There is no auxiliary secret key in broadcast encryption.

6.2.4 Realisation of Two-Tier Encryption

In this section, we give some realisation of our two-tier encryption, using the DBDH instantiation. We first divide the two-tier encryption into two subgroups depending on the relation $\mathcal{R}$: the two-tier encryption for the equivalence relation and the two-tier encryption for the non-equivalence relation. For the equivalence relation, we define the relation $\mathcal{R}$:

$$(a, b) \in \mathcal{R} \text{ if } a = b.$$

The identity-based encryption, the certificate-based encryption and the group encryption belong to the two-tier encryption for the equivalence relation. The attribute-based encryption, the predicate encryption and the broadcast encryption belong to the two-tier encryption for the non-equivalence relation.

Table 6.6: Classification of two-tier encryption. ABE stands for attribute-based encryption. We propose constructions of encryption schemes with * in this section.

Two-tier Encryption	
↙	↘
Equivalence Relation *	Non-equivalence Relation
- *Identity-based Enc* *	- *Broadcast Enc* *
- *Certificate-based Enc*	- *Predicate Enc*
- *Group Enc*	- *Key-Policy ABE* *
	- *Ciphertext-Policy ABE*

We divide the two-tier encryption into two subgroups because we can give a black box construction of the two-tier encryption for the equivalence relation using the lossy two-tier TDFs. We use the DBDH instantiation to construct an identity-based encryption as an example. On the other hand, we also demonstrate how to construct some two-tier cryptosystems for the non-equivalence relation using the DBDH instantiation. The relationship of these schemes are summarised in Table 6.6.

CPA-Secure Construction for Equivalence Relation

Let $(S_{\text{l2tdf}}, K_{\text{l2tdf}}, K^{-1}_{\text{l2tdf}}, F_{\text{l2tdf}}, F^{-1}_{\text{l2tdf}})$ give a collection of (n_f, k)-lossy two-tier TDFs. Let $\mathcal{H}$ be a family of pairwise independent hash function from $\{0,1\}^{n_f}$ to $\{0,1\}^{\ell}$, where $\ell \leq k - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. Our construction has a message space $\{0,1\}^{\ell}$.

- **Setup.** It first generates from the lossy two-tier TDF $(s, t_1) \leftarrow S_{\text{l2tdf}}(1^{\lambda})$. It also chooses a hash function $h \leftarrow \mathcal{H}$. We define the relation $\mathcal{R}$:

  $$(a, b) \in \mathcal{R} \text{ if } a = b.$$

  The master public key consists of the function index and the hash function: $mpk = (s, h)$. The master secret key consists of the trapdoor and the public key: $msk = (t_1, mpk)$.

- **KeyGen.** The interactive algorithms ($\mathbf{KeyGen}_m$, $\mathbf{KeyGen}_u$) takes as input msk and (mpk, i) respectively, where $msk = (t_1, mpk)$ is the master secret key, i is the information and mpk is the master public key.

  - $\mathbf{KeyGen}_u$ sends i to $\mathbf{KeyGen}_m$.
  - $\mathbf{KeyGen}_m$ calculates $t_2 \leftarrow K^{-1}_{\text{l2tdf}}(t_1, \mathsf{i})$ and sends t_2 to $\mathbf{KeyGen}_u$.

– $\mathbf{KeyGen}_u$ outputs $usk = (t_2, mpk)$ if $\top \leftarrow K_{\text{l2tdf}}(s, t_2, \mathsf{i})$, otherwise outputs $\perp$.

- **Encrypt.** The algorithm takes as input (mpk, i, m), where $mpk = (s, h)$ is the master public key, i is the information and $m \in \{0,1\}^\ell$ is the message. It chooses $x \in \{0,1\}^{n_f}$ uniformly at random. It computes the ciphertext $\sigma = (c_1, c_2)$ where

$$c_1 = F_{\text{l2tdf}}(s, \mathsf{i}, x), \qquad c_2 = m \oplus h(x).$$

- **Decrypt.** The algorithm takes as input (usk, σ), where $usk = (t_2, (s, h))$ is the user secret key and $\sigma = (c_1, c_2)$ is the ciphertext. It computes $x = F^{-1}_{\text{l2tdf}}(t_2, c_1)$ and outputs $m = c_2 \oplus h(x)$.

Theorem 41. *The algorithms above are two-tier encryption secure against chosen plaintext attack.*

Proof. We prove the security by first describing two games $Game_1$ and $Game_2$, where $Game_1$ is the chosen plaintext attack game for two-tier encryption. Then we show that $Game_1$ and $Game_2$ are indistinguishable. Finally we will show that the adversary must have negligible advantage in $Game_2$.

When referring to an implementation of these algorithms in a specific game i, we use a subscript i, e.g. Setup_1.

- $Game_1$: It is the same as the IND-CPA game for two-tier encryption. In particular, the simulator $\mathcal{S}$ runs Setup_1 by using the injective function $(s, t_1) \leftarrow S_{\text{inj}}(1^\lambda)$. The $\mathsf{KeyOracle}_1$ can be simulated using t_1.

- $Game_2$: It is different from $Game_1$ that in Setup_2, we replace the injective function $(s, t_1) \leftarrow S_{\text{inj}}(1^\lambda)$ with the lossy one $(s, t_1') \leftarrow S_{\text{loss}}(1^\lambda)$. The $\mathsf{KeyOracle}_2$ can be simulated using t_1', by the definition of the lossy two-tier TDFs.

We now prove two claims which lead to the main theorem.

Claim 13. *$Game_1$ and $Game_2$ are computationally indistinguishable, given the indistinguishability of the injective and lossy functions of the lossy TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, s)$. During Setup, $\mathcal{S}$ runs $h \leftarrow \mathcal{H}$. For Setup_1, s was generated by $S_{\text{inj}}(1^\lambda)$; for Setup_2, s was generated by $S_{\text{loss}}(1^\lambda)$. The master public key is $mpk = (s, h)$. By the indistinguishability of the injective and lossy functions of the lossy two-tier TDF collection, the public keys are computationally

indistinguishable from Setup_1 and Setup_2. Similarly, the $\mathsf{KeyOracle}$ simulation using t_1 or t_1' is indistinguishable by the indistinguishability of the injective and lossy functions. □

Claim 14. *No adversary has more than a negligible advantage in $Game_2$.*

Proof. Fix all the randomness in $Game_2$, except the choice of the hash function h and the randomness x^* used to produce the challenge ciphertext $\sigma^* = (c_1^*, c_2^*)$. We observe that $F_{\text{l2tdf}}(s, \mathsf{i}^*, \cdot) = f_{s,\mathsf{i}^*}(\cdot)$ are lossy functions with image size at most 2^{n-k}. Therefore the random variable $c_1^* = f_{s,\mathsf{i}^*}(x^*)$ can take at most 2^{n-k} values. By lemma 2 and the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, we have that $(c_1^*, h, h(x^*))$ and (c_1^*, h, U_ℓ) are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance. □

□

CCA-Secure Construction for Equivalence Relation

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a strongly secure one-time signature scheme where the public verification keys are in $\{0,1\}^v$. Let $(S_{\text{l2tdf}}, K_{\text{l2tdf}}, K_{\text{l2tdf}}^{-1}, F_{\text{l2tdf}}, F_{\text{l2tdf}}^{-1})$ give a collection of (n_f, k)-lossy two-tier TDFs. Let $(S_{\text{abo2}}, K_{\text{abo2}}, K_{\text{abo2}}^{-1}, G_{\text{abo2}}, G_{\text{abo2}}^{-1})$ give a collection of (n_f, k')-ABO two-tier TDFs having branches $B_\lambda = \{0,1\}^v$. We require that the total lossiness $k + k' \geq n_f + \kappa$ for some $\kappa = \omega(\log n_f)$.

Let $\mathcal{H}$ be a family of pairwise independent hash function from $\{0,1\}^{n_f}$ to $\{0,1\}^\ell$, where $\ell \leq \kappa - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. Our construction has a message space $\{0,1\}^\ell$.

- **Setup.** It first generates from the lossy two-tier TDF $(s, t_1) \leftarrow S_{\text{l2tdf}}(1^\lambda)$. Then it generates from the ABO two-tier TDF having level 2 lossy branch 0^v: $(\bar{s}, \bar{t_1}) \leftarrow S_{\text{abo2}}(1^\lambda, 0^v)$. Finally it chooses a hash function $h \leftarrow \mathcal{H}$. We define the relation $\mathcal{R}$:

  $$(a, b) \in \mathcal{R} \text{ if } a = b.$$

  The master public key is $mpk = (s, \bar{s}, h)$. The master secret key is $msk = (t_1, \bar{t_1}, mpk)$.

- **KeyGen.** The interactive algorithms ($\mathbf{KeyGen}_m$, $\mathbf{KeyGen}_u$) takes as input msk and (mpk, i) respectively, where $msk = (t_1, \bar{t_1}, mpk)$ is the master secret key, i is the information and mpk is the master public key.

  – $\mathbf{KeyGen}_u$ sends i to $\mathbf{KeyGen}_m$.

  - **KeyGen**$_m$ calculates $t_2 \leftarrow K^{-1}_{\text{l2tdf}}(t_1, \mathsf{i})$ and $\bar{t_2} \leftarrow K^{-1}_{\text{abo2}}(\bar{t_1}, \mathsf{i})$. It sends $(t_2, \bar{t_2})$ to **KeyGen**$_u$.
  - **KeyGen**$_u$ outputs $usk = (t_2, \bar{t_2}, \mathsf{i}, mpk)$ if $\top \leftarrow K_{\text{l2tdf}}(s, t_2, \mathsf{i}) \wedge K_{\text{abo2}}(\bar{s}, \bar{t_2}, \mathsf{i})$, otherwise outputs $\perp$.

- **Encrypt.** The algorithm takes as input (mpk, i, m), where $mpk = (s, \bar{s}, h)$ is the master public key, i is the information and $m \in \{0,1\}^\ell$ is the message. It first generates a key pair for the one-time signature scheme $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. It then chooses $x \in \{0,1\}^{n_f}$ uniformly at random. It computes:

$$c_1 = F_{\text{l2tdf}}(s, \mathsf{i}, x),$$
$$c_2 = G_{\text{abo2}}(\bar{s}, \mathsf{i}, vk, x),$$
$$c_3 = m \oplus h(x).$$

  It also computes $c_4 \leftarrow \mathsf{Sign}(sk, (c_1, c_2, c_3))$. The ciphertext is $\sigma = (vk, c_1, c_2, c_3, c_4)$.

- **Decrypt.** The algorithm takes as input (usk, σ), where $usk = (t_2, \bar{t_2}, \mathsf{i}, (s, \bar{s}, h))$ is the user secret key and $\sigma = (vk, c_1, c_2, c_3, c_4)$ is the ciphertext. It first checks that $\mathsf{Ver}(vk, (c_1, c_2, c_3), c_4) = 1$; if not, he outputs $\perp$. It then computes $x = F^{-1}_{\text{l2tdf}}(t_2, c_1)$ and checks that $c_1 = F_{\text{l2tdf}}(s, \mathsf{i}, x)$ and $c_2 = G_{\text{abo2}}(\bar{s}, \mathsf{i}, vk, x)$; if not, it outputs $\perp$. Finally, it outputs $m = c_3 \oplus h(x)$.

Theorem 42. *The algorithms above are two-tier encryption secure against chosen ciphertext attack, if the ABO two-tier TDF has level 2 secrecy.*

Proof. We prove the security by first describing a sequence of games $Game_1$, ..., $Game_5$, where $Game_1$ is the CCA game for two-tier encryption. Then we show that $Game_j$ and $Game_{j+1}$ are indistinguishable for $j = 1, \ldots, 4$. Finally, we will show that the adversary must have negligible advantage in $Game_5$.

When referring to an implementation of these algorithms in a specific game j, we use a subscript j, e.g. Setup_1. Before defining the individual games, we first define some operations that remain the same in all the games. Firstly, Setup_j always first chooses a one-time signature $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}(1^\lambda)$, and then proceeds as we define below. Secondly, in the challenge phase, the simulator $\mathcal{S}$ uses (vk^*, sk^*_σ) to compute the challenge ciphertext.

- $Game_1$: It is the IND-CCA game for two-tier encryption. In particular, he chooses the ABO two-tier TDF lossy branch to be 0^v and decrypts using the trapdoor t_2 from $K^{-1}_{\text{l2tdr}}(t_1, \mathsf{i})$.

- $Game_2$: In this game, the only change with respect to $Game_1$ is in the simulation of the $\mathsf{DecryptionOracle}_2$, which is defined as follows: on input a ciphertext $\sigma = (vk, c_1, c_2, c_3, c_4)$ and the information i, if $vk = vk^*$, the oracle outputs $\perp$. Otherwise, it returns $\mathsf{DecryptionOracle}_1(\sigma, \mathsf{i})$.

- $Game_3$: In this game, the only change with respect to $Game_2$ is in the simulation of the Setup_3, in which the ABO two-tier TDF lossy branch is chosen to be vk^*.

- $Game_4$: In this game, the only change with respect to $Game_3$ is in $\mathsf{DecryptionOracle}_4$, in which the decryption is now done by the ABO trapdoor $\bar{t_2}$ from $K^{-1}_{\text{abo2}}(\bar{t_1}, \mathsf{i})$. During $\mathsf{DecryptionOracle}_4$, he replaces $x = F^{-1}_{\text{l2tdf}}(t_2, c_1)$ with $x = G^{-1}_{\text{abo2}}(\bar{t_2}, vk, c_2)$.

- $Game_5$: In this game, the simulator $\mathcal{S}$ sets $\mathsf{DecryptionOracle}_5 = \mathsf{DecryptionOracle}_4$. In Setup_5, we replace the injective function $(s, t_1) \leftarrow S_{\text{inj}}(1^\lambda) = S_{\text{l2tdf}}(1^\lambda, 1)$ with the lossy one $(s, t_1') \leftarrow S_{\text{loss}}(1^\lambda) = S_{\text{l2tdf}}(1^\lambda, 0)$. In $\mathsf{KeyOracle}_5$, the simulator $\mathcal{S}$ uses t_1' to compute the secret key $t_2 \leftarrow K^{-1}_{\text{l2tdr}}(t_1', \mathsf{i})$.

We now prove a sequence of claims which lead to the main theorem.

Claim 15. *$Game_1$ and $Game_2$ are computationally indistinguishable, given the strong unforgeability of the one-time signature scheme.*

Proof. We observe that $Game_1$ and $Game_2$ are equivalent unless the adversary $\mathcal{A}$ makes a legitimate decryption query of the form $\sigma = (vk^*, c_1, c_2, c_3, c_4)$, where $\sigma \neq \sigma^*$ and $\mathsf{Ver}(vk^*, (c_1, c_2, c_3), c_4) = 1$. Denote this event as F.

Consider a simulator $\mathcal{S}'$ that mounts a chosen message attack against the one-time signature scheme as follows: on input vk generated by $\mathsf{Gen}(1^\lambda)$, it runs Setup by letting $vk^* = vk$ and obtains (mpk, msk). Upon any decryption query from $\mathcal{A}$ of the form $\sigma = (vk^*, c_1, c_2, c_3, c_4)$ such that $\mathsf{Ver}(vk^*, (c_1, c_2, c_3), c_4) = 1$. $\mathcal{S}$ immediately outputs $((c_1, c_2, c_3), c_4)$ as a forgery and returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{S}'$ decrypts using msk and returns the corresponding message to $\mathcal{A}$.

