NEW SECURITY NOTIONS AND EXTENSIONS OF CRYPTOGRAPHIC PRIMITIVES THROUGH GENERIC TRANSFORMATIONS AMONG DIFFERENT PRIMITIVES

YUSUKE SAKAI

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

MARCH 2014

NEW SECURITY NOTIONS AND EXTENSIONS OF CRYPTOGRAPHIC PRIMITIVES THROUGH GENERIC TRANSFORMATIONS AMONG DIFFERENT PRIMITIVES

YUSUKE SAKAI

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

GRADUATE SCHOOL OF INFORMATICS AND ENGINEERING

A DISSERTATION SUBMITTED FOR
THE DOCTOR OF PHILOSOPHY IN ENGINEERING

MARCH 2014

NEW SECURITY NOTIONS AND EXTENSIONS OF CRYPTOGRAPHIC PRIMITIVES THROUGH GENERIC TRANSFORMATIONS AMONG DIFFERENT PRIMITIVES

SUPERVISORY COMMITTEE

CHAIRPERSON: PROFESSOR   KAZUO OHTA

MEMBER: PROFESSOR   KIYOSHI ANDOH

MEMBER: PROFESSOR   SATOSHI KOBAYASHI

MEMBER: PROFESSOR   KAZUO SAKIYAMA

MEMBER: PROFESSOR   HIROSHI YOSHIURA

暗号要素技術の一般的構成を介した高い安全性・高度な機能を備えた暗号要素技術の構成

坂井 祐介

論文概要

通常の暗号化や認証にとどまらず, 複雑な機能を備えた暗号要素技術の提案が活発になっている. 暗号要素技術の安全性は利用形態に応じて, セキュリティ上の脅威をモデル化して安全性要件を定め, 新方式はそれぞれ安全性定義を満たすことの証明と共に提案される.

既存研究では, 次の問題があった: (1) 要素技術ごとに個別に安全性の定義を与えているため, 理論的な体系化が不十分であった. (2) 安全性定義が実用上の脅威を完全に捉えきれているかの検証が難しかった.

本研究は上記の問題を次の考え方で解決する. (1) ある要素技術 (A) を構成するには別の要素技術 (B) を部品として用いることが不可欠であることに注目し, 各要素技術の安全性要件の関連を整理・体系化して, 新方式を見通し良く構成可能とする. (2) 要素技術 (B) で考慮されていた安全性要件 (b) に対応する要素技術 (A) の安全性要件が未定義なら, それを (A) の新たな安全性要件 (a) として定式化する. これにより未知の脅威の検出が容易になる.

グループ署名と非対話開示機能付き公開鍵暗号という 2 つの要素技術について上記の考え方を適用して, グループ署名について未知の脅威を指摘する.

また, 証明書失効機能と呼ばれる拡張機能を持つグループ署名に上記の考え方を適用して, 公開鍵暗号についての新たな拡張機能である, 暗号化できる平文を制限できる公開鍵暗号の効率的な構成法を明らかにする.

NEW SECURITY NOTIONS AND EXTENSIONS OF CRYPTOGRAPHIC PRIMITIVES THROUGH GENERIC TRANSFORMATIONS AMONG DIFFERENT PRIMITIVES

YUSUKE SAKAI

ABSTRACT

Recent years have witnessed an active research on cryptographic primitives with complex functionality beyond simple encryption or authentication. A cryptographic primitive is required to be proposed together with a formal model of its usage and a rigorous proof of security under that model.

This approach has suffered from the two drawbacks: (1) security models are defined in a very specific manner for each primitive, which situation causes the relationship between these security models not to be very clear, and (2) no comprehensive ways to confirm that a formal model of security really captures every possible scenarios in practice.

This research relaxes these two drawbacks by the following approach: (1) By observing the fact that a cryptographic primitive A should be crucial for constructing another primitive B, we identify an easy-to-understand approach for constructing various cryptographic primitives. (2) Consider a situation in which there are closely related cryptographic primitives A and B, and the primitive A has no known security requirement that corresponds to some well-known security requirement (b) for the latter primitive B. We argue that this situation suggests that this unknown security requirement for A can capture some practical attack. This enables us to detect unknown threats for various cryptographic primitives that have been missed by the current security models.

Following this approach, we identify an overlooked security threat for a cryptographic primitive called group signature. Furthermore, we apply the methodology (2) to the "revocable" group signature and obtain a new extension of public-key encryption which allows to restrict a plaintext that can be securely encrypted.

Contents

Notations

$[n]$	$\{1,\ldots,n\}$
$[n,m]$	$\{n,\ldots,m\}$
ek	Encryption key for a PKE/TBE/PKENO scheme
dk	Decryption key for a PKE/TBE/PKENO scheme
vk	Verification key for a signature scheme
sk	Signing key for a signature scheme
gpk	Group public key for a group signature scheme
ik	Issuing key for a group signature scheme
ok	Opening key for a group signature scheme
(upk, usk)	User public/secret keys for a group signature scheme
gsk	Group signing key for each group member
cert	Membership certificate for each group member
$(pk, vk, (sk_i)_{i\in[n]})$	Public/verification/secret keys for a threshold encryption scheme
ck	Commitment key
$\mathcal{G}$	Parameter generator for a cryptographic (bilinear) group
$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$	Bilinear map
$\oplus$	Bit-wise exclusive OR
$y \leftarrow A(x)$	Operation of running algorithm A on input x and setting y to be its output
$y \leftarrow A(x;r)$	Same as $y \leftarrow A(x)$, to explicitly mention the randomness r for A
$\Pr[O_1;\ldots O_n : E]$	Probability of event E after operations $O_1, \ldots, O_n$ performed
$\langle x_1,\ldots,x_n\rangle$	Uniquely-decodable binary encoding of tuple $(x_1,\ldots,x_n)$

Part I

Introduction

Chapter 1

Introduction

1.1 Background

In the recent highly-computerized society, information security is increasingly important. Among various information security technologies, the public-key encryption for secret message transmission and the digital signature for message integrity are the central tools for secure communications.

In addition to these fundamental primitives, more sophisticated cryptography than the simple encryption or authentication, is also the target of active research (These cryptographic schemes are often called *cryptographic primitives* or simply *primitives*). Group signature, which is a "anonymous" authentication, is a typical example of such cryptographic primitives. For public-key encryption, various extensions, that are suitable for various types of multiparty computation, are also the target of actively research. Although most of these sophisticated cryptographic primitives are far from a practical use, some of them are almost ready to use in practice, as their standardization processes are almost complete.

1.1.1 The "Provable Security" Framework

The "provable security" framework is a nowadays standard approach for analyzing cryptographic schemes. In this framework, we first describe a mathematical "model" for the practical use of the cryptographic primitive to be analyzed, and after that we provide a mathematical prove that any attempt to attack which can be mounted under the model will fail. This approach provides a useful tool to specify "what level of security is achieved" by the cryptographic scheme and to validate the security claim of the scheme in a rigorous way.