When $\mathcal{A}$ gives $\mathcal{S}'$ two challenge messages m_0^*, m_1^* and the challenge information i^*, $\mathcal{S}'$ chooses $b \in \{0, 1\}$ and creates the challenge ciphertext $\sigma^* = (vk^*, c_1^*, c_2^*, c_3^*, c_4^*)$ by running $\mathbf{Encrypt}(mpk, \mathsf{i}^*, m_b^*)$, except that the signature c_4^* is generated by querying the signing oracle on the message (c_1^*, c_2^*, c_3^*). $\mathcal{S}'$ simulates $Game_2$ perfectly to $\mathcal{A}$.

If the event F happens during the first query phase, then $\mathcal{S}'$ outputs a valid signature without making any signing oracle query, which is a forgery. If F happens during the second query phase on a query $\sigma = (vk^*, c_1, c_2, c_3, c_4)$, then because $c \neq c^*$ we must have either $(c_1, c_2, c_3) \neq (c_1^*, c_2^*, c_3^*)$ or $c_4 \neq c_4^*$. In either case, the output of $\mathcal{S}'$ $((c_1, c_2, c_3), c_4)$

is unequal to its signature query $((c_1^*, c_2^*, c_3^*), c_4^*)$ and hence is a forgery. Since the signature scheme is one-time strongly unforgeable, we conclude that F happens with negligible probability. □

Claim 16. *$Game_2$ and $Game_3$ are computationally indistinguishable, given the level 2 hidden lossy branch property of the ABO two-tier TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, br)$ where $br \in B_\lambda$. During Setup, $\mathcal{S}$ runs $(\bar{s}, \bar{t_1}) \leftarrow S_{\text{abo2}}(1^\lambda, br)$. $\mathcal{S}$ also runs $(s, t_1) \leftarrow S_{\text{inj}}(1^\lambda)$, $h \leftarrow \mathcal{H}$. The public key is $pk = (s, \bar{s}, h)$. For Setup_2, we have $br = 0^v$; for Setup_3, we have $br = vk^*$. By the level 2 hidden lossy branch property, the public keys are computationally indistinguishable from Setup_2 and Setup_3 as well as the key extraction oracle outputs. □

Claim 17. *$Game_3$ and $Game_4$ are perfectly equivalent (if the lossy and ABO two-tier TDF collections are both perfect) or statistically close (if either the lossy or ABO two-tier TDF collections is almost-always).*

Proof. In both $Game_3$ and $Game_4$, (s, t_1) is generated by $S_{\text{inj}}(1^\lambda)$ and $(\bar{s}, \bar{t_1})$ is generated by $S_{\text{abo2}}(1^\lambda, vk^*)$. During the $\mathsf{DecryptionOracle}(\sigma, \mathsf{i})$ query, the simulator $\mathcal{S}$ checks if $c_1 = F_{\text{l2tdf}}(s,\mathsf{i},x) = f_{s,\mathsf{i}}(x)$ and $c_2 = G_{\text{abo2}}(\bar{s},vk,x) = g_{\bar{s},\mathsf{i},vk}(x)$; and outputs $\perp$ if not. Now $f_{s,\mathsf{i}}$ and $g_{\bar{s},\mathsf{i},vk}$ are both injective (except for the case $vk = vk^*$, where the oracle returns $\perp$ immediately). Therefore there is a unique x such that $(c_1, c_2) = (f_{s,\mathsf{i}}(x), g_{\bar{s},\mathsf{i},vk}(x))$. $\mathsf{DecryptionOracle}_3$ finds x by computing $F^{-1}_{\text{l2tdf}}(t_1, c_1)$ and $\mathsf{DecryptionOracle}_4$ finds x by computing $G^{-1}_{\text{abo2}}(\bar{t_1}, vk, c_2)$. Therefore $\mathsf{DecryptionOracle}$ is perfectly equivalent in the two games (or with overwhelming probability, if the trapdoor systems are almost-always). □

Claim 18. *$Game_4$ and $Game_5$ are computationally indistinguishable, given the indistinguishability of the injective and lossy functions of the lossy TDF collection.*

Proof. We describe the simulator algorithm $\mathcal{S}(1^\lambda, s)$. During Setup, $\mathcal{S}$ runs $(\bar{s}, \bar{t_1}) \leftarrow S_{\text{abo2}}(1^\lambda, vk^*)$ and $h \leftarrow \mathcal{H}$. For Setup_4, (s, t_1) was generated by $S_{\text{inj}}(1^\lambda)$; for Setup_5, (s, t_1') was generated by $S_{\text{loss}}(1^\lambda)$. The public key is $pk = (s, \bar{s}, h)$. By the indistinguishability of the injective and lossy functions of the lossy TDF collection, the public keys are computationally indistinguishable from Setup_4 and Setup_5. Similarly, the $\mathsf{KeyOracle}$ simulation using t_1 or t_1' is indistinguishable by the indistinguishability of the injective and lossy functions. □

Claim 19. *No adversary has more than a negligible advantage in $Game_5$.*

Proof. Fix all the randomness in $Game_5$, except the choice of the hash function h and the randomness x^* used to produce the challenge ciphertext C^*. We observe that $F_{\text{l2tdf}}(s, \mathsf{i}^*, \cdot) = f_{s,\mathsf{i}^*}(\cdot)$ and $G_{\text{abo2}}(\bar{s},\mathsf{i}^*,vk^*,\cdot) = g_{\bar{s},\mathsf{i}^*,vk^*}(\cdot)$ are lossy functions with image size at most 2^{n-k} and $2^{n-k'}$ respectively. Therefore the random variable $(c_1^*, c_2^*) = (f_{s,\mathsf{i}^*}(x^*), g_{\bar{s},\mathsf{i}^*,vk^*}(x^*))$ can take at most $2^{2n-(k+k')} \leq 2^{n-\kappa}$ values by the requirement that $k + k' \geq n + \kappa$. By lemma 2 and the hypothesis that $\ell \leq \kappa - 2\lg(1/\epsilon)$, we have that $(c_1^*, c_2^*, h, h(x^*))$ and $(c_1^*, c_2^*, h, U_\ell)$ are within $\epsilon = \mathsf{negl}(\lambda)$ in statistical distance. □

□

If we replace the level 2 secrecy ABO two-tier TDF to a level 1 secrecy ABO two-tier TDF in Setup, then we obtain a two-tier encryption scheme $\mathcal{E}$ which is secure against outsider, chosen ciphertext attack.

Lemma 19. *The two-tier encryption scheme $\mathcal{E}$ is a two-tier encryption secure against outsider, chosen ciphertext attack, if the ABO two-tier TDF has level 1 secrecy.*

The proof of the lemma is almost the same as theorem 42, except that in claim 16, a level 1 secrecy ABO two-tier TDF suffices if the adversary is not allowed to query the key oracle.

Instantiation in IBE. We now consider the information i as the identity string $\mathsf{i} = (\mathsf{id}_1, \ldots, \mathsf{id}_{n_k}) \in \{0,1\}^{n_k}$. We instantiate the function $\mathsf{h}_j(\mathsf{i})$ as the Water's hash function $\mathsf{h}_j(\mathsf{i}) = u'_j \prod_{l=1}^{n_k} u_{l,j}^{\mathsf{id}_l}$ for $j \in [n]$, where $u'_j, u_{1,j}, \ldots, u_{n_k,j}$ are public keys. Then we obtain an (CPA secure) IBE scheme which is similar to the Waters IBE [206]. The only difference is that our message space is in the group $\{0,1\}^\ell$ while Waters' is in the group $\mathbb{G}_T$. As a result, a discrete logarithm computation is needed for each bit of the message. Similar to the Waters IBE, the full CCA security can be achieved via the CHK transformation [54].

Instantiations of Non-Equivalence Relation

Instantiation in Key-Policy Attribute-Based Encryption. We can construct an key-policy attribute-based encryption (KP-ABE) from two-tier encryption using the key system by Goyal *et al.* [103]. We first review some definitions for KP-ABE in [103].

We define $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value.

If num_x is the number of children of a node x and th_x is its threshold value, then $0 < th_x \leq num_x$. Each leaf node x is described by an attribute and a threshold value 1. We denote the parent of the node x in the tree by parent(x). We denote the attribute associated with the leaf node x in the tree by att(x). The children of a node are numbered from 1 to num. The function index(x) returns such a number associated with the node x. Denote by r the root of $\mathcal{T}$ and denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. If a set of attributes γ satisfies $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows: if x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of the node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if att(x)$\in \gamma$.

We also define the Lagrange coefficient $\Delta_{l,S}$ for $l \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$: $\Delta_{l,S}(x) = \prod_{k \in S, k \neq l} \frac{x-k}{l-k}$. The KP-ABE construction follows:

- **Setup.** The algorithm defines a universe $\mathcal{U} = \{1, \ldots, u\}$ (where $u \approx n_1 / \lg p$). It also generates the secret keys $t_1, \ldots, t_u$ and the public parameters $T_{i,l} = g^{\gamma_i t_l}$ for $i \in [n]$, $l \in [u]$, where γ_i is the parameter used to compute $\mathbf{C}$ in the DBDH two-tier TDF. The two-tier master secret key is still $g^{\alpha_j \beta_j}$ and the corresponding public key is $\mathbf{C} = (c_{i,j})$ for $i, j \in [n]^2$.

- **Key Generation.** The algorithm computes the user secret key with respect to the information (which is the access tree $\mathcal{T}$ now) and the secret sharing scheme. First, for $j \in [n]$, choose a polynomial $q_{x,j}$ for each node x in the tree $\mathcal{T}$. Set the degree of the polynomial $d_{x,j} = th_x - 1$. Choose a random $\mu_j \in \mathbb{Z}_p$. Now, for the root node r, set $q_{r,j}(0) = \mu_j$. Other points of the polynomial $q_{r,j}$ are chosen randomly. For any other node x, set $q_{x,j}(0) = q_{\text{parent}(x),j}(\text{index}(x))$. Other points of the polynomial $q_{x,j}$ are chosen randomly.

  After the polynomial is defined, for each leaf node x and for $j \in [n]$, we give the following secret values to the user:

  $$z_j = d_j g^{\mu_j}, \quad D_{x,j} = g^{\frac{q_{x,j}(0)}{t_l}} \quad \text{where } l = \text{att}(x).$$

  The set of the above secret values is the decryption key t_2.[1]

- **Encryption.** To encrypt a message $M \in \{0,1\}^\ell$ under a set of attributes δ,

[1] When we compare our KP-ABE construction with our two-tier encryption framework, observe that we set $\mathsf{h}_j(\mathsf{i}) = \frac{\mu_j t_l}{q_{r,j}(0)}$ and the randomness in our two-tier TDF key generation is $\bar{r} = \frac{q_{r,j}(0)}{t_l}$. Therefore $\mathsf{h}_j(\mathsf{i})^{\bar{r}} = g^{\mu_j}$ and $g^{\bar{r}}$ is distributed by the polynomials $q_{r,j}$.

choose a random $w = (w_1, \ldots, w_n) \in \{0,1\}^n$ and publish the ciphertext as:

$$E = (\delta, \quad \{y_j = \bigodot_{i\in[n]} c_{i,j}^{w_i}\}_{j\in[n]}, \ \{E_l = \prod_{i\in[n]} T_{i,l}^{w_i}\}_{l\in\delta}, \ E' = M \oplus h(w)).$$

- **Decryption.** We first define a recursive algorithm $\mathrm{DecryptNode}(E, t_2, x, j)$ that takes as input the ciphertext $E = (\delta, \{y_j\}_{j\in[n]}, \{E_l\}_{l\in\delta}, E')$, the private key $t_2 = (\{z_j, D_{x,j}\}_{j\in[n]})$, and a node x in the tree $\mathcal{T}$. Let $l = \mathrm{att}(x)$. If the node x is a leaf node, then:

$$\mathrm{DecryptNode}(E, t_2, x, j) = \begin{cases} \hat{e}(D_{x,j}, E_l) & \text{if } l \in \delta, \\ \perp & \text{otherwise.} \end{cases}$$

Notice that $\hat{e}(D_{x,j}, E_l) = \hat{e}(g^{\frac{q_{x,j}(0)}{t_l}}, \prod_{i\in[n]} T_{i,l}^{w_i}) = \hat{e}(g,g)^{q_{x,j}(0)\sum_{i\in[n]}\gamma_i w_i}$.

We now consider the recursive case when x is a non-leaf node. The algorithm $\mathrm{DecryptNode}(E, t_2, x, j)$ proceeds as follows: for all nodes v that are children of x, it calls $\mathrm{DecryptNode}(E, t_2, v, j)$ and stores the output as F_v. Let S_x be an arbitrary th_x-sized set of child nodes v such that $F_v \neq \perp$. If no such set exists, then the node was not satisfied and the function returns $\perp$. Otherwise we let $l = \mathrm{index}(v)$ and $S'(x) = \{\mathrm{index}(v) : v \in S_x\}$, and compute:

$$\begin{aligned} F_{x,j} &= \prod_{v\in S_x} F_v^{\Delta_{l,S'_x}(0)} \\ &= \prod_{v\in S_x} (\hat{e}(g,g)^{q_{v,j}(0)\sum_{i\in[n]}\gamma_i w_i})^{\Delta_{l,S'_x}(0)} \\ &= \prod_{v\in S_x} (\hat{e}(g,g)^{q_{\mathrm{parent}(v),j}(\mathrm{index}(v))\sum_{i\in[n]}\gamma_i w_i})^{\Delta_{l,S'_x}(0)} \\ &= \prod_{v\in S_x} (\hat{e}(g,g)^{q_{x,j}(l)\sum_{i\in[n]}\gamma_i w_i})^{\Delta_{l,S'_x}(0)} \\ &= \hat{e}(g,g)^{q_{x,j}(0)\sum_{i\in[n]}\gamma_i w_i}. \end{aligned}$$

The decryption algorithm simply calls the function on the root of the tree. We observe that $\mathrm{DecryptNode}(E,\ t_2,\ r,\ j) = e(g,g)^{\mu_j\sum_{i\in[n]}\gamma_i w_i}$ if and only if the ciphertext satisfies the tree. Denote $y_j = (y_{1,j}, y_{2,j})$ as in the DBDH matrix encryption. Then for $j \in [n]$, it computes:

$$g_t^{w_j} = \frac{y_{2,j} \cdot F_{r,j}}{\hat{e}(z_j, y_{1,j})}.$$

Therefore $w = \{w_1, \ldots, w_n\}$ can be recovered and hence $M = E' \oplus h(w)$.

The simulation for the private key oracle is similar to the one by Goyal *et al.* [103]. Methods to achieve CCA security from CPA-secure ABE is discussed in [183] and [103]: it may use a simulation sound NIZK, or using the CHK transformation [54].

Instantiation in Broadcast Encryption. We can construct a broadcast encryption from two-tier encryption using the key system of the broadcast encryption by Boneh *et al.* [35]. Denote the information i as the user index $l \in \{1, \ldots, n'\}$. Suppose the master secret key is $\alpha_j \in \mathbb{Z}_p$ for $j \in [n]$. The master public key includes $g^{\alpha_j^l}, h \in \mathbb{G}$ for $l \in \{1, \ldots, n\}$. Then we define the function $\mathsf{h}_j(l) := hg^{-\alpha_j^{n'+1-l}}$. The broadcast encryption construction follows:

- **Setup.** Denote n' as the maximum number of users in the system. The algorithms first picks a random $\alpha_j \in \mathbb{Z}_p$ for $j \in [n]$. It computes the DBDH encryption matrix by setting $\beta_j = \alpha_j^{n'}$. Then it obtains the function index s, including the encrypted matrix $\mathbf{C} = (c_{i,j})$. It picks a random $h \in \mathbb{G}$ as the public parameter. The private key for user l is computed by setting the randomness $r = \alpha_j^l$:

  $$z_{j,l} = g^{\alpha_j \beta_j} \mathsf{h}_j(l)^r = g^{\alpha_j^{n'+1}} (hg^{-\alpha_j^{n'+1-l}})^{\alpha_j^l} = h^{\alpha_j^l},$$

  for $j \in [n]$. Notice that $g^r = g^{\alpha_j^l}$ is also given as the public parameter. Recall that for $i \in [n]$, γ_i is used to compute $\mathbf{C}$. Denote $u'_i = h^{\gamma_i}$, and $u_{i,j,l} = g^{\alpha_j^l \gamma_i}$ for $i, j \in [n], l \in [n']$.

  Therefore the algorithm outputs public key $mpk = \{g,\ h,\ s,\ \{g^{\alpha_j}, \ldots,\ g^{\alpha_j^{n'}}\}_{j\in[n]},\ \{u'_i,\ u_{i,j,1}, \ldots,\ u_{i,j,n'}\}_{i,j\in[n]}\}$.

- **Encrypt.** The algorithm takes as input (mpk, M, S), where mpk is the master public key, $M \in \{0,1\}^\ell$ is the message and S is the set of users. It chooses a random $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and publishes the ciphertext as:

  $$E = (\{y_j = \bigodot_{i\in[n]} c_{i,j}^{x_i}\}_{j\in[n]},\quad \{\tilde{y}_j = \prod_{i\in[n]} (u'_i \prod_{q\in S} (u_{i,j,n'+1-q}))^{x_i}\}_{j\in[n]},\quad C' = M \oplus h(x)).$$

- **Decrypt.** The algorithm takes as input $(E, \{z_{j,l}\}_{j\in[n]})$, where $E = (\{y_j\}_{j\in[n]},\ \{\tilde{y}_j\}_{j\in[n]},\ C')$ is the ciphertext and $\{z_{j,l}\}_{j\in[n]}$ is user l's secret key. It computes:

  $$g_t^{x_j} = \frac{y_{2,j} \cdot \hat{e}(z_{j,l} \prod_{q\in S, q\neq l} g^{\alpha^{n'+1-q+l}}, y_{1,j})}{\hat{e}(g^{\alpha^l}, \tilde{y}_j)}.$$

  Therefore x can be computed by solving the discrete logarithm (by enumeration). Finally, the algorithm outputs $M = C' \oplus h(x)$.

We first show that the decryption algorithm is correct. Denote $R = \sum_{i\in[n]} x_i\gamma_i$ and $y_j = (y_{1,j}, y_{2,j})$ as in the DBDH matrix encryption. Then we have:

$$\begin{aligned}\frac{y_{2,j}\cdot\hat{e}(z_{j,l}\prod_{q\in S,q\neq l} g^{\alpha^{n'+1-q+l}}, y_{1,j})}{\hat{e}(g^{\alpha^l},\tilde{y}_j)} &= \frac{g_t^{x_j}\cdot\hat{e}(g^{\alpha_j}, g^{\alpha_j^{n'}})^R\cdot\hat{e}(h^{\alpha_j^l}\prod_{\substack{q\in S\\ q\neq l}} g^{\alpha^{n'+1-q+l}}, g^R)}{\hat{e}(g^{\alpha^l}, \prod_{i\in[n]}(u_i'\prod_{q\in S}(u_{i,j,n'+1-q}))^{x_i})}\\ &= \frac{g_t^{x_j}\cdot\hat{e}(g,g)^{\alpha_j^{n'+1}R}\cdot\hat{e}(h^{\alpha_j^l}\prod_{\substack{q\in S\\ q\neq l}} g^{\alpha^{n'+1-q+l}}, g^R)}{\hat{e}(g^{\alpha^l}, (h\prod_{q\in S} g^{\alpha_j^{n'+1-q}})^R)}\\ &= g_t^{x_j}.\end{aligned}$$

In the security proof, we also need the value g^{α^l} for $l \in [n]$. Therefore we need to use the stronger assumption called the decisional bilinear Diffie-Hellman exponent (BDHE) problem [30]. This is similar to the CPA-secure broadcast encryption by Boneh *et al.* [35]. As shown in [35], the full CCA security can be achieved via the CHK transformation [54].

6.3 A Formal Treatment of Publish/Subscribe Systems

Publish/subscribe (pub/sub) is an efficient communication infrastructure that supports dynamic, many-to-many data dissemination in a distributed environment. It allows decoupled messaging between: (1) *subscribers*, having *subscriptions* to the interested information, and (2) *publishers*, providing *notifications* for the information they provide. The pub/sub system is illustrated in Figure 6.2. This kind of many-to-many communication is being more and more popular in social networking websites.

All pub/sub technologies use subject or topic names as the loosely coupled link between publishers and subscriber systems. Publishers produce messages on a particular subject or topic name and subscribers receive those messages by registering interest in the subject name either explicitly or through some broader subscription scheme using wildcards. Subscribers and publishers are loosely coupled by a network of *brokers* that route the notifications to the interested subscribers. Pub/sub allows subscribing applications to select messages that these applications receive by topic (as specified by the publishing application) or by content (by specifying filters). The latter is usually referred to as the content-based pub/sub systems (CBPS). Any messages addressed to a topic are delivered to all the topic's subscribers. Every subscriber receives a copy of

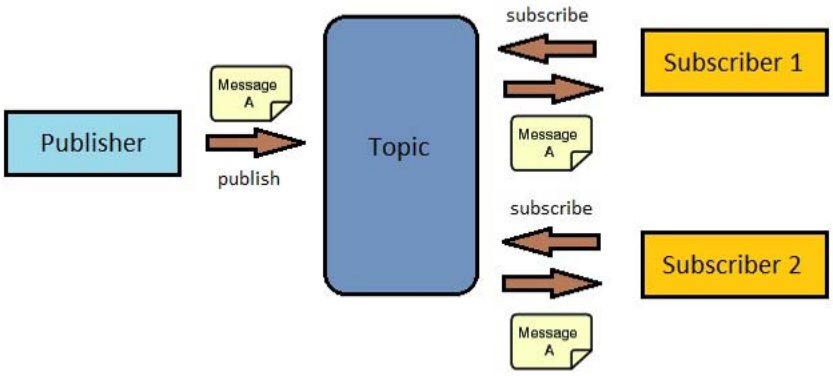


Figure 6.2: Publish/subscribe system.

each message. Information is automatically *pushed* to subscribing applications without them having to *pull* or request it. In short, pub/sub topologies publish messages directly to the bus or network, and these topologies are known as *shared bus*-based solutions.

The current proposed or existing pub/sub systems tend to focus on the performance, scalability and expressiveness issues of the mechanism. Security issues and requirements are firstly addressed by Wang *et al.* [204]. The main issues include authentication, integrity and anonymity, which can usually be achieved by minor modification to the existing approaches. On the other hand, confidentiality is considered more difficult to achieve. Therefore, we refer Wang *et al.*'s work as addressing the security of pub/sub network *partially*.

Our Contributions. Publish/subscribe systems are important to the future social networking services. However, there is no formal security model for the pub/sub systems. Wang *et al.* [204] proposed some security issues and requirements, without defining a formal model. Nikander and Giannis [163] points out the difficulty of modeling pub/sub systems using traditional send/receive paradigm. They only give a general model to reflect the multicast nature of the pub/sub systems, without concerning the security requirements. As a result, most of the existing schemes do not have any security theorem or proof. To the best of the author's knowledge, only Raicius and Rosenblum [179] proposed the security model for confidentiality and they proved the confidentiality of their scheme. A complete security theorem and proof is essential to

analysis the security level of a CBPS protocol. Moreover, a complete security model is needed to identify the security requirements and the attacker's capability. Therefore, in this paper, we propose a formal security model for all security requirements for the CBPS.

Secondly, Wang *et al.* [204] suggested some possible solutions for each security requirements that they proposed. However, it is not clear that if these methods can work together under the *same* threat model. Moreover, some methods are out-of-band solutions and are handled independently of the pub/sub infrastructure. Additionally, most of the existing CBPS schemes enabling confidentiality do not consider authenticity and integrity simultaneously. In this paper, we propose a comprehensive CBPS scheme which fulfills most security requirements concurrently. We prove the security of our scheme under the new security model.

In this section, *for the first time in the literature*, we provide a formal cryptographic treatment of Content-based pub/sub systems (CBPS). Our model can therefore be used to analyze the security of any CBPS system. Furthermore, we also provide a concrete construction of CBPS that satisfies our model. We provide a security proof for our scheme that our scheme is secure under our proposed model.

6.3.1 Publish/Subscribe Systems and their Security Models

In this section we first give the definition of publish/subscribe system. After that, we describe the security requirements and security models for CBPS. It is the *first* comprehensive security model of pub/sub system. Without a formal security model, we cannot analysis the concrete security of any pub/sub system.

Publish/Subscribe Systems

A publish/subscribe system is a system with interactions between four parties:

- Publishers *notify* the brokers for the information they provide in the pub/sub system. They do not know who will obtain the information.
- Subscribers *subscribe* to the interested information. They only receive the information which matches their subscription.
- Brokers *match* the subscription and the notification by the subscribers and the publishers. The broker network will route and forward the packets to the matching subscribers. Sometimes they are further categorized into:

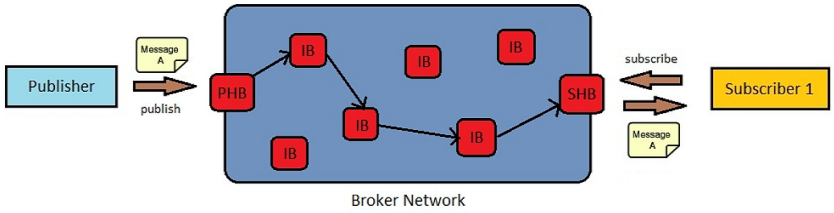


Figure 6.3: Brokers in publish/subscribe system. PHB stands for Publisher Hosting Brokers, SHB stands for Subscriber Hosting Brokers and IB stands for Intermediate Brokers. PHB and SHB are both border brokers.

  - Intermediate brokers. They only route packets within the broker network.
  - Border brokers. They act as a link between the broker network and the other parties in the pub/sub network.
  - Publisher hosting brokers. They are a kind of border brokers that connect between the broker network and the publishers.
  - Subscriber hosting brokers. They are a kind of border brokers that connect between the broker network and the subscribers.

  Different types of brokers are shown in Figure 6.3.

- Managers maintains and coordinates the keys used within the pub/sub system. According to the basic concept of pub/sub system, the publisher does not know the public keys of subscribers. Therefore, the publisher cannot encrypt using subscribers' public keys. In order to provide confidentiality, managers are needed to act as the target of the encryption scheme. If confidentiality is not considered in the pub/sub system, then this party can be ignored. Some papers [179, 140] assume that the publishers and the subscribers have a pre-shared session key. They do not concern about how the managers help to share the session key within the pub/sub system. The managers are called *key distribution center* in [195], *accounting server* in [128] and *secure administrator* in [216].

A content-based publish/subscribe system consists of ten algorithms defined as follows:

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter 1^λ, it outputs the system parameters param and the manager's secret key msk.
- $\mathsf{KeyGen}(\mathsf{param})$: On input the system parameters param, it outputs a secret key and a public key. It can be further divided into publisher's KeyGen_p to generate

publisher secret key psk and public key ppk; subscriber's KeyGen_s to generate subscriber secret key ssk and public key spk; and broker's KeyGen_b to generate broker secret key bsk and public key bpk.

- $\mathsf{RegP}(\mathsf{param}, filter, psk)$, $\mathsf{IssueP}(\mathsf{param}, msk, ppk)$: The interactive algorithms RegP and IssueP are run by the publisher and the manager respectively, where param is the system parameters, $filter$ is the filter set by the publisher, (psk, ppk) is the publisher's secret key, public key pairs and msk is the manager's secret key. RegP first sends the notification to IssueP and IssueP returns a publisher key K_p to RegP.

- $\mathsf{RegS}(\mathsf{param}, sub, ssk)$, $\mathsf{IssueS}(\mathsf{param}, msk, spk)$: The interactive algorithms RegS and IssueS are run by the subscriber and the manager respectively, where param is the system parameters, sub is the subscription by the subscriber, (ssk, spk) is the subscriber's secret key, public key pairs and msk is the manager's secret key. RegS first sends the subscription to IssueS and IssueS returns a subscriber key K_s to RegS.

- $\mathsf{Pub}(\mathsf{param}, m, psk, K_p)$: On input $(\mathsf{param}, m, psk, K_p)$ where param is the system parameters, m is the message, psk is the publisher's secret key and K_p is the publisher key for some $filter$, the publisher outputs a notification n[2] to the broker network.

- $\mathsf{Sub}(\mathsf{param}, sub, ssk, bpk)$: On input $(\mathsf{param}, sub, ssk, bpk)$ where param is the system parameters, sub is the subscription, ssk is the subscriber's secret key and bpk is the (subscriber hosting) broker public key, the subscriber outputs a ciphertext for subscription C_{sub} to the broker network.

- $\mathsf{Match}(\mathsf{param}, n, C_{sub}, bsk)$: On input $(\mathsf{param}, n, C_{sub})$ where param is the system parameters, n is the notification, C_{sub} is the subscription ciphertext and bsk is the (subscriber hosting) broker secret key, the broker first outputs spk for matching the subscription by spk, 0 for not match, $\perp_p$ for invalid notification or $\perp_s$ for invalid subscription.

- $\mathsf{Retrieve}(\mathsf{param}, n, K_s)$: On input (param, n, K_s) where param is the system parameters, n is the notification and K_s is the subscriber key, the subscriber outputs a

[2]We use the term "notification" instead of "ciphertext" for a few reasons. Firstly, part of the information sent by the publisher may not be encrypted, such as the keyword of the message, to facilitate routing. Secondly, the subscription by the subscriber may also be encrypted as another ciphertext in the system.

pair (m, ppk) or $\perp$ for invalid, where m is the message and ppk is the publisher's public key.

We define that the subscription condition sub, the filter condition $filter$ and also the notification n (except the encrypted part) are in the format of $(name, op, value)$. For example, it can be $(price, =, 10)$ or $(age, \geq, 30)$. We define the symbol $n \subseteq_s F$ as the notification n satisfies the boolean relationship in F. In the CBPS system, we require that the publisher's notification n should satisfy the publisher's filter ($n \subseteq_s filter$). Otherwise, the publisher does not have a valid publisher key K_p for n. Similarly, we require that the notification n should satisfy the subscription sub ($n \subseteq_s sub$). Otherwise, the subscriber does not have a valid subscriber key K_s for decryption of n. For example if

$$\begin{aligned} sub &= \langle((code, =, ABC) \text{ AND } (date, <, \text{May 30 2008})) \text{ OR } (code, =, DEF)\rangle, \\ filter &= \langle(code, =, ABC)\rangle, \\ n &= \langle(code, =, ABC) \text{ AND } (price, =, ??) \\ &\qquad \text{AND } (time, =, 14:20) \text{ AND } (date, =, \text{May 12 2008})\rangle, \end{aligned}$$

then $n \subseteq_s sub$ and $n \subseteq_s filter$.

Correctness

The content-based publish/subscribe system has two types of correctness: matching correctness and retrieval correctness. Matching correctness means that an honest broker can always correctly match a valid notification to a subscriber with a valid satisfying subscription. Retrieval correctness means that an honest subscriber can obtain the message if the notification which matches his subscription criteria.

Formally, they are defined as follows:

- Matching Correctness. We require that

$$\mathsf{Match}(\mathsf{param}, n, \mathsf{Sub}(\mathsf{param}, sub, ssk, bpk), bsk) = spk,$$

  where $(ssk, spk) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $(bsk, bpk) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, psk, \mathsf{RegP}(\mathsf{param}, filter))$, and $n \subseteq_s sub$.