1.1.2 The Problem and Our Approach

However, even the provable security framework has suffered from several weakness. The most important drawback is that the security model for various cryptographic primitives (sometimes they are called as "security notion") have been defined in an *ad hoc* and primitive-specific manner. This situation is not very satisfiable, as it tends to cause a bunch of extension of primitives, their security notions, and ad hoc constructions of these primitives. The worse thing is, it often occurs that once researcher defines a security notions for some primitive, they tend to stick to constructing a scheme that satisfying this security notion and not to explore further practical threats that are not included in the current security notion.

To relax the above drawbacks, this research takes the following approach.

- By clarifying the relationship among security notions of different cryptographic primitives, we try to identify techniques in common among various primitives and techniques specific to some primitives, even for quite complex primitives.
- we try to clarify the relative strength of various security notions (in particular, in the form of any scheme that satisfies some security notion can be *generically* transformed to another scheme that satisfies a different security notion).

This approach is not only interesting in a theoretical point of view, but we below argue that it is also helpful for the above-mentioned problems.

1.1.3 The Benefit of Our Approach

In the following we describe the merit of our approach.

(I) Guidelines for Designing Cryptographic Schemes. Firstly, our approach can provide a useful guideline for designing a concrete cryptographic scheme. In particular, following our approach of clarifying the relationship of security notions, when we need to design a new cryptographic scheme satisfying new security notion (possibly stronger than known notions), we can easily choose appropriate building blocks, can easily provide precise security proof.

In particular, let us assume that by following our approach, we observe that a scheme with the new security notion can be generically transformed to another well-known primitive with a well-known security notion. This can be interpreted as a strong evidence that the latter well-known primitive is crucial to construct the former, less-understood primitive. Thus it further suggests that to use the former primitive is a promising approach to construct the latter new primitive.

In contrast to this, let us assume that any scheme(s) with well-known security notion(s) can be generically transformed to a scheme with the new security notion. In this case, the obvious merit is of course that by following this generic transformation we can obtain a scheme that satisfies the new security notion. However, often such a generic construction will not necessarily achieve practical performance, and for these cases we will deviate (or optimize) the generic construction to construct a more practical scheme. A demerit of this optimized scheme is that we need to provide a security proof for the optimized scheme separately, since the security proof for the generic construction no longer ensures the security of the optimized scheme. Fortunately, for most cases, the security proofs for both the generic construction and the optimized scheme share the basic principle. Then the security proof for the generic construction can serve as a guideline for the security proof for the optimized scheme.

This approach is potentially useful when we need to move from current (number-theoretic) cryptography from cryptography based on completely different hardness assumption due to, for example, practical quantum computers. Even for such a case, since our approach identifies the core technique for various cryptographic primitives, at first we will realize these core techniques from the new hardness assumptions, and thus over these core techniques we can re-construct various cryptographic primitives.

(II) More Comprehensive Detection of Overlooked Threat. The other merit of our approach is that it potentially enable us to identify overlooked practical threat more easily. This is because investigating the relationship between among various primitives we may find closely related two primitives A and B in which the former primitive A has no well-known security notion which corresponds to a well-known security notion for the latter primitive B. In this case, we can define a new security notion for A to complement the correspondence between two primitives. This new security notion might be artificial and of only theoretical interest. However, if their counterparts well capture some practical threats, it is natural to expect such new security notions to have practical importance. Moreover, it is plausible for recent sophisticated cryptographic primitives with quite complex usage to having overlooked practical threats.

1.2 Contributions

This section describes the contribution of this thesis.

1.2.1 On Public-key Encryption with Non-interactive Opening

The first part of the contribution is on the relationship among the primitive called *public-key encryption with non-interactive opening (PEKNO)* [DHKT08] and other cryptographic primitives.

1.2.1.1 Application to Group Signature

Group signature is a cryptographic primitive for anonymous authentication. It is well-known that a group signature scheme can be constructed from a combination of a public-key encryption scheme (satisfying a certain security notion) and other cryptographic primitives [BMW03, BSZ05, AW04, OFHO09].

We will investigate this relationship between group signature and public-key encryption, from the viewpoint (II) of the above. In particular, we observed that the security levels achieved by PKENO (PKENO can be seen as an extension of public-key encryption with a stronger security notion) will not correspond to any known security notions for group signature, and thus from the viewpoint (II) we will obtain a new security notion for group signature (We name this *opening soundness*). We stress that this security notion is not included in the Bellare-Shi-Zhang model, which it a currently standard security definition for group signature.

Furthermore, we also argue that opening soundness captures several threat of practical importance. For example, the lack of opening soundness will be crucial in an anonymous auctions using group signature (it is a typical and well-known application of group signature [ACJT00]). We will discuss that if the group signature scheme does *not* have opening soundness, it is possible for a malicious bidder who did not bid the highest price to claim falsely in a publicly verifiable way that he bid the highest price.

In addition to this, from the viewpoint (I), we will investigate what is a crucial building block for achieving opening soundness. Firstly, we will construct group signature schemes with opening soundness, using techniques of PKENO. In contrast, we also shows that any group signature scheme that satisfies opening soundness can be transformed to a PKENO scheme. Following the viewpoint (I), these two result shows that PKENO is a crucial tool for achieving opening soundness.

1.2.1.2 Applications to Threshold Encryption

Threshold (public-key) encryption is an extension of public-key encryption which divides a secret keys of public-key encryption into several decryption servers to avoid a single point of failure. As the name suggest, a ciphertext can be decrypted if decryption servers more than some threshold cooperate. Robustness is a security notion for threshold encryption, which intuitively requires that any deviation from the protocol of decryption server can be detected.

We will investigate the robustness notion from the viewpoint (I) of the above.

Galindo et al. showed that a robust threshold encryption scheme can be constructed from any secure PKENO scheme [GLF$^+$10]. Following the viewpoint (I), their result suggests that using PKENO is crucial to for constructing TPKE. The other direction, however, is not clearly and exhaustively stated in the literature.

We will investigate the latter direction, and will show that from any PKENO scheme we can construct a robust threshold encryption scheme. From the viewpoint (I) this result can be interpreted as the evidence for the equivalence between the existence of PKENO and that of robust threshold encryption, and will provide a useful suggestion for constructing both robust threshold encryption and PKENO.

1.2.2 On Revocable Group Signature

Revocable group signature is an extension of group signatures with an additional functionality for securely expires some secret keys for authentication. As mentioned above, a group signature scheme can be constructed using public-key encryption as a building block [BMW03, BSZ05, AW04, OFHO09]. The viewpoint (II) suggests that this relationship between group signature and public-key encryption can be extended to revocable group signature, and when doing so we have some extended functionality for public-key encryption.

By promoting this idea, we will present a new extension of public-key encryption called *restrictive public-key encryption (restrictive PKE)*, which enable us to restrict plaintext that is securely encrypted. Furthermore, an efficient construction of restrictive PKE from using techniques of revocable group signature schemes.

Chapter 2

Preliminary

In this chapter we introduce notations and definitions used throughout this thesis.

2.1 Public-key Encryption and Its Extensions

2.1.1 Chosen-ciphertext Secure Public-key Encryption

A public-key encryption scheme $(\mathsf{PKg}, \mathsf{PEnc}, \mathsf{PDec})$ consists of the following three probabilistic polynomial-time algorithms:

PKg. The key-generation algorithm PKg takes as input the security parameter 1^λ and outputs a pair (ek, dk) of the encryption key and the decryption key. The encryption key ek implicitly specifies the message space $\mathcal{M}_{ek}$, which determines the set of plaintexts that can be encrypted under that encryption key.