- Retrieval Correctness. We require that

$$\mathsf{Retrieve}(\mathsf{param}, n, \mathsf{RegS}(\mathsf{param}, sub, ssk)) = (m, ppk),$$

  where $(psk, ppk) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$, $(ssk, spk) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$, $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, psk, \mathsf{RegP}(\mathsf{param}, filter, psk))$, $n \subseteq_s sub$ and $n \subseteq_s filter$.

Trust Model

There are three types of trust regarding the underlying broker network:

1. *A complete trust to the broker network.* The adversary is not given any information within the broker network.

2. *A trust to the border brokers only.* The adversary can access any information in the intermediate brokers, but not the border brokers.

3. *Untrusted broker network.* The adversary can access any information in the the broker network.

Notice that the trust we discuss here is whether the brokers' keys (if any) and data accessed by the brokers are available to the adversary. We always assume that the brokers honestly route the packets. If not, the subscribers may never receive any packets.

According to different trust level of the broker network, the broker public keys and secret keys can be used as the input in the Pub, Sub or Match protocol.

Confidentiality

The publish/subscribe system has three types of confidentiality: information confidentiality, subscription confidentiality and publisher confidentiality as discussed by Wang *et al.* [204].

In some cases, part of the information can be known by the brokers to facilitate routing (e.g. stock code, update date in a pub-sub stock quote application) while part of the information must be kept secret from untrusted brokers (e.g. stock price, percentage change). Therefore we consider the confidentiality for the secret information instead of the whole document to be sent.

Information confidentiality means that the secret information in the notification should not be known by the untrusted brokers and all outsiders. Subscription confidentiality means that the secret information in the subscription should not be known by the untrusted brokers, publishers, other subscribers and all outsiders. Publisher confidentiality means that the secret information in the notification should not be known by the non-subscribers of that notification. It includes the untrusted brokers and all outsiders. Therefore publisher confidentiality implies information confidentiality.

We note that our model for confidentiality only involves one trusted manager only. In real system, there may be many managers. Our model can be modified for multiple managers. We give the current confidentiality model of one manager for simplicity.

Publisher Confidentiality. We describe the publisher confidentiality for the secret information in the pub/sub network. The indistinguishability game is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the following oracles:

   - IssueS Oracle: On input the subscription sub and the subscriber's public key spk, it runs the $\mathsf{IssueS}(\mathsf{param}, msk, spk)$ protocol and interacts with the $\mathsf{RegS}(\mathsf{param}, sub, \cdot)$ run by $\mathcal{A}$. The oracle outputs the subscriber key K_s from IssueS.
   - IssueP Oracle: On input the publisher's filter $filter$ and the publisher's public key ppk, it runs the $\mathsf{IssueP}(\mathsf{param}, msk, ppk)$ protocol and interacts with the $\mathsf{RegP}(\mathsf{param}, filter, \cdot)$ run by $\mathcal{A}$. The oracle outputs the publisher key K_p from IssueP.
   - Retrieval Oracle: On input (n, sub) where n is the notification and sub is the subscription, the oracle first runs $(ssk, spk) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$. Then the oracle runs both $\mathsf{IssueS}(\mathsf{param}, msk, spk)$ and $\mathsf{RegS}(\mathsf{param}, sub, ssk)$ by itself and obtains K_s. Finally, it outputs the secret information $(m, ppk)/\perp \leftarrow \mathsf{Retrieve}(\mathsf{param}, n, K_s)$.

3. $\mathcal{A}$ sends two messages m_0^* and m_1^* from the message space, a publisher secret key psk^*[3] and a filter $filter^*$ to the challenger. The messages m_0^* and m_1^* are only different in the part of the secret information. The challenger encrypts m_b^* as $n_b^* \leftarrow \mathsf{Pub}(\mathsf{param}, m_b^*, psk^*, \mathsf{RegP}(\mathsf{param}, filter^*, psk^*))$. There should be no subscription sub queried to the IssueS Oracle, such that $n_b^* \subseteq_s sub$. The challenger picks a bit $b \in \{0, 1\}$ and sends the notification n_b^* to $\mathcal{A}$.

[3]Our publisher confidentiality is a strong model since the publisher secret key of the challenge notification is chosen by the adversary. It is possible to define a weaker model where the adversary is only given the publisher public key.

4. $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription *sub* queried to the IssueS Oracle, such that $n_b^* \subseteq_s sub$; and n^* should not be queried to the Retrieval Oracle.

5. Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 41. *A CBPS scheme is $(\epsilon, t, q_s, q_p, q_r)$-publisher confidential against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ϵ in the game.*

Information Confidentiality. Due to the similarity of the definition between information confidentiality and publisher confidentiality, we can define the indistinguishability game of publisher confidentiality same as the one of information confidentiality without query to the IssueS Oracle.

Definition 42. *A CBPS scheme is (ϵ, t, q_p, q_r)-information confidential against chosen ciphertext attack if there is no t-time adversary with q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ϵ in the game.*

Notice that for both publisher and information confidentiality, we say that a system is *selectively* secure if we require the adversary commits to the challenge filter $filter^*$ at the beginning of the game.

Subscription Confidentiality. We describe the confidentiality for the subscription in the pub/sub system. The subscribers may want their subscriptions to be confidential against the broker network[4]. Then the brokers need to match the "encrypted subscriptions" with the notifications[5]. The indistinguishability game is defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.

[4]For example, an investor may not want other people to know which stock price he has subscribed, since it may leak information of which stock he may buy.

[5]Public key Encryption with Keyword Search (PEKS)[33] can be one of the method to solve this dilemma. We will explain in details in the full version of the paper.

2. $\mathcal{A}$ is allowed to query the IssueS Oracle, IssueP Oracle and Retrieval Oracle defined in the publisher confidentiality game.

3. $\mathcal{A}$ sends two subscription sub_0^* and sub_1^* and the subscriber secret key ssk^*, where sub_0^* and sub_1^* have never been queried to the IssueS Oracle. The challenger picks a bit $b \in \{0,1\}$ and computes $C_{sub}^* \leftarrow \mathsf{Sub}(\mathsf{param}, sub_b^*, ssk^*, bpk)$. He sends the resulting ciphertext C_{sub}^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription sub_0^* and sub_1^* are queried to the IssueS Oracle.

5. Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 43. *A CBPS scheme is $(\epsilon, t, q_s, q_p, q_r)$-subscription confidential against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ϵ in the game.*

Notice that all of the above definitions for confidentiality is against chosen ciphertext attack (CCA). If we do not allow any query to the retrieval oracle, then the above confidentiality definition is reduced to against chosen plaintext attack (CPA).

Unforgeability

We describe the unforgeability in the pub/sub system. It provides authentication and integrity for the pub/sub system. Wang *et al.* [204] mentioned that authentication (end-to-end and point-to-point), information integrity, subscription integrity and service integrity are important security requirements for the pub/sub system. We use the standard notion of unforgeability for digital signature to cover the authentication and integrity requirements.

Information Unforgeability. Information unforgeability means that the subscriber believes that the notification is produced by the publisher and is not altered in the broker network. The game for information unforgeability is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(psk, ppk) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk, the secret/public key pairs of the untrusted

brokers and the publisher public key ppk to the adversary $\mathcal{A}$. The publisher's secret key psk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the Pub Oracle: On input the message m and the publisher filter $filter$, the oracle first runs both $\mathsf{IssueP}(\mathsf{param}, msk, ppk)$ and $\mathsf{RegP}(\mathsf{param}, filter, psk)$ by itself and obtains K_p. Then, it outputs the notification $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, psk, K_p)$.

3. $\mathcal{A}$ returns a message m^*, a notification n^* for a subscription sub^*.

$\mathcal{A}$ wins the game if $(m^*, ppk) \leftarrow \mathsf{Retrieve}(\mathsf{param}, n^*, K_s)$, where K_s is the output of $\mathsf{RegS}(\mathsf{param}, sub^*, ssk))$ interacting with $\mathsf{IssueS}(\mathsf{param}, msk, spk)$, n^* was not the output of Pub Oracle query with input m^* and $(ssk, spk) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$.

Definition 44. *A CBPS scheme is (ϵ, t, q_p)-information unforgeable against chosen message attack if there is no t-time adversary winning the above game with probability at least ϵ with q_p queries to the Pub oracle.*

Subscription Unforgeability. Subscription unforgeability means that the broker believes that the subscription is produced by the subscriber and is not altered in the broker network. The game for subscription unforgeability is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(ssk, spk) \leftarrow \mathsf{KeyGen}_s(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk, the secret/public key pairs of the untrusted brokers and the subscriber's public key spk to the adversary $\mathcal{A}$. The subscriber's secret key ssk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the Sub Oracle: On input the subscription sub, the oracle first runs both $\mathsf{IssueS}(\mathsf{param}, msk, spk)$ and $\mathsf{RegS}(\mathsf{param}, sub, ssk)$ by itself and obtains K_s. Then, it outputs the subscription ciphertext $C_{sub} \leftarrow \mathsf{Sub}(\mathsf{param}, sub, ssk, bpk)$.

3. $\mathcal{A}$ returns a subscription ciphertext C^*_{sub} and a notification n^*.

$\mathcal{A}$ wins the game if $spk \leftarrow \mathsf{Match}(\mathsf{param}, n^*, C^*_{sub}, bsk)$ and C^*_{sub} was not the output of Sub Oracle query.

Definition 45. *A CBPS scheme is (ϵ, t, q_s)-subscription unforgeable against chosen message attack if there is no t-time adversary winning the above game with probability at least ϵ with q_s queries to the Sub oracle.*

Service Unforgeability. Service unforgeability means that the broker believes that the notification is produced by the publisher and is not altered in the previous broker network. It ensures that once malicious faults arises at the infrastructure level, it could be detected by the next broker. Information unforgeability provides end-to-end authentication of the publisher, while service unforgeability provides authentication of the publisher to every point in the network. It minimizes the damage by a malicious broker who insert bogus notifications into the pub/sub network. The game for information unforgeability is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$ and $(psk, ppk) \leftarrow \mathsf{KeyGen}_p(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk, the secret/public key pairs of the untrusted brokers and the publisher public key ppk to the adversary $\mathcal{A}$. The publisher's secret key psk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the Pub Oracle: On input the message m and the publisher filter $filter$, the oracle first runs both $\mathsf{IssueP}(\mathsf{param}, msk, ppk)$ and $\mathsf{RegP}(\mathsf{param}, filter, psk)$ by itself and obtains K_p. Then, it outputs the the notification $n \leftarrow \mathsf{Pub}(\mathsf{param}, m, psk, K_p)$.

3. $\mathcal{A}$ returns a notification n^*, a subscription ciphertext C^*_{sub}, a subscriber's public key spk^* and the corresponding subscriber key K^*_s.

$\mathcal{A}$ wins the game if $(m^*, ppk) \leftarrow \mathsf{Retrieve}(\mathsf{param}, n^*, K^*_s)$, $spk^* \leftarrow \mathsf{Match}(\mathsf{param}, n^*, C^*_{sub}, bsk)$ and n^* was not the output of any Pub Oracle query.

Definition 46. *A CBPS scheme is* (ϵ, t, q_p)*-service unforgeable against chosen message attack if there is no* t*-time adversary winning the above game with probability at least* ϵ *with* q_p *queries to the Pub oracle.*

Anonymity

The anonymity in the pub/sub system is different for the publishers and subscribers. We will consider two cases separately. The trust model for anonymity is different from confidentiality and unforgeability, since the border brokers directly connecting to the publisher and subscriber must know who is communicating with them. Therefore the border brokers must be trusted for anonymity. To be more specific, publisher hosting broker is trusted for publisher anonymity; and subscriber hosting broker is trusted for subscriber anonymity.

Publisher Anonymity. The anonymity for the publisher means the publisher remains anonymous when he sends a notification. Only the legitimate subscribers can know the identity of the publisher (for authentication purpose). The publisher anonymity game is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param and the secret/public key pairs of the untrusted brokers to the adversary $\mathcal{A}$. The manager's secret key msk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the IssueS Oracle, IssueP Oracle and Retrieval Oracle defined in the publisher confidentiality game.

3. $\mathcal{A}$ sends two publisher key pairs (ppk_0^*, psk_0^*) and (ppk_1^*, psk_1^*), a message m^* and a filter $filter^*$ to the challenger. The challenger computes $n_b^* \leftarrow \mathsf{Pub}(\mathsf{param}, m^*, psk_b^*, \mathsf{RegP}(\mathsf{param}, filter^*, psk_b^*))$. There should be no subscription sub queried to the IssueS Oracle, such that $n_b^* \subseteq_s sub$, no matter $b = 0$ or 1. He picks a bit $b \in \{0, 1\}$ and sends the notification n_b^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription sub queried to the IssueS Oracle, such that $n_b^* \subseteq_s sub$; and n_b^* should not be queried to the Retrieval Oracle.

5. Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 47. *A CBPS scheme is $(\epsilon, t, q_s, q_p, q_r)$-publisher anonymous against chosen ciphertext attack if there is no t-time adversary with q_s queries to the IssueS oracle, q_p queries to the IssueP oracle and q_r queries to the retrieval oracle has an advantage over ϵ in the game.*

Subscriber Anonymity. The anonymity for the subscriber means that the subscriber remains anonymous when he sends a subscription. The subscriber anonymity game is formally defined as follows:

1. The challenger runs $(\mathsf{param}, msk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(bsk, bpk) \leftarrow \mathsf{KeyGen}_b(\mathsf{param})$. The challenger gives the public parameters param, the manager's secret key msk and the subscriber hosting broker's public key bpk to the adversary $\mathcal{A}$. The subscriber hosting broker's secret key bsk is unknown to $\mathcal{A}$.

2. $\mathcal{A}$ is allowed to query the following oracles: Match Oracle: On input (n, C_{sub}) where n is the notification and C_{sub} is the subscription ciphertext to bpk, it outputs the matching result: spk, 0, $\perp_s$ and/or $\perp_p$ which is the output from $\mathsf{Match}(\mathsf{param}, n, C_{sub}, bsk)$.

3. $\mathcal{A}$ sends two publisher key pairs (spk_0^*, ssk_0^*) and (spk_1^*, ssk_1^*), a subscription sub^* to the challenger. The challenger picks a bit $b \in \{0, 1\}$ and computes $C_{sub}^* \leftarrow \mathsf{Sub}(\mathsf{param}, sub^*, ssk_b^*, bpk)$. He sends the subscription ciphertext C_{sub}^* to $\mathcal{A}$.

4. $\mathcal{A}$ is allowed to query the oracles, with the exception that no subscription ciphertext C_{sub}^* queried to the Match Oracle.

5. Finally $\mathcal{A}$ output his guess b'.

The advantage of $\mathcal{A}$ in the game is $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 48. *A CBPS scheme is (ϵ, t, q_m)-subscriber anonymous against chosen ciphertext attack if there is no t-time adversary with q_m queries to the Match oracle has an advantage over ϵ in the game.*

Notice that subscription anonymity may contradict the accountability requirement in [204]. In commercial pub/sub applications, publishers may want to charge subscribers for the information they provide. If the charge is time basis, subscribers pay when they get the subscription key for a period of time from the manager. Each independent subscription can still be anonymous and our current subscription anonymity model can still be used. However if the charge is per notification basis, subscribers' identities must be revealed for accountability and auditability purposes. The security model need to be changed, such that a publisher needs to know whose subscription matches his notification.

6.3.2 Our Construction

In this section, we will construct some CBPS protocols and we will prove their security against the security model defined in the previous section. We first describe the main idea of the scheme. We review the relevant cryptographic background and then we show the basic construction. Our basic construction satisfies most security requirements in confidentiality and unforgeability. However, our construction does not satisfy all the security requirements mentioned in previous section. Finally, we demonstrate that our basic scheme can be further extended to satisfy the other security requirements.