PEnc. The encryption algorithm PEnc takes as input an encryption key ek and a plaintext $m \in \mathcal{M}_{ek}$ and outputs a ciphertext c.

PDec. The decryption algorithm PDec takes as input a decryption key dk and a ciphertext c and outputs a plaintext $m \in \mathcal{M}_{ek}$ or a special symbol $\perp$ indicating the decryption failure.

A public-key encryption scheme is required to satisfy the following correctness condition: for all $\lambda \in \mathbb{N}$, all $(ek, dk) \leftarrow \mathsf{PKg}(1^\lambda)$, all $m \in \mathcal{M}_{ek}$, it holds that $\mathsf{PDec}(dk, \mathsf{PEnc}(ek, m)) = m$

with probability one.

Chosen-ciphertext security [NY90, RS92] is a nowadays standard security notion for public-key encryption schemes. This security notion intuitively requires that even when an adversary knows the decryption result of some ciphertexts of her choice, it does not help her for recovering the plaintext of any other ciphertext. The formal definition of this notion is defined by the following experiment.

```
$\mathrm{Exp}^{\mathrm{CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$:
    $b \leftarrow \{0,1\}$
    $(ek, dk) \leftarrow \mathsf{PKg}(1^\lambda)$
    $(m_0, m_1, state) \leftarrow \mathcal{A}_1{}^{\mathsf{PDec}(dk,\cdot)}(ek)$
    $c^* \leftarrow \mathsf{PEnc}(ek, m_b)$
    $b' \leftarrow \mathcal{A}_2{}^{\mathsf{PDec}(dk,\cdot)}(state, c^*)$
    return $b = b'$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2)$ is required to output m_0 and m_1 which satisfy $|m_0| = |m_1|$, and $\mathcal{A}_2$ is required not to submit c^* to its oracle.

Definition 2.1. A public-key encryption scheme $(\mathsf{PKg}, \mathsf{PEnc}, \mathsf{PDec})$ is chosen-ciphertext secure if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$ the advantage $\mathrm{Adv}^{\mathrm{CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}^{\mathrm{CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1] - 1|$ is negligible in λ.

2.1.2 Tag-based Encryption

Tag-based encryption [MRY04] is an extension of public-key encryption. It allows a sender to associate the ciphertext with an arbitrary string called the tag, and the decryption is performed under some specific tag. The formal syntax is as follows.

TKg. The key-generation algorithm PKg takes as input the security parameter 1^λ and outputs a pair (ek, dk) of the encryption key and the decryption key. The encryption key ek implicitly specifies the message space $\mathcal{M}_{ek}$, which determines the set of plaintexts that can be encrypted under that encryption key.

TEnc. The encryption algorithm PEnc takes as input an encryption key ek, a tag t, a plaintext $m \in \mathcal{M}_{ek}$ and outputs a ciphertext c associated with the tag t.

TDec. The decryption algorithm PDec takes as input a decryption key dk, a tag t, and a ciphertext c and outputs a plaintext $m \in \mathcal{M}_{ek}$ or a special symbol $\perp$ indicating the decryption failure.

We require a tag-based encryption scheme to satisfy the following correctness condition: for all $\lambda \in \mathbb{N}$, all $(ek, dk) \leftarrow \mathsf{TKg}(1^\lambda)$, all $m \in \mathcal{M}_{ek}$, and any tag t it holds that $\mathsf{TDec}(dk, t, \mathsf{TEnc}(ek, t, m)) = m$ with probability one.

In this thesis we will utilize the security notion for tag-based encryption schemes called selective-tag weak chosen-ciphertext security [Kil06]. This notion is formally defined by the following experiment.

```
$\mathrm{Exp}^{\text{sTag-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda)$:
  $b \leftarrow \{0,1\}$
  $(t^*, state_1) \leftarrow \mathcal{A}_1(1^\lambda)$
  $(ek, dk) \leftarrow \mathsf{TKg}(1^\lambda)$
  $(m_0, m_1, state_2) \leftarrow \mathcal{A}_2{}^{\mathsf{TDec}(dk,\cdot,\cdot)}(state_1, ek)$
  $c^* \leftarrow \mathsf{TEnc}(ek, t^*, m_b)$
  $b' \leftarrow \mathcal{A}_3{}^{\mathsf{TDec}(dk,\cdot,\cdot)}(state_2, c^*)$
  return $b = b'$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ is required to output m_0 and m_1 which satisfy $|m_0| = |m_0|$, and not to submit any decryption query of the form (t^*, c) with any c throughout the experiment[*1].

Definition 2.2. A tag-based encryption scheme $(\mathsf{TKg}, \mathsf{TEnc}, \mathsf{TDec})$ is selective-tag weak chosen-ciphertext secure if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ the advantage $\mathrm{Adv}^{\text{sTag-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}^{\text{sTag-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(k) = 1] - 1|$ is negligible in λ.

[*1] The name "weak" comes from this restriction. In contrast to this, the (ordinary, non-weak) chosen-ciphertext security for tag-based encryption only forbids $\mathcal{A}_3$ to submit the query (t^*, c^*).

2.1.3 Public-key Encryption with Non-interactive Opening

Public-key encryption with non-interactive opening is another extension of public-key encryption [DHKT08]. This allows the receiver to demonstrate the decryption result of any given ciphertext to any third-party, without making the decryption key itself public. Formally this primitive is constituted by the following five algorithms.

NOKg. The key-generation algorithm NOKg takes as input the security parameter 1^λ and outputs a pair (ek, dk) of the encryption key and the decryption key. The encryption key ek implicitly specifies the message space $\mathcal{M}_{ek}$, which determines the set of plaintexts that can be encrypted under that encryption key.

NOEnc. The encryption algorithm NOEnc takes as input an encryption key ek and a plaintext $m \in \mathcal{M}_{ek}$ and outputs a ciphertext c.

NODec. The decryption algorithm NODec takes as input a decryption key dk and a ciphertext c and outputs a plaintext $m \in \mathcal{M}_{ek}$ or a special symbol $\perp$ indicating the decryption failure.

$\mathsf{NOProve}$. The proof algorithm $\mathsf{NOProve}$ takes as input a decryption key dk and a ciphertext c and outputs a proof π.

$\mathsf{NOVerify}$. The verification $\mathsf{NOVerify}$ takes as input an encryption key ek, a ciphertext c, a decryption result $m \in \mathcal{M}_{ek} \cup \{\perp\}$, and a proof π, and outputs a symbol $\top$ or $\perp$ indicating the validity or invalidity of the proof, respectively.

As correctness requirements, a PKENO scheme is required to satisfy the following conditions:

1. for all $\lambda \in \mathbb{N}$, any $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$, and any plaintext $m \in \mathcal{M}_{ek}$, it holds that $\mathsf{NODec}(dk, \mathsf{NOEnc}(ek, m)) = m$ with probability one, and
2. for all $\lambda \in \mathbb{N}$, any $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$, and any ciphertext c not necessarily generated honestly by NOEnc, it holds that $\mathsf{NOVerify}(ek, c, \mathsf{NODec}(dk, c), \mathsf{NOProve}(dk, c)) = \top$.

Basically there are two different kinds of security requirements for PKENO schemes, the first of which is usual plaintext secrecy, and the other is soundness of the proof. The former is defined as an extension of chosen-ciphertext security of public-key encryption. The latter has, actually, two variations of definitions, which is relatively weaker and stronger.

The secrecy requirement is defined as a natural extension of chosen-ciphertext security (of public-key encryption), and is also called the chosen-ciphertext security. The formal definition of the chosen-ciphertext security for PKENO schemes are defined by the following experiment.

```
$\mathrm{Exp}^{\text{PKENO-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$:
    $b \leftarrow \{0,1\}$
    $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$
    $(m_0, m_1, state) \leftarrow \mathcal{A}_1{}^{\mathsf{NODec}(dk,\cdot),\mathsf{NOProve}(dk,\cdot)}(ek)$
    $c^* \leftarrow \mathsf{NOEnc}(ek, m_b)$
    $b' \leftarrow \mathcal{A}_2{}^{\mathsf{PDec}(dk,\cdot),\mathsf{NOProve}(dk,\cdot)}(state, c^*)$
    return $b = b'$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2)$ is required to output m_0 and m_1 with $|m_0| = |m_1|$, and $\mathcal{A}_2$ is required not to submit c^* to its both oracles $\mathsf{NODec}(dk, \cdot)$ and $\mathsf{NOProve}(dk, \cdot)$.

Definition 2.3. A PKENO scheme ($\mathsf{NOKg}, \mathsf{NOEnc}, \mathsf{NODec}, \mathsf{NOProve}, \mathsf{NOVerify}$) is chosen-ciphertext secure if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$ the advantage $\mathrm{Adv}^{\text{PKENO-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}^{\text{PKENO-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1] - 1|$ is negligible in λ.

The first variant of the soundness notions is proof soundness. This security notion requires even an malicious receiver, who possesses the decryption key, to be unable to produce a false proof that claims that a ciphertext would decrypted to a different result than that is actually is. The exact definition of the experiment is as follows.

```
$\mathrm{Exp}^{\text{PKENO-Sound}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$:
    $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$
    $(m, state) \leftarrow \mathcal{A}_1(ek, dk)$
    $c \leftarrow \mathsf{NOEnc}(ek, m)$
    $(m', \pi') \leftarrow \mathcal{A}_2(state, c)$
    return $(\mathsf{NOVerify}(ek, c, m', \pi') = \top) \wedge (m \neq m')$
```

In the experiment, m' is allowed to be $\perp$. This is interpreted as it excludes the possibility of producing two different proofs for single ciphertext in which one proof claims that the ciphertext is decrypted successfully, and the other claims that it will be rejected as an invalid ciphertext.

Definition 2.4. A PKENO scheme $(\mathsf{NOKg}, \mathsf{NOEnc}, \mathsf{NODec}, \mathsf{NOProve}, \mathsf{NOVerify})$ satisfies proof soundness if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$ the advantage $\mathrm{Adv}^{\text{PKENO-Sound}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = \Pr[\mathrm{Exp}^{\text{PKENO-Sound}}_{(\mathcal{A}_1,\mathcal{A}_2)} = 1]$ is negligible in λ.

The last security notion is called the committing property, which is firstly defined by Galindo et al. [GLF+10]. This security notion, intuitively, requires soundness of the proof to hold even for maliciously generated ciphertexts. As opposed to this, the proof soundness notion, only requires soundness to hold for honestly generated ciphertext. It comes from the description of the experiment of proof soundness, in which the ciphertext c is generated by the encryption algorithm NOEnc but not generated by the adversary. The experiment for the committing property is as follows.

```
$\mathrm{Exp}^{\text{PKENO-Commit}}_{\mathcal{A}}(\lambda)$:
  $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$
  $(c, m, \pi, m', \pi') \leftarrow \mathcal{A}(ek, dk)$
  return $(\mathsf{NOVerify}(ek, c, m, \pi) = \top) \wedge (\mathsf{NOVerify}(ek, c, m', \pi') = \top) \wedge (m \neq m')$
```

In the experiment, the plaintexts m and m' is required to be in $\mathcal{M}_{ek} \cup \{\perp\}$. As in the proof soundness, m and m' are allowed to be $\perp$.

Definition 2.5. A PKENO scheme $(\mathsf{NOKg}, \mathsf{NOEnc}, \mathsf{NODec}, \mathsf{NOProve}, \mathsf{NOVerify})$ is committing if for all probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{PKENO-Commit}}_{\mathcal{A}}(\lambda) = \Pr[\mathrm{Adv}^{\text{PKENO-Commit}}_{\mathcal{A}}(\lambda) = 1]$ is negligible in λ.

2.2 Digital Signatures

A signature scheme consists of the following three algorithms:

SgKg. The key-generation algorithm SgKg takes as input the security parameter 1^λ and outputs a pair (vk, sk) of the verification key and the signing key.

SgSign. The signing algorithm SgSign takes as input a signing key sk and a message m and outputs a signature σ.

$\mathsf{SgVerify}$. The verification algorithm $\mathsf{SgVerify}$ takes as input a verification key vk, a message m, and a signature σ and outputs a symbol $\top$ or $\bot$ indicating validity or invalidity, respectively, of the signature for the message.

As a correctness requirement, it is required that for all $\lambda \in \mathbb{N}$, any message m it holds that $\mathsf{SgVerify}(vk, m, \mathsf{SgSign}(sk, m)) = \top$ with probability one.

The standard security notion for signature scheme is existential unforgeability under chosen-message attacks [GMR88]. This notion is defined via the following experiment.

```
$\mathrm{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}(\lambda)$:
  $(vk, sk) \leftarrow \mathsf{SgKg}(1^\lambda)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SgSign}(sk,\cdot)}(vk)$
  return $\mathsf{SgVerify}(vk, m^*, \sigma^*) = \top$ and $m^*$ is not queried by $\mathcal{A}$
```

Definition 2.6. A signature scheme $(\mathsf{SgKg}, \mathsf{SgSign}, \mathsf{SgVerify})$ is existentially unforgeable under chosen-message attacks if for all probabilistic polynomial-time adversary $\mathcal{A}$ the advantage $\mathrm{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(\lambda) = \Pr[\mathrm{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}(\lambda) = 1]$ is negligible in λ.

Another useful notion of security of signature scheme is the strong unforgeability. Th strong unforgeability further requires that no adversary, given a correct signature–message pair (m, σ), is able to alter the signature σ to σ' which is verified as a valid signature for the *same* message m. In this thesis we will utilize a strongly unforgeable signature scheme that resists one-time chosen-message attacks. This requirement is formalized by the following experiment.