Main Idea of Our Basic Scheme

The main weakness of the existing CBPS protocols is that they do not have any proof of security. Furthermore, some of them only consider either confidentiality or authenticity. A secure CBPS protocol should have security proofs for both confidentiality and unforgeability. To provide information confidentiality and information unforgeability at the same time, we use an approach commonly used in signcryption schemes. It means that the randomness used in the signature and the encryption are the same. It ensures that the signature and the encryption protocol are run by the same party. An adversary cannot use the ciphertext from a legitimate user and append the adversary's signature to it; nor use the signature from a legitimate user and append a ciphertext computed by the adversary.

To facilitate routing while providing confidentiality in the pub/sub system, we employ the approach that only the part of the document containing the secret information is encrypted. For example, in a pub-sub stock quote application, a publisher (the bank) provides stock quote to subscribers (the bank's customers). The stock price is encrypted while the stock name is not. Therefore the document can be routed to subscribers who are interested in a particular stock.

Unforgeability. The challenge of encrypting the partial document is how the brokers authenticate the document without knowing the plaintext. Refer to the previous example, an obvious solution is to sign on the stock name and the encrypted stock price. However, a signature on the encrypted stock price does not guarantee the authenticity of the stock price. A more complicated solution in [128] is to encrypt the stock price and the signature of the stock price. After that the stock name and the whole ciphertext is signed again.

In this paper, we use a simpler approach by sanitizable signatures [158]. A sanitizable signature scheme allows one to verify a signature even when part to the original message is not known. Therefore, we can use compute a sanitizable signature to the whole document and encrypt the stock price. The brokers only needs to verify the signature for the part of the stock name. For the subscribers, the same signature is verified against the whole document after decryption. By the property of sanitizable signatures, it is difficult to obtain any information on the sanitized messages from the sanitizable signature. Therefore authenticity is preserved while having confidentiality in the (untrusted) broker network. Our scheme uses the sanitizable signatures by Suzuki *et al.* [200].

Confidentiality. The challenge of confidentiality is that how publishers can restrict the access of the secret information. By the loose coupling property of the pub/sub network, publishers do not know who are going to subscribe the notifications. Hence publishers have no public key to encrypt the secret information. Some schemes ([179, 140]) assume that publishers and subscribers share a symmetric key. However, it contradicts the very first assumption of decoupling of publishers and subscribers. These schemes are only suitable for private pub/sub systems over public networks. An internet-scale, dynamic pub/sub network with a universe of publishers and subscribers are unlikely to share a symmetric key. Another possible solution ([175]) for confidentiality is through access control to the broker network. Encryption and decryption is performed by the border brokers and therefore trust must be placed upon them. If the broker network is not trusted, it is difficult for the publisher to find a suitable public key for encryption (brokers are not trusted and subscribers are not known).

Our scheme uses the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) by Waters [207] to solve this problem. In CP-ABE, attributes are used to describe the user's (subscriber's) credentials and the encrypting party (publisher) can encrypt the message according to some formulas over these credentials. Therefore, publishers can encrypts the secret information by some attributes that describe the information. Subscribers can request keys from the manager about the attributes that they are interested in.

The Basic Scheme

We use the sanitizable signature scheme by Suzuki et al. [200] and CP-ABE scheme by Waters [207]. Some input parameters described in §6.3.1 are omitted here when they are not used in the basic scheme. Denote $\oplus$ as the bit-wise XOR function.

- Setup. On input 1^λ, it picks the pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and generators $g, g_1 \in \mathbb{G}$ and collision resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{G}$. It chooses a random exponent $\alpha \in \mathbb{Z}_p$. The manager secret key is g^α. It outputs the system parameters $\mathsf{param} = \{g, g_1, \hat{e}(g,g)^\alpha, \hat{e}, H_1, H_2\}$. Let $(\mathsf{Sig}, \mathsf{Vfy})$ be a secure signature scheme.

- KeyGen. On input the system parameters param, the publisher randomly picks his secret key $x_p \leftarrow \mathbb{Z}_p$. He outputs his public key $y_p = g^{x_p}$. The subscriber randomly picks his secret key $x_s \leftarrow \mathbb{Z}_p$. He outputs his public key $y_s = g^{x_s}$.

- RegP, IssueP. The publisher chooses the filter as an LSSS access structure (M, ρ). We limit ρ to be an injective function, that is an attribute is associated with at most one row of an $\ell \times n$ matrix M. It is the same as the publisher key K_p and therefore he does not need to interact with the manager.

- RegS, IssueS. On input the subscription as a set of attributes S, the subscriber sends it to the manager. the manager with master secret key g^α first chooses a random $t \in \mathbb{Z}_p$. He creates the subscriber key as

$$K = g^\alpha g_1^t, \quad L = g^t, \qquad \forall x \in S \;\; K_x = H_1(x)^t$$

  The manager sends the subscriber key $K_s = (K, L, \{K_x : \forall x \in S\})$ to the subscriber.

- Pub. On input $(\mathsf{param}, m, x_p, K_p)$ where param is the system parameters, m is the message, x_p is the publisher's secret key and $K_p = (M, \rho)$ is the publisher key.

  The publisher first chooses a random vector $\vec{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$. For $i = 1, \ldots, \ell$, he calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the i-th row of M. The publisher then chooses random $r_1, r_2 \in \mathbb{G}$ and $s \in \mathbb{Z}_p$. He computes

$$\begin{aligned}
&w_1 = H_2(m||r_1), \quad && w_2 = H_2(M||\rho||r_2), && C = \hat{e}(g,g)^{\alpha s} \oplus (m, \sigma_1, r_1),\\
&C' = g^s, && C_1 = g_1^{\lambda_1} H_1(\rho(1))^{-s}, && \ldots, \quad C_\ell = g_1^{\lambda_\ell} H_1(\rho(\ell))^{-s},\\
&\sigma_1 = w_1^{x_p}, && w_3 = H_2(w_1||w_2||C'||C||C_1||\ldots||C_\ell), && \sigma_2 = (w_2 w_3)^{x_p}.
\end{aligned}$$

  The notification is published as $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$[6].

- Sub. On input (param, x_s) where param is the system parameters and x_s is the subscriber's secret key, the subscriber signs the subscription attributes S by $\sigma_s = \mathsf{Sig}(x_s, S)$. He sends $C_{sub} = (S, \sigma_s, y_s)$ to the broker network.

- Match. On input $(\mathsf{param}, n, C_{sub})$ where param is the system parameters, $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$ is the notification from publisher y_p and $C_{sub} = (S, \sigma_s, y_s)$ is the subscription, the broker first computes

$$w_2 = H_2(M||\rho||r_2), \quad w_3 = H_2(w_1||w_2||C'||C||C_1||\ldots||C_\ell).$$

[6] (σ_1, σ_2) can be viewed as the sanitizable signature part of the notification. Even without the knowledge of m, the broker can check the validity of σ_2 in the Match algorithm to ensure that the notification is authenticated.

If $\hat{e}(\sigma_2, g_2) \neq \hat{e}(w_2 w_3, y_p)$, the broker outputs $\perp_p$. If $\mathsf{Vfy}(y_s, S, \sigma_s) = 0$, the broker also outputs $\perp_s$.

Otherwise, when S satisfies the access structure (M, ρ), the broker forwards the notification to the subscriber and outputs y_p. If S does not satisfy, the broker outputs 0.

- Retrieve. On input (param, n, K_s) where param is the system parameters, $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$ is the notification and $K_s = (K, L, \{K_x : \forall x \in S\})$ is the subscriber key, suppose that S satisfies the access structure (M, ρ). The subscriber finds the set $I = \{i : \rho(i) \in S\}$. Let $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. Then he computes

$$\frac{\hat{e}(C', K)}{\left(\prod_{i \in I} (\hat{e}(C_i, L)\hat{e}(C', K_{\rho(i)}))^{\omega_i}\right)} = \frac{\hat{e}(g, g)^{\alpha s}\hat{e}(g, g_1)^{st}}{\left(\prod_{i \in I} \hat{e}(g, g_1)^{t\lambda_i \omega_i}\right)} = \hat{e}(g, g)^{\alpha s}.$$

The subscriber obtains $(m, \sigma_1, r_1) = \hat{e}(g, g)^{\alpha s} \oplus C$. He computes

$$w_1 = H_2(m||r_1), \quad w_2 = H_2(M||\rho||r_2), \quad w_3 = H_2(w_1||w_2||C'||C||C_1|| \cdots ||C_\ell).$$

If $\hat{e}(\sigma_1 \sigma_2, g) \neq \hat{e}(w_1 w_2 w_3, y_p)$, the subscriber outputs $\perp$. Otherwise, he outputs a pair (m, ppk), where m is the message and ppk is the publisher's public key.

Security

Our basic scheme has publisher confidentiality, information confidentiality, information unforgeability, subscription unforgeability and service unforgeability.

Theorem 43. *Suppose the (ϵ, t')-decisional q-BDHE assumption holds. Then our basic scheme is (ϵ, t, q_s, q_p)-selectively publisher confidential against chosen plaintext attack in the random oracle model, with a challenge filter (M^*, ρ^*) and*

$$t' = t + (q_s + q_h)O(n^*(\tau_m + \tau_e)),$$

where M^* is of size $\ell^* \times n^*$ and $n^* \leq q$, q_h is the number of query to the H_1 oracle, τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively.

Proof. Assume there is a (ϵ, t, q_s)-adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the decisional q-BDHE problem with probability at least ϵ and in time at most t'.

$\mathcal{B}$ is given the BDHE challenge $(g, g^s, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, T)$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. The adversary $\mathcal{A}$ gives the challenge filter (M^*, ρ^*), where M^* has $n^* \leq q$ columns. $\mathcal{B}$ chooses random $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $\hat{e}(g,g)^{\alpha} = \hat{e}(g^a, g^{a^q})\hat{e}(g,g)^{\alpha'}$. $\mathcal{B}$ also sets $g_1 = g^a$.

Oracle Simulation. By the construction of our scheme, the IssueP oracle is not needed. The H_2 oracle is simulated as normal hash function. The other oracles are simulated as follows.

- H_1 Oracle. On input x, if $H_1(x)$ was already defined in the table, then simply return the same answer as before; otherwise, choose a random value $z_x \in \mathbb{Z}_p$. If there exists an i such that $\rho^*(i) = x$, then let

  $$H_1(x) = g^{z_x} \cdot g^{aM^*_{i,1}} \cdot g^{a^2 M^*_{i,2}} \cdots g^{a^{n^*} M^*_{i,n^*}}.$$

  Otherwise, let $H_1(x) = g^{z_x}$.

  Notice that the oracle outputs are randomly distributed due to the g^{z_x} factor. Since we restrict that ρ^* is an injective function, for any x there is at most one i such that $\rho^*(i) = x$.

- IssueS Oracle. On input the subscription S where S does not satisfy M^*, $\mathcal{B}$ first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have $\vec{w} \cdot M^*_i = 0$.

  $\mathcal{B}$ begins by implicitly defining $t = r + w_1 a^q + w_2 a^{q-1} + \ldots + w_{n^*} a^{q-n^*+1}$ by setting

  $$L = g^r \prod_{i=1}^{n^*} (g^{a^{q+1-i}})^{w_i}, \qquad K = g^{\alpha'} g^{ar} \prod_{i=2}^{n^*} (g^{a^{q+2-i}})^{w_i}.$$

  Next $\mathcal{B}$ have to calculate K_x for all $x \in S$. First, if there is no i such that $\rho^*(i) = x$, $\mathcal{B}$ can simply let $K_x = L^{z_x}$. Otherwise, $\mathcal{B}$ calculates

  $$K_x = L^{z_x} \prod_{j=1}^{n^*} \left(g^r \prod_{\substack{k=1 \\ k \neq j}}^{n^*} (g^{a^{q+1+j-k}})^{w_k} \right)^{M_{i,j}}.$$

  Notice that the term $g^{a^{q+1}}$ term cancels when combined since $M_i \cdot \vec{w} = 0$.

Challenge. The adversary gives two messages m_0^*, m_1^* and a publisher secret key x_p^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β. It chooses random $r_1, r_2, y_2', \ldots, y_n' \in \mathbb{Z}_p$ and creates

$$\begin{aligned} C' &= g^s, \qquad w_1 = H_2(m_\beta^* || r_1), \qquad w_2 = H_2(M^* || \rho^* || r_2), \\ \sigma_1 &= w_1^{x_p}, \qquad C = T \oplus (m_\beta^*, \sigma_1, r_1). \end{aligned}$$

$\mathcal{B}$ implicitly defines $\vec{v} = (s, sa + y_2', sa^2 + y_3', \ldots, sa^{n^*-1} + y_{n^*}')$ by setting

$$C_i = \Big(\prod_{j=2}^{n^*} (g^a)^{M_{i,j}^* y_j'} \Big) (g^s)^{-z_{\rho^*(i)}}.$$

$\mathcal{B}$ computes $w_3 = H_2(w_1 || w_2 || C' || C || C_1 || \ldots || C_\ell)$ and $\sigma_2 = (w_2 w_3)^{x_p}$. The challenge notification is published as $n^* = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M^*, \rho^*)$.

Output Calculation. $\mathcal{A}$ will eventually output a guess β'. If $\beta = \beta'$, $\mathcal{B}$ then outputs his guess that $T = \hat{e}(g,g)^{a^{q+1}s}$; otherwise, $\mathcal{B}$ outputs his guess that T is a random group element in $\mathbb{G}$.

Probability Analysis. Notice that only C and w_1 contains the information about m_β^*. Since H_2 is a collision resistant hash function and r_1 is also encrypted in C, m_β^* is completely hidden from $\mathcal{A}$ when T is a random group element. Therefore $\mathcal{B}$'s probability of solving the problem is the same as $\mathcal{A}$'s advantage in this game.

Time Analysis. In the proof, $\mathcal{A}$ has to compute at most $O(n^*)$ multiplication and exponentiation for every IssueS oracle query and H_1 oracle query.

□

As discussed in §6.3.1, publisher confidentiality implies information confidential. Therefore our basic scheme is also selectively secure for information confidentiality against the chosen plaintext attack. However, we can give a direct proof without selective model and use a weaker assumption.

Theorem 44. *Suppose the (ϵ, t')-DBDH assumption holds. Then our basic scheme is (ϵ, t, q_p)-information confidential against chosen plaintext attack in the random oracle model, with $t' = t + q_h O(\tau_e)$, where q_h is the number of query to the H_1 oracle and τ_e is the time for an exponentiation in $\mathbb{G}$.*

Proof. Assume there is a (ϵ, t, q_p)-adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the decisional BDH problem with probability at least ϵ and in time at most t'.

$\mathcal{B}$ is given the BDH challenge (g, g^s, g^a, g^b, T). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. $\mathcal{B}$ implicitly sets $\alpha = ab$ by letting $\hat{e}(g,g)^{\alpha} = \hat{e}(g^a, g^b)$. $\mathcal{B}$ picks a random $\mu \in \mathbb{Z}_p$ and sets $g_1 = g^{\mu}$. The rest of the system parameters are honestly generated.

Oracle Simulation. By the construction of our scheme, the IssueP oracle is not needed. The H_2 oracle is simulated as normal hash function. The H_1 Oracle is simulated as follows: On input x, if $H_1(x)$ was already defined in the table, then simply return the same answer as before; otherwise, choose a random value $z_x \in \mathbb{Z}_p$ and return $H_1(x) = g^{z_x}$.

Challenge. The adversary gives two messages m_0^*, m_1^*, a challenge filter (M^*, ρ^*) and a publisher secret key x_p^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β. $\mathcal{B}$ chooses random $r_1, r_2, y_2, \ldots, y_n \in \mathbb{Z}_p$ and creates

$$\begin{aligned} C' &= g^s, \qquad w_1 = H_2(m_\beta^* || r_1), \qquad w_2 = H_2(M^* || \rho^* || r_2), \\ \sigma_1 &= w_1^{x_p}, \qquad C = T \oplus (m_\beta^*, \sigma_1, r_1). \end{aligned}$$

$\mathcal{B}$ implicitly defines $\vec{v} = (s, y_2, y_3, \ldots, y_{n^*})$ by setting

$$C_i = \Big(\prod_{j=2}^{n^*} g_1^{M_{i,j}^* y_j} \Big) (g^s)^{\mu(M_{i,1}^* - z_{\rho^*(i)})}.$$

$\mathcal{B}$ computes $w_3 = H_2(w_1 || w_2 || C' || C || C_1 || \ldots || C_\ell)$ and $\sigma_2 = (w_2 w_3)^{x_p}$. The challenge notification is published as $n^* = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M^*, \rho^*)$.

Output Calculation. $\mathcal{A}$ will eventually output a guess β'. If $\beta = \beta'$, $\mathcal{B}$ then outputs his guess that $T = \hat{e}(g,g)^{abs}$; otherwise, $\mathcal{B}$ outputs his guess that T is a random group element in $\mathbb{G}$.

Probability Analysis. Notice that only C and w_1 contains the information about m_β^*. Since H_2 is a collision resistant hash function and r_1 is also encrypted in C, m_β^* is completely hidden from $\mathcal{A}$ when T is a random group element. Therefore $\mathcal{B}$'s probability of solving the problem is the same as $\mathcal{A}$'s advantage in this game.

Time Analysis. In the proof, $\mathcal{A}$ has to compute at most $O(1)$ exponentiation for every H_1 oracle query.

□

Theorem 45. *Suppose the (ϵ', t')-CDH assumption holds. Then our basic scheme is (ϵ, t, q_p)-information unforgeable against chosen message attack in the random oracle model, where*

$$\epsilon' \geq \epsilon(\frac{3}{q_h} - \frac{3}{q_h^2} + \frac{1}{q_h^3}), \qquad t' = t + (q_p + q_h)O(\tau_m + \tau_e),$$

where τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively; and q_h is the number of query to the H_2 oracle.

Proof. Assume there is a (ϵ, t, q_p)-adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the CDH problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given the CDH challenge (g, g^a, g^b). In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates the oracles for $\mathcal{A}$ in the following way.

Setup. $\mathcal{B}$ chooses random $\alpha \in \mathbb{Z}_p$ and gives the master secret key g^α to the adversary $\mathcal{A}$. $\mathcal{B}$ also gives the publisher public key g^a to $\mathcal{A}$. The rest of the system parameters are honestly generated.

Oracle Simulation. The H_1 oracle is simulated as normal hash function. The other oracles are simulated as follows.

- H_2 Oracle. On input x, if $H_2(x)$ was already defined in the table, then simply return the same answer as before; otherwise, choose a random value $b_x \in \mathbb{Z}_p$. With probability $1/q_h$, the oracle outputs

  $$H_2(x) = g^b \cdot g^{b_x},$$

  and stores $(x, b_x, 1)$ in the table. Otherwise, the oracle outputs $H_2(x) = g^{b_x}$ and stores $(x, b_x, 0)$ in the table.

  Notice that the oracle outputs are randomly distributed due to the g^{b_x} factor.

- Pub Oracle. On input the message m and filter (M, ρ), $\mathcal{B}$ first chooses random $r_1, r_2 \in \mathbb{Z}_p$. $\mathcal{B}$ queries the H_2 oracle and if $(m||r_1, \cdot, 1)$ or $(M||\rho||r_2, \cdot, 1)$ appears in the table, $\mathcal{B}$ picks another random number and starts again. Otherwise, let

$w_1 = g^{b_1}$ and $w_2 = g^{b_2}$. For $i = 1, \ldots, \ell$, he calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the i-th row of M. $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_p$ and calculates

$$\begin{aligned} \sigma_1 &= (g^a)^{b_1}, \qquad C' = g^s, \qquad C = \hat{e}(g,g)^{\alpha s} \oplus (m, \sigma_1, r_1), \\ C_1 &= g_1^{\lambda_1} H_1(\rho(1))^{-s}, \qquad \ldots, \qquad C_\ell = g_1^{\lambda_\ell} H_1(\rho(\ell))^{-s}. \end{aligned}$$

$\mathcal{B}$ queries the H_2 oracle and if $(w_1||w_2||C'||C||C_1||\ldots||C_\ell, \cdot, 1)$ appears in the table, $\mathcal{B}$ picks another random number s and starts again. Otherwise, let $w_3 = g^{b_3}$. Then $\mathcal{B}$ calculates

$$\sigma_2 = (g^a)^{b_2+b_3}.$$

The oracle outputs the notification $(C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho)$.

Output Calculation. $\mathcal{A}$ will eventually output a notification $n^* = (C'^*, r_2^*, w_1^*, \sigma_2^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$. We require that n^* is not the output from the Pub oracle. $\mathcal{B}$ computes

$$\begin{aligned} &(m^*, \sigma_1^*, r_1^*) = C^* \oplus \hat{e}(g^\alpha, C'^*), \qquad w_2^* = H_2(M^*||\rho^*||r_2^*), \\ &w_3^* = H_2(w_1^*||w_2^*||C'^*||C^*||C_1^*||\ldots||C_\ell^*). \end{aligned}$$

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is not computed in the Pub oracle, then $\mathcal{B}$ searches the H_2 table for $w_1^*.w_2^*$ and w_3^*. If there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$, $\mathcal{B}$ declares failure and exits. WLOG, assume $w_1^* = g^b \cdot g^{b_1}$, $w_2^* = g^{b_2}$ and $w_3^* = g^{b_3}$. If $\mathcal{A}$ outputs a valid publication, then it satisfies:

$$\begin{aligned} \hat{e}(\sigma_1^*\sigma_2^*, g) &= \hat{e}(w_1^* w_2^* w_3^*, g^a) \\ &= \hat{e}(g^b \cdot g^{b_1+b_2+b_3}, g^a). \end{aligned}$$

Then $\mathcal{B}$ can output $\sigma_1^*\sigma_2^*/(g^a)^{b_1+b_2+b_3}$ as the solution to the CDH problem.

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is computed in the Pub oracle, it means that $(C'^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$ are different from the oracle output. Since

$$w_2^* = H_2(M^*||\rho^*||r_2^*), \qquad w_3^* = H_2(w_1^*||w_2^*||C'^*||C^*||C_1^*||\ldots||C_\ell^*),$$

and H_2^* is collision resistant, the probability of this happening is negligible.

Probability Analysis. $\mathcal{B}$ aborts only when there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$. For each time, it happens with probability $1 - 1/q_h$. Therefore

the probability that $\mathcal{B}$ does not abort is greater than $\frac{3}{q_h} - \frac{3}{q_h^2} + \frac{1}{q_h^3}$.

Time Analysis. In the proof, $\mathcal{A}$ has to compute at most $O(1)$ multiplication and exponentiation for every Pub oracle query and H_1 oracle query.

□

Theorem 46. *Suppose that (*Sig, Vfy*) is EUF-CMA secure. Then no poly-time adversary can break the subscription unforgeability.*

Theorem 46 is straightforward by the construction of our basic scheme.

Theorem 47. *Suppose the (ϵ', t')-CDH assumption holds. Then our basic scheme is (ϵ, t, q_p)-service unforgeable against chosen message attack in the random oracle model, where*

$$\epsilon' \geq \epsilon(\frac{2}{q_h} - \frac{1}{q_h^2}), \qquad t' = t + (q_p + q_h)O(\tau_m + \tau_e),$$

where τ_m and τ_e are the time for a multiplication and an exponentiation in $\mathbb{G}$, respectively; and q_h is the number of query to the H_2 oracle.

Proof. The setup and the oracle simulations are the same as information unforgeability. Therefore it is omitted for simplicity.

Output Calculation. $\mathcal{A}$ will eventually output a notification $n^* = (C'^*, r_2^*, w_1^*, \sigma_2^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$. We require that n^* is not the output from the Pub oracle. $\mathcal{B}$ computes

$$(m^*, \sigma_1^*, r_1^*) = C^* \oplus \hat{e}(g^\alpha, C'^*), \qquad w_2^* = H_2(M^*||\rho^*||r_2^*),$$
$$w_3^* = H_2(w_1^*||w_2^*||C'^*||C^*||C_1^*||\ldots||C_\ell^*)$$

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is not computed in the Pub oracle, then $\mathcal{B}$ searches the H_2 table for $w_1^*.w_2^*$ and w_3^*. If there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 2, 3$, $\mathcal{B}$ declares failure and exits. WLOG, assume $w_2^* = g^b \cdot g^{b_2}$ and $w_3^* = g^{b_3}$. If $\mathcal{A}$ outputs a valid publication, then it satisfies:

$$\begin{aligned} \hat{e}(\sigma_2^*, g) &= \hat{e}(w_2^* w_3^*, g^a) \\ &= \hat{e}(g^b \cdot g^{b_2+b_3}, g^a). \end{aligned}$$

Then $\mathcal{B}$ can output $\sigma_2^*/(g^a)^{b_2+b_3}$ as the solution to the CDH problem.

If the tuple $(\sigma_1^*, \sigma_2^*, w_1^*, w_2^*, w_3^*, m^*, C'^*, r_1^*, r_2^*)$ is computed in the Pub oracle, it means that $(C'^*, C^*, C_1^*, \ldots, C_\ell^*, M^*, \rho^*)$ are different from the oracle output. Since

$$w_2^* = H_2(M^*||\rho^*||r_2^*), \qquad w_3^* = H_2(w_1^*||w_2^*||C'^*||C^*||C_1^*||\ldots||C_\ell^*),$$

and H_2^* is collision resistant, the probability of this happening is negligible.

Probability Analysis. $\mathcal{B}$ aborts only when there is no entry that $(w_i^*, b_i, 1)$ appears in the table for $i = 1, 2, 3$. For each time, it happens with probability $1 - 1/q_h$. Therefore the probability that $\mathcal{B}$ does not abort is greater than $\frac{2}{q_h} - \frac{1}{q_h^2}$.

Time Analysis. In the proof, $\mathcal{A}$ has to compute at most $O(1)$ multiplication and exponentiation for every Pub oracle query and H_1 oracle query.

□

Extensions to Our Basic Scheme

Subscriber Confidentiality. Our basic scheme does not provide subscriber confidentiality. The subscriber needs to send his subscription (set of attributes) in plaintext to the brokers. We propose an extension for subscriber confidentiality, if the subscriber knows the public key of the publisher in advance.

Public key Encryption with Keyword Search (PEKS) [33] is a public key encryption scheme that encrypt the message as well as the keyword. The decryptor calculates a trapdoor function using his secret key and the keyword that he is interested in. He forwards the trapdoor information to a third party. The third party can compare the trapdoor and the encrypted keyword to see if they match or not. In PEKS, the message and the keyword are encrypted by the sender. However in the pub/sub system, the message (notification) is encrypted by the publisher and the keyword sent by the subscriber should be encrypted (for subscriber confidentiality). Therefore some modifications are needed to apply the PEKS into our scheme.

Suppose we have a PEKS scheme with the following algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: On input a security parameter 1^λ, it outputs a public and private key pair pk and sk.
- $\mathsf{PEKS}(pk, w)$: On input the public key pk and a keyword w, it outputs a searchable encryption S.
- $\mathsf{Trapdoor}(sk, w)$: On input the private key sk and a keyword w, it outputs a trapdoor T_w.
- $\mathsf{Test}(T_w, S)$: On input a trapdoor T_w and a searchable encryption S, it tests whether S encrypts w. It outputs 1 for "accept" or 0 for "reject".

A PEKS scheme must satisfy that

$$\Pr[\mathsf{Test}(\mathsf{Trapdoor}(sk, w), \mathsf{PEKS}(pk, w)) = 1] \;=\; 1,$$

where the probability is taken over the choice of $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and the coins of all the above algorithms. The security requirements of PEKS include consistency (perfect, computational or statistical) and privacy.

Assume the subscriber knows the public key of the publisher in advance. We have the following modified CBPS protocol:

- Pub. On input $(\mathsf{param}, m, x_p, K_p)$ where param is the system parameters, m is the message, x_p is the publisher's secret key and $K_p = (M, \rho)$ is the publisher key. The publisher computes $(C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell,\ M, \rho)$ as in the basic scheme. He also calculates $T_{\rho(i)} = \mathsf{Trapdoor}(x_p, \rho(i))$ for all i. The notification is published as $n = (C', r_2, w_1, \sigma_2, C, C_1, \ldots, C_\ell, M, \rho, \{T_{\rho(i)}\})$.

- Sub. On input the set of attributes S and the publisher public key y_p, it outputs $S_x = \mathsf{PEKS}(y_p, x)$ for all $x \in S$.

- Match. On input $(\mathsf{param}, n, \{S_i\})$ where param is the system parameters, $n = (C', r_2, w_1, \sigma_2, C, C_1,\ \ldots,\ C_\ell, M, \rho, \{T_{\rho(i)}\})$ is the notification from publisher y_p and S_x is the set of subscription ciphertext, the broker first computes

  $$w_2 = H_2(M||\rho||r_2), \qquad w_3 = H_2(w_1||w_2||C'||C||C_1||\ldots||C_\ell).$$

  If $\hat{e}(\sigma_2, g_2) \neq \hat{e}(w_2 w_3, y_p)$, the broker outputs $\perp$. Otherwise, the broker forwards the notification to the subscriber when there exist some i such that $\mathsf{Test}(T_{\rho(i)}, S_x) = 1$ for all x.

It is straightforward that the subscriber confidentiality is provided by the privacy of the PEKS scheme. If the PEKS scheme has perfect consistency, then the brokers in our modified pub/sub scheme can always forward the notification correctly. However, the existing PEKS schemes are all secure against CPA only.

CCA Security for Publisher Confidentiality. The ciphertext-policy attribute-based encryption (CP-ABE) from [207] allows delegation of secret keys by deleting attributes from a key. For example, if a user has a key for the attribute set {"Professor", "CS Department"}, he can delegate a key for the attribute "Professor" only. As suggested in [207], CCA-security for publication confidentiality can be realized in the standard model by using the techniques of Canetti, Halevi and Katz [53] to the delegation system.

Subscriber Anonymity. Subscriber anonymity can be achieved be a few ways. The subscriber can compute a ring signature instead of a standard signature in the Sub protocol. The security model for subscriber anonymity has to be changed to the one similar to the anonymity of ring signatures.

The subscriber may also compute a designated verifier signature instead of a standard signature in the Sub protocol, with the subscriber hosting broker as the designated verifier. The security model for subscriber anonymity has to be changed to the one similar to the non-transferability of designated verifier signatures.

Since both methods involve non-trivial changes to the existing security model, we omit the detailed construction for simplicity.

Denial-of-service Attack. Denial-of-service attack is a significant risk for pub/sub system like other communication systems. The damage may be even more severe when the attacker publishes some fake notifications and they are spread by (honest) brokers to multiple subscribers. By the broadcasting property of the pub/sub system, the damage may be amplified.d by (honest) brokers to multiple subscribers. By the broadcasting property of the pub/sub system, the damage may be amplified.

Although it is very difficult to prevent the denial-of-service attack in general, we can try to minimize the damage of such attack. Besides the out-of-band solution suggested in [204], we provide a possible solution in our security model. The service unforgeability model requires the honest brokers to check the validity of the notification before forwarding it to the next brokers. Malicious publications are dropped by the broker who firstly encounters them. Since our basic scheme has service unforgeability, it has better protection against the denial-of-service attack by malicious publications.