```
$\mathrm{Exp}_{(\mathcal{A}_1,\mathcal{A}_2)}^{\text{sEUF-OT-CMA}}(\lambda)$:
  $(vk, sk) \leftarrow \mathsf{SgKg}(1^\lambda)$
  $(m, state) \leftarrow \mathcal{A}_1(vk)$
  $\sigma \leftarrow \mathsf{SgSign}(sk, m)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}_2(\sigma, state)$
  return $(\mathsf{SgVerify}(vk, m^*, \sigma^*) = \top) \wedge ((m^*, \sigma^*) \neq (m, \sigma))$
```

Definition 2.7. A signature scheme $(\mathsf{SgKg}, \mathsf{SgSign}, \mathsf{SgVerify})$ is strongly unforgeable under one-time chosen-message attack if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$ the advantage $\mathrm{Adv}^{\text{sEUF-OT-CMA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = \Pr[\mathrm{Exp}^{\text{sEUF-OT-CMA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1]$ is negligible in λ.

For simplicity, we sometimes call a signature scheme that is strongly unforgeable under one-time chosen-message attack as a strongly-unforgeable one-time signature.

2.3 Group Signatures

2.3.1 Brief History of Models for GS Schemes

Since its introduction, GS has enjoyed a fair amount of interest, leading to a number of concrete schemes being proposed, but also resulting in many kinds of security requirements of GS schemes being considered, e.g. unforgeability, exculpability, traceability, coalition resistance, framing resistance, anonymity, and unlinkability [CvH91, ACJT00, AT99, CP95]. Therefore, there was a need to consolidate the security requirements of GS schemes, and in 2003, Bellare, Micciancio, and Warinschi [BMW03] (BMW) showed that their formulations of full-anonymity and full-traceability are strong enough to capture all the above security requirements. In addition, they gave a concrete GS scheme in their model based on a simulation-sound non-interactive zero-knowledge proof system and an enhanced trapdoor permutation. Although the BMW model has made a great contribution towards the modeling of GS, it has the shortcoming of only considering the static group setting, i.e., the number of members is decided in the initial setup phase, and new members cannot be added later. In addition, since the group manager (GM) in the BMW model generates all secret signing keys, the GM can construct a signature such that the opening procedure identifies an honest user as the signer, even though this user never signed the given message, and there is furthermore no way for a user to verify whether the claimed result of an opening performed by the GM is true or not. To address this, Bellare, Shi, and Zhang [BSZ05] proposed a more general functionality of GS schemes which allows users to join dynamically and the correctness of an opening result to be publicly verifiable, and defined a corresponding set of security requirements (anonymity, traceability, and non-frameability) which are strong enough to capture all existing security requirements.

2.3.2 Definitions of the BSZ Model

In the BSZ model, the GM is split into two separated entities: the issuing manager (IM) and the opening manager (OM). The IM is provided with an issuer key ik, and is responsible for running the interactive joining algorithm with a user who, at the end of the interaction, will obtain a group signing key which can be used to sign messages on behalf of the group. The OM is provided with an opening key ok, and is responsible for running the opening algorithm which reveals which group member constructed a given signature. The opening algorithm furthermore provides a proof which shows that the given group member was indeed the signer, and which is verifiable through a new algorithm called Judge. This functionality is important to handle a situation in which the OM might be corrupted or malicious. In the following, we introduce the formal definition of a GS scheme in the BSZ model.

In this section, we briefly review the model and the security notions of group signatures, presented by Bellare, Shi, and Zhang [BSZ05]. A group signature scheme consists of the following seven algorithms:

GKg**:** This is the group key generation algorithm which, on input 1^k, returns the keys (gpk, ik, ok), where gpk is a group public key, ik is an issuing key, and ok is an opening key.

UKg**:** This is the user key generation algorithm which, on input gpk, returns a personal public and private key pair (upk, usk). Each user i will generate a personal key pair (upk_i, usk_i) before engaging in the joining protocol which is described below. It is assumed that anyone can obtain an authentic copy of the public key of any user. (This might be implemented via a standard public key infrastructure.)

$\mathsf{Join}/\mathsf{Issue}$**:** This is the pair of interactive algorithms which implement the joining protocol run by a user i and the issuer. The algorithm Join, which is run by the user, takes (gpk, upk, usk) as input, whereas Issue, which is run by the issuer, takes (gpk, upk, ik)

as input. Upon successful completion of the protocol, Join outputs a private signing key gsk_i for user i, and Issue outputs the registration information of user i which is stored in $\mathsf{reg}[i]$, where reg is a registration table maintained by the issuer.

GSig: This is the group signing algorithm run by a user i, which, on input gpk, a signing key gsk_i, and a message m, returns a group signature Σ.

GVf: This is the group signature verification algorithm which, on input (gpk, m, Σ), returns 1 to indicate that Σ is a valid signature on m, or 0 otherwise.

Open: This is the opening algorithm run by the opener, which, on input $(gpk, ok, \mathsf{reg}, m, \Sigma)$, returns (i, τ), where i specifies that the originator of the signature Σ is the user i, and τ is a non-interactive proof of this. In case the algorithm fails to identify the originator of the signature, it outputs $i = 0$. Note that Open requires access to the registration table reg.

Judge: This is the judge algorithm which, on input $(gpk, i, upk_i, m, \Sigma, \tau)$, outputs either 1 or 0 indicating that the proof τ is accepted as valid or invalid, respectively.

The model in [BSZ05] introduces four requirements for a group signature scheme, namely, correctness, anonymity, non-frameability, and traceability. The correctness notion requires that honestly generated signatures will be accepted as valid by the verification algorithm, can be opened by the opening algorithm, and that the judging algorithm will accept the resulting proof as valid. The anonymity notion requires that no information about the identity of a signer is leaked from a group signature, even if the signing keys of all group members and the issuer are exposed. The non-frameability notion requires that no adversary corrupting both the opener and the issuer, can produce a signature and an opening proof that identify an uncorrupted group member as the signer, when the uncorrupted group member did not produce the signature in question. The traceability notion requires that an adversary corrupting the opener and controlling a group of malicious group members, cannot produce a valid signature that cannot be opened correctly.

The formal definitions of the four notions are given as follows. We first define several oracles needed for security notions:

$\mathsf{AddU}(i)$**:** The add-user oracle runs $\mathsf{UKg}(gpk)$ and $\mathsf{Join}/\mathsf{Issue}$ protocol to add an honest user. It returns the user public key upk of the user. The oracle add i to the set HU.

$\mathsf{RReg}(i)$**:** The read-registration-table oracle reveals the content of the registration table $\mathsf{reg}[i]$.

$\mathsf{SndToU}(i, M)$**:** The send-to-user oracle at first sets up a user public/secret key pair by $(upk_i, usk_i) \leftarrow \mathsf{UKg}(gpk)$ and add i to the set HU. The oracle then allows the adversary to engage a group-joining protocol of the user i on the behalf of the corrupted issuer. The message M is sent to the user i who follows the protocol $\mathsf{Join}(gpk, upk_i, usk_i)$. The response of the user is returned to the adversary.

$\mathsf{WReg}(i, M)$**:** The write-registration-table oracle updates $\mathsf{reg}[i]$ to M.

$\mathsf{USK}(i)$**:** The user-secret-keys oracle reveals the secret keys (usk_i, gsk_i) of the user i to the adversary.

$\mathsf{CrptU}(i, M)$**:** The corrupt-user oracle sets the user public key of the user i to M and add i to the set CU.

$\mathsf{Open}(m, \Sigma)$**:** The open oracle returns the opening $(i, \tau) \leftarrow \mathsf{Open}(gpk, ok, m, \Sigma)$ of the signature Σ under the message m.

$\mathsf{Ch}_b(m, i_0, i_1)$**:** The challenge oracle returns a challenge $\Sigma^* \leftarrow \mathsf{GSig}(gpk, gsk_{i_b}, m)$. The users i_0 and i_1 needs to be in the set HU.

$\mathsf{GSig}(i, m)$**:** The signing oracle returns a signature $\Sigma \leftarrow \mathsf{GSig}(gpk, gsk_i, m)$ on the message m of the user i, who needs to be in the set HU.

$\mathsf{SndToI}(i, M)$**:** The send-to-issuer oracle allows the adversary to engage a group-joining protocol on behalf of the corrupted user i. The message M is sent to the issuer who follows the protocol $\mathsf{Issue}(gpk, upk_i, ik)$. The response of the issuer is returned to the adversary. If the protocol is successfully completed, the output of Issue will be recorded at $\mathsf{reg}[i]$. The user i needs to be in the set CU.

The correctness and security requirements for a group signature scheme are as follows:

Definition 2.8. A group signature scheme is said to have *correctness* if

$$\begin{aligned}&\Pr[(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k); (i, m) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{RReg}}(gpk);\\ &\Sigma \leftarrow \mathsf{GSig}(gpk, gsk_i, m); (j, \tau) \leftarrow \mathsf{Open}(gpk, ok, \mathsf{reg}, m, \Sigma)\\ &\qquad : \mathsf{GVf}(gpk, m, \Sigma) = 0 \vee i \neq j \vee \mathsf{Judge}(gpk, i, upk_i, m, \Sigma, \tau) = 0]\end{aligned}$$

is negligible in k for any adversary $\mathcal{A}$.

Definition 2.9. A group signature scheme is said to have *anonymity* if

$$\begin{aligned}&\Pr[b \leftarrow \{0, 1\}; (gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k);\\ &\qquad b' \leftarrow \mathcal{A}^{\mathsf{SndToU},\mathsf{WReg},\mathsf{USK},\mathsf{CrptU},\mathsf{Open},\mathsf{Ch}_b}(gpk, ik) : b = b'] - \frac{1}{2}\end{aligned}$$

is negligible in k for any probabilistic polynomial-time adversary $\mathcal{A}$.

Definition 2.10. A group signature scheme is said to have *non-frameability* if

$$\begin{aligned}&\Pr[(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k); (m, \Sigma, i, \tau) \leftarrow \mathcal{A}^{\mathsf{SndToU},\mathsf{WReg},\mathsf{USK},\mathsf{CrptU},\mathsf{GSig}}(gpk, ok, ik)\\ &: \mathsf{GVf}(gpk, m, \Sigma) = 1 \wedge i \in HU \wedge \mathsf{Judge}(gpk, i, upk_i, m, \Sigma, \tau) = 1\\ &\qquad \wedge \mathcal{A} \text{ queried neither } \mathsf{USK}(i) \text{ nor } \mathsf{GSig}(i, m)]\end{aligned}$$

is negligible in k for any probabilistic polynomial-time adversary $\mathcal{A}$.

Definition 2.11. A group signature scheme is said to have *traceability* if

$$\begin{aligned}&\Pr[(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k); (m, \Sigma) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{SndToI},\mathsf{AddU},\mathsf{USK},\mathsf{RReg}}(gpk, ok);\\ &(i, \tau) \leftarrow \mathsf{Open}(gpk, ok, \mathsf{reg}, m, \Sigma) : \mathsf{GVf}(gpk, m, \Sigma) = 1\\ &\qquad \wedge (i = 0 \vee \mathsf{Judge}(gpk, i, upk_i, m, \Sigma, \tau) = 0)]\end{aligned}$$

is negligible in k for any probabilistic polynomial-time adversary $\mathcal{A}$.

2.4 Threshold Encryption

Threshold encryption is an extension of public-key encryption [DF90]. This primitive enables to store the decryption key in a distributed form among several decryption servers, in order for protection against (possibly accidental) key exposure. The following syntax and definitions are adopted from Shoup and Gennaro [SG02], Boneh, Boyen, and Halevi [BBH06], and Galindo et al. [GLF+10]. The primitive consists of the following five algorithms.

ThKg. The key-generation algorithm ThKg takes as input the security parameter 1^λ, the number n of decryption servers, and the threshold k where $1 \leq k \leq n$ and outputs a triple $(pk, vk, (sk_i)_{i \in [n]})$ where pk is the public key, vk is the verification key, and sk_i is the decryption key for the i-th decryption server. The public key ek implicitly specifies the message space $\mathcal{M}_{pk}$, which determines the set of plaintexts that can be encrypted under that public key.

ThEnc. The encryption algorithm ThEnc takes as input a public key pk and a plaintext $m \in \mathcal{M}_{pk}$ and outputs a ciphertext C.

ThDec. The partial decryption algorithm ThDec takes as input the public key pk, the index i of a server, the decryption key sk_i for the i-th server, and a ciphertext C and outputs a decryption share $\mu = (i, \hat{\mu})$.

ThVerify. The share verification algorithm ThVerify takes as input the public key pk, the verification key vk, a ciphertext C, and a decryption share μ and outputs $\top$ or $\perp$.

ThCombine. The combining algorithm ThCombine takes as input the public key pk, the verification key vk, a ciphertext C, and k decryption shares μ_1, ..., μ_k and outputs a plaintext m or $\perp$.

A threshold encryption scheme is required to satisfy the following correctness conditions. For all $\lambda \in \mathbb{N}$, any integers n and k ($1 \leq k \leq n$), any $(pk, vk, (sk_i)_{i \in [n]}) \leftarrow \mathsf{ThKg}(1^\lambda, n, k)$, it holds that

1. for any plaintext $m \in \mathcal{M}_{pk}$ and any k-subset $\{\iota_1, \ldots, \iota_k\} \subset [n]$, if we let

$C \leftarrow \mathsf{ThEnc}(pk, m)$ and $\mu_i \leftarrow \mathsf{ThDec}(pk, \iota_i, sk_{\iota_i}, C)$ for all $i \in [k]$, it holds that $\mathsf{ThCombine}(pk, vk, C, \mu_1, \ldots, \mu_k) = m$, and

2. for any C and any $\iota \in [n]$, $\mathsf{ThVerify}(pk, vk, C, \mathsf{ThDec}(pk, \iota, sk_\iota, C))$ outputs $\top$.

There are two basic security requirements for threshold encryption schemes: chosen-ciphertext security and decryption consistency. Chosen-ciphertext security is a natural threshold variant of the chosen-ciphertext security of public-key encryption. Decryption consistency requires even malicious server cannot control the result of entire decryption by submitting a maliciously generated decryption share. Formal definitions are as follows.

Chosen-ciphertext security of a threshold encryption scheme is defined by the following experiment.