Furthermore, malicious subscriptions are dropped by the subscriber hosting broker who firstly encounters them. This is captured by the subscription unforgeability model. Since our basic scheme has subscription unforgeability, it has better protection against the denial-of-service attack by malicious subscriptions.

Weaker Assumption. Methods to remove the restriction of the injectiveness in ρ function and to use the weaker decisional BDH assumption can be found in [207].

6.3.3 Related Works

In this section we compare our basic CBPS scheme and the extension with the existing CBPS schemes providing confidentiality. The result of the comparison can be found in

Table 6.7: Comparison of pub/sub schemes providing confidentiality. For confidentiality (Conf), I stands for information confidentiality, S stands for subscription confidentiality and P stands for publisher confidentiality. For unforgeability (Unf), I stands for information unforgeability, S stands for subscription unforgeability and V stands for service unforgeability. For anonymity (Anon), S stands for subscriber anonymity. A small letter means that it is secure in a weaker security model in the original cited paper only. BB. stands for border brokers. For * in the table, it will be explained in §6.3.3.

Scheme	Conf	Unf	Anon	Pre-Shared Key	Proof Proof	Trust
Li et al. [140]	i, s	-	-	Yes	No	No
Khurana [128]	I	I, V	-	No	No	*
Zhao and Sturman [216]	I	i, s	-	No	No	BB.
Raicius and Rosenblum [179]	I, S	i, s	-	Yes	Yes	No
Srivatsa and Liu [195]	i, p, s	-	-	No	No	No
Pesonen et al. [175]	I, S	*	-	No	No	BB.
Zhang et al. [215]	i, s	-	-	No	No	No
Our Basic Scheme	I, P	I, S, V	-	No	Yes	No
Our Extension (in §6.3.2)	I, P, S	I, S, V	s	No	Yes	No

Table 6.7.

The scheme of Li, Lu and Shi [140] and Srivatsa and Liu [195] use prefix-preserving tree structure for information and subscription confidentiality as well as efficient matching. However, if the adversary have a large number of matching notification and subscription pairs, then the adversary may obtain some information about the prefix in the notification and subscription. Therefore they are only secure if the adversary knows a few notification and subscription pairs. They are not secure in our security model.

Khurana [128] proposed a CBPS scheme with a threshold key sharing scheme such that t out of n managers are responsible to generates keys for subscribers and publishers. It reduces the trust to a single manager. However, the group of n managers must help to calculate the notification when the notification travels from a broker to another. It greatly increases the workload of the managers.

Zhao and Sturman [216] placed a complete trust to the border brokers in their CBPS scheme. Encryption is performed between border brokers. Publishers and subscribers access pub/sub system through the access control list. Information confidentiality and authenticity is protected by this access control. The scheme is not secure in our unforgeability model.

Pesonen, Eyers and Bacon [175] also placed a trust to the border brokers in their CBPS scheme. Publishers and subscribers access pub/sub system through the access control list. Information and subscription confidentiality are protected by this access control. Since authenticated encryption is used, integrity is also protected. However, the scheme is not secure in our unforgeability model.

Raicius and Rosenblum [179] proposed the first CBPS scheme with proof of information and subscription confidentiality. It comes with the cost of publishers and subscribers having a pre-shared key. The unforgeability of the scheme is also protected by this pre-shared key, since encryption cannot be performed without the symmetric key. The scheme is not secure in our unforgeability model.

Zhang et al. [215] proposed a CBPS scheme using a new mechanism called information foiling. The publishers and subscribers generate a set of fake messages to hide the authentic message. Their new algorithm does not fit into our model since their confidentiality is in a probabilistic sense.

In this section, we introduced the *first* security model for all security requirements of CBPS. We proposed a new CBPS scheme that fulfills most of the security requirements concurrently. We proved its security according to our new model. The open problem includes constructing a CBPS scheme that also satisfies the publisher and subscriber anonymity requirement.

Chapter 7

Conclusion and Future Work

This thesis contributed to various uses of pairings in cryptography. We used pairings to construct efficient and secure digital signature schemes. A large part of the thesis was dedicated to the area of identity-based cryptography, and its extension to two-tier encryption.

In Chapter 4, we showed that pairings can be used to construct more efficient and more secure digital signatures. We first used pairings to construct two important variants of digital signatures: *undeniable signatures* and *sanitisable signatures*. We proposed the first convertible undeniable signatures which is secure under simple static assumptions in the standard model. We formalised the security model for sanitisable signatures. We provided a first concrete construction which is secure under the CDH assumption in the standard model. Furthermore, we proposed the notion of *concinnous signatures* for the fair exchange of digital signatures without trusted third party. In Chapter 4, we showed how to use pairings to construct different types of digital signatures. It demonstrated the importance of pairings in cryptography.

In Chapter 5, we explored the key escrow problem in identity-based cryptography. We classified the existing solutions to the key escrow problem in the literature and discussed their limitations. After that, we showed that the key escrow problem can be prevented in identity-based signatures. We proposed the notion of *escrow-free identity-based signatures*. We gave a generic construction and a concrete construction for it. On the other hand, we demonstrated the impossibility of ideal escrow-free identity-based encryption. Based on this result, we developed the notion of *fully anonymous identity-based encryption* as the preventive measure of the key escrow problem. We gave two different constructions of fully anonymous identity-based encryption. Furthermore, we presented a new construction of black-box accountable authority identity-based encryption, which provided a blaming mechanism to the key escrow problem. We believed that Chapter 5 showed a comprehensive solution to the key escrow problem in identity-based cryptography using pairings.

In Chapter 6, we used pairings to construct new cryptographic primitives and cryptosystems. Firstly, we instantiated the general primitive *lossy trapdoor function* using pairings, and proposed some new applications of lossy trapdoor function. From lossy trapdoor function, we developed the new general primitive of *lossy two-tier trapdoor function* and instantiated it using pairings. Finally, we use the lossy two-tier trapdoor function to construction the new cryptosystem called the *two-tier encryption*. The two-tier encryption can be viewed as the generalised version of identity-based encryption. From two-tier encryption, we can instantiate many encryption schemes. To conclude, Chapter 6 used pairings to further generalise the identity-based encryption.

There are some future works that we can do to improve the schemes proposed in this thesis. The signature size of our concrete construction of escrow-free identity-based signatures is much larger than that of the existing identity-based signatures. There are lots of room for improvement in terms of efficiency. It is also beneficial to have a construction in the standard model. For the fully anonymous identity-based encryption scheme in prime order groups, the ciphertext size is about 3 times more than the Waters-IBE. In addition, the security is based on some new intractability assumptions. It is possible to improve the scheme in these areas.

Although we have discussed a large number of pairing-based cryptosystems in this thesis, there are still tremendous number of cryptographic protocols which use pairings. Even within the area of identity-based cryptography, there are still some open problems. One of the most challenging problems is to construct an efficient identity-based encryption with tight security reduction to standard, simple assumption. Most of the existing identity-based encryption has a reduction loss of q, where q is the number of key extraction query, or reduced to a non-static assumption related to q (e.g. the q-SDH assumption). It would be a breakthrough if one can construct a practical identity-based encryption scheme with tight security reduction to standard assumption.

In Chapter 6, we introduce some new cryptographic primitives. They worth further investigations on what they can achieve and how to give a better and more efficient constructions.

Appendix A

Remark

My main contribution to the research included in this thesis includes: the construction of various cryptographic schemes, the security proofs and the comparison to the existing schemes. My contribution in this thesis is approximately 75% of the thesis, mainly in collaboration with my two supervisors.

For the journal paper: T. H. Yuen, W. Susilo, and Y. Mu. How to construct identity-based signatures without the key escrow problem: Formal definitions and constructions. *Int. J. Inf. Secur.*, 9(4):297-311, 2010. My contribution to the paper is the concept of escrow-free identity-based signatures, the generic construction, the concrete construction and the security proofs. My contribution in this paper is approximately 75% of the paper, in collaboration with my two supervisors.

Bibliography

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, 2005.

[2] D. Aggarwal and U. Maurer. Breaking rsa generically is equivalent to factoring. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 36–53. Springer, 2009.

[3] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(65), 1996.

[4] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In C.-S. Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 452–473. Springer, 2003.

[5] N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, 1998.

[6] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In K. Nyberg, editor, *EUROCRYPT '98*, volume 1403 of *LNCS*, pages 591–606. Springer, 1998.

[7] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 159–177. Springer, 2005.

[8] M. H. Au, Q. Huang, J. K. Liu, W. Susilo, D. S. Wong, and G. Yang. Traceable and retrievable identity-based encryption. In S. M. Bellovin, R. Gennaro, A. D.

Keromytis, and M. Yung, editors, *ACNS 2008*, volume 5037 of *LNCS*, pages 94–110, 2008.

[9] B. Barak. How to go beyond the black-box simulation barrier. In *FOCS 2001*, pages 106–115. IEEE Computer Soceity, 2001.

[10] B. Barak, Y. Lindell, and S. P. Vadhan. Lower bounds for non-black-box zero knowledge. In *FOCS 2003*, pages 384–393. IEEE Computer Soceity, 2003.

[11] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[12] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 2002*, volume 2576 of *LNCS*, pages 257–267. Springer, 2002.

[13] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, 2005.

[14] A. Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Department of Computer Science, Israel Institute of Technology, 1996.

[15] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, 2001.

[16] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 171–188. Springer, 2004.

[17] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286. Springer, 2004.

[18] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.

[19] M. Bellare and A. Palacio. GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, 2002.

[20] M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, 2009.

[21] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93*, pages 62–73. ACM, 1993.

[22] M. Bellare and S. Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In T. Okamoto and X. Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, 2007.

[23] T. Beth. Efficient zero-knowledge identification scheme for smart cards. In C. G. Günther, editor, *EUROCRYPT '88*, volume 330 of *LNCS*, pages 77–86. Springer, 1988.

[24] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. *IEEE Symposium on Security and Privacy*, 2007:321–334, 2007.

[25] G. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317. AFIPS Press, 1979.

[26] M. Blum. How to exchange (secret) keys. *ACM Trans. Comput. Syst.*, 1(2):175–193, 1983.

[27] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[28] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, 2004.

[29] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[30] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[31] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[32] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[33] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522, 2004.

[34] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[35] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, 2005.

[36] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[37] D. Boneh and J. Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103. Springer, 2005.

[38] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.

[39] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, 2000.

[40] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, 2006.

[41] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *CCS 2006*, pages 211–220. ACM, 2006.

[42] J. Boyar, D. Chaum, I. Damgård, and T. P. Pedersen. Convertible undeniable signatures. In A. Menezes and S. A. Vanstone, editors, *CRYPTO '90*, volume 537 of *LNCS*, pages 189–205. Springer, 1990.

[43] X. Boyen. The bb_1 identity-based cryptosystem: A standard for encryption and key encapsulation. Submitted to IEEE 1363.3, (August 2006), 2006.

[44] X. Boyen. A tapestry of identity-based encryption: practical frameworks compared. *IJACT*, 1(1):3–21, 2008.

[45] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[46] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer, 2006.

[47] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In A. M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *LNCS*, pages 234–238. Springer, 1986.

[48] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptography*, 37(1):133–141, 2005.

[49] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.

[50] J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 169–187. Springer, 2005.

[51] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer, 2007.

[52] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[53] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, 2003.

[54] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[55] J. C. Cha and J. H. Cheon. An identity-based signature from gap diffie-hellman groups. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 18–30. Springer, 2003.

[56] E.-C. Chang, C. L. Lim, and J. Xu. Short redactable signatures using random trees. In M. Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 133–147. Springer, 2009.

[57] M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, 2006.

[58] M. Chase and A. Lysyanskaya. Simulatable vrfs with applications to multi-theorem nizk. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 303–322. Springer, 2007.

[59] D. Chaum. Designated confirmer signatures. In A. D. Santis, editor, *EUROCRYPT '94*, volume 950 of *LNCS*, pages 86–91. Springer, 1994.

[60] D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *CRYPTO '89*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.

[61] L. Chen and Z. Cheng. Security proof of sakai-kasahara's identity-based encryption scheme. In N. P. Smart, editor, *IMA Int. Conf. 2005*, volume 3796 of *LNCS*, pages 442–459. Springer, 2005.

[62] L. Chen, C. Kudla, and K. G. Paterson. Concurrent signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 287–305. Springer, 2004.

[63] X. Chen, F. Zhang, and K. Kim. New id-based group signature from pairings. *JOURNAL OF ELECTRONICS (CHINA)*, 23(6):892–900, Nov. 2006.

[64] J. H. Cheon. Security analysis of the strong diffie-hellman problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

[65] S. S. M. Chow. Removing escrow from identity-based encryption. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 256–276. Springer, 2009.

[66] S. S. M. Chow and W. Susilo. Generic construction of (identity-based) perfect concurrent signatures. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *ICICS 2005*, volume 3783 of *LNCS*, pages 194–206. Springer, 2005.

[67] S. S. M. Chow, V. K.-W. Wei, J. K. Liu, and T. H. Yuen. Ring signatures without random oracles. In F.-C. Lin, D.-T. Lee, B.-S. P. Lin, S. Shieh, and S. Jajodia, editors, *ASIACCS 2006*, pages 297–302. ACM, 2006.

[68] C. Cocks and R. Pinch. Identity-based cryptosystems based on the weil pairing. Unpublished manuscript, 2001.

[69] I. Damgård and T. P. Pedersen. New convertible undeniable signature schemes. In U. M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 372–386. Springer, 1996.

[70] Y. Desmedt and J.-J. Quisquater. Public-key systems based on the difficulty of tampering (is there a difference between des and rsa?). In A. M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *LNCS*, pages 111–117. Springer, 1986.

[71] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[72] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous Identification in Ad Hoc Groups. In C. Cachin and J. 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 *LNCS*, pages 609–626. Springer, 2004.

[73] Y. Dodis, P. J. Lee, and D. H. Yum. Optimistic fair exchange in a multi-user setting. In T. Okamoto and X. Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 118–133. Springer, 2007.

[74] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, 2004.

[75] R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small mov degree over finite prime fields. *J. Cryptology*, 18(2):79–89, 2005.

[76] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 342–360. Springer, 2004.

[77] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *CRYPTO '93*, volume 773 of *LNCS*, pages 480–491. Springer, 1994.

[78] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[79] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[80] M. Fischlin and R. Fischlin. The representation problem based on factoring. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 96–113. Springer, 2002.

[81] M. K. Franklin and M. K. Reiter. Fair exchange with a semi-trusted third party (extended abstract). In *CCS '97*, pages 1–5. ACM, 1997.

[82] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In F. Hess, S. Pauli, and M. E. Pohst, editors, *Algorithmic Number Theory 2006*, volume 4076 of *LNCS*, pages 452–465. Springer, 2006.

[83] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptol.*, 23(2):224–280, 2010.

[84] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.

[85] G. Frey, M. Müller, and H.-G. Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Inform. Theory*, 45(5):1717–1719, 1999.

[86] S. D. Galbraith, J. F. McKee, and P. C. Valença. Ordinary abelian varieties having small embedding degree. *Finite Fields and Their Applications*, 13(4):800–814, 2007.

[87] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[88] D. Galindo. Boneh-franklin identity based encryption revisited. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 791–802. Springer, 2005.

[89] D. Galindo, J. Herranz, and E. Kiltz. On the generic construction of identity-based signatures with additional properties. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 178–193. Springer, 2006.

[90] R. Gennaro, T. Rabin, and H. Krawczyk. Rsa-based undeniable signatures. *J. Cryptology*, 13(4):397–416, 2000.