```
$\mathrm{Exp}^{(k,n)\text{-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda)$:
  $b \leftarrow \{0,1\}$
  $(S, state_1) \leftarrow \mathcal{A}_1(1^\lambda, n, k)$
  $(pk, vk, (sk_i)_{i\in[n]}) \leftarrow \mathsf{ThKg}(1^\lambda, n, k)$
  $(m_0, m_1, state_2) \leftarrow \mathcal{A}_2^{\mathsf{ThDec}(pk, sk_{(\cdot)}, \cdot)}(pk, vk, (sk_i)_{i\in S}, state_1)$
  $C^* \leftarrow \mathsf{ThEnc}(pk, m_b)$
  $b' \leftarrow \mathcal{A}_3^{\mathsf{ThDec}(pk, sk_{(\cdot)}, \cdot)}(C^*, state_2)$
  return $b = b'$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ is required to output m_0 and m_1 with $|m_0| = |m_1|$. Furthermore, S output by $\mathcal{A}_1$ is required to be cardinality $k-1$, and $\mathcal{A}_3$ is required not to submit any query of the form (i, C^*), regardless of i.

Definition 2.12. A threshold encryption scheme $(\mathsf{ThKg}, \mathsf{ThEnc}, \mathsf{ThDec}, \mathsf{ThVerify}, \mathsf{ThCombine})$ is chosen-ciphertext secure if for any integer k and n $(0 \le k \le n)$ and any probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ the advantage $\mathrm{Adv}^{(k,n)}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda) = \Pr[\mathrm{Exp}^{(k,n)\text{-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda) = 1]$ is negligible in λ.

The formal definition of decryption consistency is as follows. This definition is actually the "known secret key" variant, which is originally defined by Galindo et al. [GLF+10]. The experiment for the formal definition is as follows.

```
$\mathrm{Exp}_{\mathcal{A}}^{(k,n)\text{-Consistent}}(\lambda)$:
  $(pk, vk, (sk_i)_{[n]}) \leftarrow \mathsf{ThKg}(1^\lambda, n, k)$
  $(C, ((\iota_i, \hat{\mu}_i))_{i\in[k]}, ((\iota'_i, \hat{\mu}'_i))_{i\in[k]}) \leftarrow \mathcal{A}(pk, vk, (sk_i)_{[n]})$
  return 1 if the following conditions are satisfied:
    $\iota_1, \ldots, \iota_k$ are mutually distinct
    $\mathsf{ThVerify}(pk, vk, C, (\iota_i, \hat{\mu}_i)) = \top_{\mathrm{valid}}$ for all $i \in [k]$
    $\iota'_1, \ldots, \iota'_k$ are mutually distinct
    $\mathsf{ThVerify}(pk, vk, C, (\iota'_i, \hat{\mu}'_i)) = \top_{\mathrm{valid}}$ for all $i \in [k]$
    $\mathsf{ThCombine}(pk, vk, C, ((\iota_1, \hat{\mu}_1))_{i\in[k]}) \neq \mathsf{ThCombine}(pk, vk, C, ((\iota'_1, \hat{\mu}'_1))_{i\in[k]})$
  return 0 otherwise
```

Definition 2.13. A threshold encryption scheme $(\mathsf{ThKg}, \mathsf{ThEnc}, \mathsf{ThDec}, \mathsf{ThVerify}, \mathsf{ThCombine})$ has decryption consistency with known secret keys if for any integer k and n $(1 \leq k \leq n)$ and any probabilistic polynomial-time adversary $\mathcal{A}$ the advantage $\mathrm{Adv}_{\mathcal{A}}^{(k,n)\text{-Consistent}}(\lambda) = \Pr[\mathrm{Exp}^{(k,n)\text{-Consistent}}(\lambda) = 1]$ is negligible in λ.

2.5 Target Collision-Resistant Hash Functions

A family of functions is called target collision-resistant if no algorithms, which firstly chooses an input and then is given a description of a function in the family, can find another input that produces the same output to the first input. The formal definition we need is as follows: A function generator $\mathrm{HashGen}(1^\ell)$ takes as input a security parameter and outputs a function $\mathcal{H}$. The family of functions is said to be target collision-resistant when $\Pr[(x, s) \leftarrow \mathcal{A}; \mathcal{H} \leftarrow \mathrm{HashGen}(1^\ell); x' \leftarrow \mathcal{A}(\mathcal{H}, s) : \mathcal{H}(x) = \mathcal{H}(x') \wedge x \neq x']$ is negligible for any polynomial-time algorithm $\mathcal{A}$.

2.6 Commitment

A commitment scheme consists of the three algorithms ComKg, Commit, and $\mathsf{ComVerify}$. The algorithm ComKg takes the security parameter 1^λ and outputs a parameter ck. The commitment algorithm takes as input the parameter ck and a string m, and outputs a pair (c, r), where c is the commitment string for m and r is the corresponding decommitment string. The decommitment algorithm $\mathsf{ComVerify}$ takes as input the security parameter 1^λ, a commitment string c, a string m, and a decommitment string r and outputs 0 or 1, indicating the decommitment is valid or invalid. It is required that for any ck output by $\mathsf{ComKg}(1^\lambda)$, any $m \in \{0,1\}^*$, and any (c,r) output by $\mathsf{Commit}(ck,m)$, it holds that $\mathsf{ComVerify}(ck,c,m,r) = 1$. As the hiding property it is required that for any probabilistic polynomial-time adversary $\mathcal{A}$ the probability $|\Pr[b \leftarrow \{0,1\}; ck \leftarrow \mathsf{ComKg}(1^\lambda); b' \leftarrow \mathcal{A}^{O_b}(ck) : b = b'] - 1/2|$ is negligible in λ, where the oracle O_b, when receives a pair (m_0, m_1) as input, runs $\mathsf{Commit}(ck, m_b)$ to obtain (c,r) and returns c. As the binding property it is required that for any probabilistic polynomial-time adversary $\mathcal{A}$ the probability $\Pr[ck \leftarrow \mathsf{ComKg}(1^\lambda); (c,m,r,m',r') \leftarrow \mathcal{A}(ck) : m \neq m' \wedge \mathsf{ComVerify}(ck,c,m,r) = 1 \wedge \mathsf{ComVerify}(ck,c,m',r') = 1]$ is negligible in λ.

2.7 Non-interactive Proofs

A non-interactive proof system for an NP-relation $R \in \{0,1\}^* \times \{0,1\}^*$ defining $L = \{x | (x,w) \in R \text{ for some } w\}$ consists of three algorithms (K,P,V), which satisfy the following correctness and soundness conditions: For correctness, it is required that for any security parameter $\ell \in \mathbb{N}$, any common reference string $crs \leftarrow K(1^\ell)$, and any pair $(x,w) \in R$, it holds that $V(1^\ell, crs, x, P(1^\ell, crs, x, w)) = \top$; for soundness, it is required that for any $\ell \in \mathbb{N}$ and any probabilistic polynomial-time algorithm $\mathcal{A}$, the probability $\Pr[crs \leftarrow K(1^\ell); (x,\pi) \leftarrow \mathcal{A}(1^\ell, crs) : V(1^\ell, crs, x, \pi) = \top \wedge x \notin L]$ is negligible. We will later use three types of proof systems, one which is witness indistinguishable [FS90], one which is zero-knowledge [MDSMP91, FLS90] and one which is simulation-sound zero-knowledge [Sah99].