[91] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 272–293. Springer, 2003.

[92] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.

[93] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[94] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, 2009.

[95] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In I. Damgård, editor, *EUROCRYPT '90*, volume 473 of *LNCS*, pages 481–486. Springer, 1990.

[96] M. Girault. Self-certified public keys. In D. W. Davies, editor, *EUROCRYPT '91*, volume 547 of *LNCS*, pages 490–497. Springer, 1991.

[97] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2000.

[98] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS 2003*, pages 102–113. IEEE Computer Soceity, 2003.

[99] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.

[100] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 163–178. Springer, 2004.

[101] V. Goyal. Reducing trust in the PKG in Identity Based Cryptosystems. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 430–447. Springer, 2007.

[102] V. Goyal, S. Lu, A. Sahai, and B. Waters. Black-box accountable authority identity-based encryption. In P. Ning, P. F. Syverson, and S. Jha, editors, *CCS 2008*, pages 427–436. ACM, 2008.

[103] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *CCS 2006*, pages 89–98. ACM, 2006.

[104] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, 2007.

[105] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, 2008.

[106] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[107] L. C. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 216–231. Springer, 1989.

[108] F. Guo, Y. Mu, and W. Susilo. How to prove security of a signature with a tighter security reduction. In A. Joux, editor, *ProvSec 2009*, volume 5848 of *LNCS*, pages 90–103. Springer, 2009.

[109] S. Haber, Y. Hatano, Y. Honda, W. G. Horne, K. Miyazaki, T. Sander, S. Tezoku, and D. Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In M. Abe and V. D. Gligor, editors, *ASIACCS 2008*, pages 353–362. ACM, 2008.

[110] S.-H. Heng and K. Kurosawa. k-resilient identity-based encryption in the standard model. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 67–80. Springer, 2004.

[111] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. M. Heys, editors, *Selected Area in Cryptography 2002*, volume 2595 of *LNCS*, pages 310–324. Springer, 2003.

[112] F. Hess. Pairing lattices. In S. D. Galbraith and K. G. Paterson, editors, *Pairing 2008*, volume 5209 of *LNCS*, pages 18–38. Springer, 2008.

[113] F. Hess, N. P. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[114] J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. In J. Buhler, editor, *ANTS '98*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998.

[115] Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Ambiguous optimistic fair exchange. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 74–89. Springer, 2008.

[116] Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. In

T. Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 106–120. Springer, 2008.

[117] X. Huang, Y. Mu, W. Susilo, and W. Wu. A generic construction for universally-convertible undeniable signatures. In F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, editors, *CANS 2007*, volume 4856 of *LNCS*, pages 15–33. Springer, 2007.

[118] X. Huang, Y. Mu, W. Susilo, and W. Wu. Provably secure pairing-based convertible undeniable signature with short signature length. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing 2007*, volume 4575 of *LNCS*, pages 367–391. Springer, 2007.

[119] M. Izabachène and D. Pointcheval. New anonymity notions for identity-based encryption. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN 2008*, volume 5229 of *LNCS*, pages 375–391. Springer, 2008.

[120] T. Izu, N. Kanaya, M. Takenaka, and T. Yoshioka. Piats: A partially sanitizable signature scheme. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *ICICS 2005*, volume 3783 of *LNCS*, pages 72–83. Springer, 2005.

[121] T. Izu, N. Kunihiro, K. Ohta, M. Takenaka, and T. Yoshioka. A sanitizable signature scheme with aggregation. In E. Dawson and D. S. Wong, editors, *ISPEC 2007*, volume 4464 of *LNCS*, pages 51–64. Springer, 2007.

[122] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 143–154. Springer, 1996.

[123] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, 2002.

[124] A. Joux. A one round protocol for tripartite diffie-hellman. In W. Bosma, editor, *ANTS 2000*, volume 1838 of *LNCS*, pages 385–394. Springer, 2000.

[125] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In S. D. Galbraith and K. G. Paterson, editors, *Pairing 2008*, volume 5209 of *LNCS*, pages 126–135. Springer, 2008.

[126] B. G. Kang, J. H. Park, and S. G. Hahn. A certificate-based signature scheme. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 99–111. Springer, 2004.

[127] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.

[128] H. Khurana. Scalable security and accounting services for content-based publish/subscribe systems. In H. Haddad, L. M. Liebrock, A. Omicini, and R. L. Wainwright, editors, *SAC 2005*, pages 801–807. ACM, 2005.

[129] A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, 2007.

[130] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, 2006.

[131] M. Klonowski and A. Lauks. Extended sanitizable signatures. In M. S. Rhee and B. Lee, editors, *ICISC 2006*, volume 4296 of *LNCS*, pages 343–355. Springer, 2006.

[132] K. Kurosawa and J. Furukawa. Universally composable undeniable signature. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP 2008*, volume 5126 of *LNCS*, pages 524–535. Springer, 2008.

[133] K. Kurosawa and S.-H. Heng. 3-move undeniable signature scheme. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 181–197. Springer, 2005.

[134] K. Kurosawa and T. Takagi. New approach for selectively convertible undeniable signature schemes. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 428–443. Springer, 2006.

[135] F. Laguillaumie and D. Vergnaud. Short undeniable signatures without random oracles: The missing link. In S. Maitra, C. E. Veni Madhavan, and R. Venkatesan, editors, *INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 283–296. Springer, 2005.

[136] F. Laguillaumie and D. Vergnaud. Time-selective convertible undeniable signatures. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 154–171. Springer, 2005.

[137] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, CA, USA, 1979.

[138] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[139] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.

[140] J. Li, C. Lu, and W. Shi. An efficient scheme for preserving confidentiality in content-based publish-subscribe systems, 2004. Tech. Rep. GIT-CC-04-01, Georgia Institute of Technology.

[141] Y. Li, D. He, and X. Lu. Accountability of perfect concurrent signature. In *ICCEE 2008*, pages 773–777. IEEE Computer Society, 2008.

[142] B. Libert and J.-J. Quisquater. Identity based undeniable signatures. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 112–125. Springer, 2004.

[143] B. Libert and D. Vergnaud. Towards black-box accountable authority ibe with short ciphertexts and private keys. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 235–255. Springer, 2009.

[144] J. Liu, R. Sun, W. Ma, Y. Li, and X. Wang. Fair exchange signature schemes. In *AINAW 2008*, pages 422–427. IEEE Computer Society, 2008.

[145] W. Mao. *Modern Cryptography: Theory and Practice*. Prentice-Hall, 2003.

[146] U. M. Maurer and S. Wolf. The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. *SIAM J. Comput.*, 28(5):1689–1721, 1999.

[147] U. M. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D. W. Davies, editor, *EUROCRYPT '91*, volume 547 of *LNCS*, pages 498–507. Springer, 1991.

[148] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[149] R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *CRYPTO '87*, volume 293 of *LNCS*, pages 369–378. Springer, 1987.

[150] R. C. Merkle. One way hash functions and des. In G. Brassard, editor, *CRYPTO '89*, volume 435 of *LNCS*, pages 428–446. Springer, 1989.

[151] M. Michels, H. Petersen, and P. Horster. Breaking and repairing a convertible undeniable signature scheme. In *CCS '96*, pages 148–152. ACM, 1996.

[152] M. Michels and M. Stadler. Efficient convertible undeniable signature schemes. In *SAC '97*, pages 231–244, 1997.

[153] V. S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.

[154] S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002.

[155] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, 2001.

[156] K. Miyazaki, G. Hanaoka, and H. Imai. Digitally signed document sanitizing scheme based on bilinear maps. In F.-C. Lin, D.-T. Lee, B.-S. Lin, S. Shieh, and S. Jajodia, editors, *ASIACCS 2006*, pages 343–354. ACM, 2006.

[157] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Transactions*, 88-A(1):239–246, 2005.

[158] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura. Digital documents sanitizing problem. *IEICE Technical Report*, ISEC2003-20:61–67, 2003.

[159] J. Monnerat and S. Vaudenay. Generic homomorphic undeniable signatures. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 354–371. Springer, 2004.

[160] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *STOC '89*, pages 33–43. ACM, 1989.

[161] K. Nguyen. Asymmetric concurrent signatures. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *ICICS 2005*, volume 3783 of *LNCS*, pages 181–193. Springer, 2005.

[162] L. Nguyen. Accumulators from bilinear pairings and applications. In A. J. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.

[163] P. Nikander and M. F. Giannis. Towards understanding pure publish/subscribe cryptographic protocols. In *Sixteenth International Workshop on Security Protocols*, 2008.

[164] M. Nishioka. Reconsideration on the security of the boneh-franklin identity-based encryption scheme. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 270–282. Springer, 2005.

[165] W. Ogata, K. Kurosawa, and S.-H. Heng. The security of the fdh variant of chaum's undeniable signature scheme. *IEEE Transactions on Information Theory*, 52(5):2006–2017, 2006.

[166] K. Ohta and T. Okamoto. A modification of the fiat-shamir scheme. In S. Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 232–243. Springer, 1990.

[167] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. F. Brickell, editor, *CRYPTO '92*, volume 740 of *LNCS*, pages 31–53. Springer, 1993.

[168] T. Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In Y. Desmedt, editor, *CRYPTO '94*, volume 839 of *LNCS*, pages 61–74. Springer, 1994.

[169] S. Orzan and M. T. Dashti. Fair exchange is incomparable to consensus. In J. S. Fitzgerald, A. E. Haxthausen, and H. Yenigün, editors, *ICTAC 2008*, volume 5160 of *LNCS*, pages 349–363. Springer, 2008.

[170] H. Pagnia and F. C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical report, TUD-BS-1999-02, Department of Computer Science, Darmstadt University of Technology, 1999.

[171] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[172] K. G. Paterson and S. Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In S. D. Galbraith and K. G. Paterson, editors, *Pairing 2008*, volume 5209 of *LNCS*, pages 354–375. Springer, 2008.

[173] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*. Springer, 2008.

[174] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *STOC 2008*, pages 187–196. ACM, 2008.

[175] L. I. W. Pesonen, D. M. Eyers, and J. Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In H.-A. Jacobsen, G. Mühl, and M. A. Jaeger, editors, *DEBS 2007*, volume 233 of *ACM International Conference Proceeding Series*, pages 104–115. ACM, 2007.

[176] L. T. Phong, K. Kurosawa, and W. Ogata. New dlog-based convertible undeniable signature schemes in the standard model. Cryptology ePrint Archive, Report 2009/394, 2009. `http://eprint.iacr.org/`.

[177] L. T. Phong, K. Kurosawa, and W. Ogata. New rsa-based (selectively) convertible undeniable signature schemes. In B. Preneel, editor, *AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 116–134. Springer, 2009.

[178] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Jan. 1979.

[179] C. Raiciu and D. S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Securecomm '06: Proceedings of the Second IEEE/CreatNet International Conference on Security and Privacy in Communication Networks*, 2006.

[180] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[181] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 371–388. Springer, 2004.

[182] A. Rosen and G. Segev. Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134, 2008. `http://eprint.iacr.org/`.

[183] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[184] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/`.

[185] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. The 2000 Symposium on Cryptography and Information Security, 2000.

[186] C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO '89*, volume 435 of *LNCS*, pages 239–252. Springer, 1990.

[187] M. Scott and P. S. L. M. Barreto. Generating more mnt elliptic curves. *Des. Codes Cryptography*, 38(2):209–217, 2006.

[188] A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22(11), pages 612–613. ACM Press, 1979.

[189] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[190] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, Jul, Oct 1948.

[191] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[192] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *EUROCRYPT '98*, volume 1403 of *LNCS*, pages 1–16. Springer, 1998.

[193] J. H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer, 1986.

[194] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[195] M. Srivatsa and L. Liu. Secure event dissemination in publish-subscribe networks. In *ICDCS 2007*, page 22. IEEE Computer Society, 2007.

[196] P. Stanica. Good lower and upper bounds on binomial coefficients. *J. Inequal. Pure and Appl. Math*, 2(3), 2001. Art. 30.

[197] R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In K. Kim, editor, *ICISC 2001*, volume 2288 of *LNCS*, pages 285–304. Springer, 2001.

[198] A. F. Sui, S. S. M. Chow, L. C. K. Hui, S.-M. Yiu, K. P. Chow, W. W. Tsang, C. F. Chong, K. K. H. Pun, and H. W. Chan. Separable and anonymous identity-based key issuing. In *ICPADS 2005*, pages 275–279. IEEE Computer Society, 2005.

[199] W. Susilo, Y. Mu, and F. Zhang. Perfect concurrent signature schemes. In J. Lopez, S. Qing, and E. Okamoto, editors, *ICICS 2004*, volume 3269 of *LNCS*, pages 14–26. Springer, 2004.

[200] M. Suzuki, T. Isshiki, and K. Tanaka. Sanitizable signature with secret information. Symposium on Cryptography and Information Security, 2006. 4A1-2.

[201] H. Tanaka. A realization scheme for the identity-based cryptosystem. In C. Pomerance, editor, *CRYPTO '87*, volume 293 of *LNCS*, pages 341–349. Springer, 1987.

[202] D. Tonien, W. Susilo, and R. Safavi-Naini. Multi-party concurrent signatures. In S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B. Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 131–145. Springer, 2006.

[203] S. Tsujii and T. Itoh. An id-based cryptosystem based on the discrete logarithm problem. *IEEE J. Selected Areas in Comm.*, 7(4):467–473, 1989.

[204] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. *Hawaii International Conference on System Sciences*, 9:303, 2002.

[205] G. Wang, F. Bao, and J. Zhou. The fairness of perfect concurrent signatures. In P. Ning, S. Qing, and N. Li, editors, *ICICS 2006*, volume 4307 of *LNCS*, pages 435–451. Springer, 2006.

[206] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[207] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008. `http://eprint.iacr.org/`.

[208] B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.

[209] C.-K. Wu and V. Varadharajan. Fair exchange of digital signatures with offline trusted third party. In S. Qing, T. Okamoto, and J. Zhou, editors, *ICICS 2001*, volume 2229 of *LNCS*, pages 466–470. Springer, 2001.

[210] T. H. Yuen, M. H. Au, J. K. Liu, and W. Susilo. (Convertible) undeniable signatures without random oracles. In S. Qing, H. Imai, and G. Wang, editors, *ICICS 2007*, volume 4861 of *LNCS*, pages 83–97. Springer, 2007.

[211] T. H. Yuen, W. Susilo, J. K. Liu, and Y. Mu. Sanitizable signatures revisited. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *CANS 2008*, volume 5339 of *LNCS*, pages 80–97. Springer, 2008.

[212] T. H. Yuen, W. Susilo, and Y. Mu. Cryptographic treatment of publish/subscribe systems. In S.-H. Heng, R. N. Wright, and B.-M. Goi, editors, *CANS 2010*, volume 6467 of *LNCS*, page 201220. Springer, 2010.

[213] T. H. Yuen, W. Susilo, and Y. Mu. How to construct identity-based signatures without the key escrow problem. In F. Martinelli and B. Preneel, editors, *EuroPKI 2009*, volume 6391 of *LNCS*, pages 286–301. Springer, 2010.

[214] T. H. Yuen, W. Susilo, and Y. Mu. How to construct identity-based signatures without the key escrow problem: Formal definitions and constructions. *Int. J. Inf. Secur.*, 9(4):297–311, 2010.

[215] H. Zhang, A. Sharma, H. Chen, G. Jiang, X. Meng, and K. Yoshihira. Enabling information confidentiality in publish/subscribe overlay services. In *ICC 2008*, pages 5624–5628. IEEE, 2008.

[216] Y. Zhao and D. C. Sturman. Dynamic access control in a content-based publish/subscribe system with delivery guarantees. In *ICDCS 2006*, page 60. IEEE Computer Society, 2006.