2.8 Number-theoretic Assumptions

Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that on input 1^λ generates $gk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ where p is a prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of order p, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map, and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We call such an algorithm by an asymmetric bilinear group generator. In the case that $\mathbb{G}_1 = \mathbb{G}_2$ and $g_1 = g_2$ we abuse the notation as $gk = (p, \mathbb{G}, \mathbb{G}_T, g, e)$ and call this bilinear group generator as a *symmetric* bilinear group generator.

Occasionally we will employ groups without efficiently computable bilinear maps. For such groups we further assumes that we can generate such a group with specified order. Formally, we assume that we have a probabilistic polynomial-time algorithm $\mathcal{G}_{\mathrm{ddh}}$ which on input a prime p outputs a pair $(\mathbb{G}, \hat{g})$ of (a description of) a group with order p and a generator of the group. We call such an algorithm by a group generator.

On these group generators, we state the following complexity-theoretic assumptions.

Definition 2.14. We say that the decision linear assumption on a symmetric bilinear group generator $\mathcal{G}$ if for any probabilistic polynomial-time algorithm $\mathcal{A}$, we have that $|\Pr[gk = (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda); u, v \leftarrow \mathbb{G}; \alpha, \beta \leftarrow \mathbb{Z}_p : \mathcal{A}(gk, u, v, g, u^\alpha, v^\beta, g^{\alpha+\beta}) = 1] - \Pr[gk = (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda); u, v \leftarrow \mathbb{G}; \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p : \mathcal{A}(gk, u, v, g, u^\alpha, v^\beta, g^\gamma) = 1]|$ is negligible in λ.

Definition 2.15. We say that the decision bilinear Diffie-Hellman assumption on a symmetric bilinear group generator $\mathcal{G}$ if for any probabilistic polynomial-time algorithm $\mathcal{A}$, we have that $|\Pr[gk = (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda); \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p : \mathcal{A}(gk, g^\alpha, g^\beta, g^\gamma, e(g, g)^{\alpha\beta\gamma}) = 1] - \Pr[gk = (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(1^\lambda); \alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p : \mathcal{A}(gk, g^\alpha, g^\beta, g^\gamma, e(g, g)^\delta) = 1]|$ is negligible in λ.

2.9 Useful Constructions of Cryptographic Primitives

2.9.1 Kiltz's Tag-based Encryption Scheme

In this paper we use Kiltz's construction of tag-based encryption [Kil06], which is explained below. The scheme can be built on bilinear groups. Let $gk = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a group description. The key generation algorithm chooses random integers $\phi, \eta \leftarrow \mathbb{Z}_p$ and random elements $K, L \leftarrow \mathbb{G}$, and sets $pk = (F, H, K, L)$ where $F = g^\phi$ and $H = g^\eta$ and $dk = (\phi, \eta)$. A ciphertext of a plaintext m under a tag t is computed as $y = (y_1, y_2, y_3, y_4, y_5) = (F^r, H^s, mg^{r+s}, (g^t K)^r, (g^t L)^s)$. The decryption algorithm decrypts a ciphertext $(y_1, y_2, y_3, y_4, y_5)$ under a tag t by checking $e(F, y_4) = e(y_1, g^t K)$ and $e(H, y_5) = e(y_2, g^t L)$ and outputs $y_3 / y_1^\phi y_2^\eta$

if the two equations hold, otherwise outputs $\perp$. This encryption scheme is secure against selective-tag weak chosen-ciphertext attacks if the decisional linear assumption holds [Kil06]. Another interesting property is that the scheme has public verifiability in the sense that it can be efficiently checked whether a given five-tuple $(y_1, y_2, y_3, y_4, y_5)$ lies in the range of the encryption algorithm under a given public key pk and a given tag t by checking the two equations $e(F, y_4) = e(y_1, g^t K)$ and $e(H, y_5) = e(y_2, g^t L)$.

2.9.2 BB and BBS+ signatures

Here we describe the BB signature, the BBS+ signature, and their related definitions, which are used as important components in the proposed constructions presented in the following section.

Definition 2.16. Bilinear groups are a tuple $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ such that $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, $g \in \mathbb{G}$ is a generator of $\mathbb{G}$, and e is an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: for all g, g', h, $h' \in \mathbb{G}$, $e(gg', h) = e(g, h)e(g', h)$ and $e(g, hh') = e(g, h)e(g, h')$, and $e(g, g)$ is not the unit of $\mathbb{G}_T$.

The description of the BB signature [BB08] is as follows:

$\mathsf{SgKg}_{\mathrm{BB}}(1^\kappa)$**:** Choose $X \in_R \mathbb{Z}_p$ and $\tilde{g} \in \mathbb{G}$, computes $Y = \tilde{g}^X$, and outputs a verification key $vk = (\tilde{g}, Y) \in \mathbb{G}^2$ and a private signing key $sigk = X$.

$\mathsf{SgSign}_{\mathrm{BB}}(pk, sk, M)$**:** Output a signature $F = \tilde{g}^{\frac{1}{X+M}}$.

$\mathsf{SgVerify}_{\mathrm{BB}}(pk, F)$**:** Check whether $e(F, Y\tilde{g}^M) \overset{?}{=} e(g, g)$.

In our RPKE scheme, to prove $M \in [1, N]$ we apply the BB signature whose signatures are represented as $\mathsf{SgSign}_{\mathrm{BB}}(1)$, $\mathsf{SgSign}_{\mathrm{BB}}(2)$, …, $\mathsf{SgSign}_{\mathrm{BB}}(N)$. To prove the knowledge of a BB signature $F_k = \mathsf{SgSign}_{\mathrm{BB}}(k)$ is as follows: Let $g_5 \in \mathbb{G}$ (we use g_5 for the same purpose in our RPKE, and therefore for the sake of clarity we use it here). Choose $\beta \in \mathbb{Z}_p$ and compute $C = F_k g_5^\beta$. Then, C satisfies the relation: $e(C, Y)/e(\tilde{g}, g) = e(g_5, Y)^\beta e(g_5, g)^\theta / e(C, g)^k$ where $\theta = \beta k$. Therefore, we prove the knowledge of DLs β and θ to prove the knowledge of $\mathsf{SgSign}_{\mathrm{BB}}(k)$. This relation is appeared in the REnc algorithm of our RPKE scheme for the relations of C_2, C_3, C_4, and C_5. Note that, for C_3 and C_5, we use the notation $\dot{g}$ instead of $\tilde{g}$ in our RPKE scheme.

The BBS+ Signature [ASM06, BBS04, FI06] can be described as follows:

$\mathsf{SgKg}_{\mathrm{BBS+}}(1^\kappa, L)$ Choose $X \in_R \mathbb{Z}_p$ and $g, g_1, \ldots, g_{L+1} \in \mathbb{G}$, where L is the length of messages. Computes $Y = g^X$, and outputs a verification key $vk = (g, g_1, \ldots, g_{L+1}, Y) \in \mathbb{G}^{L+3}$ and a private signing key $sigk = X$.

$\mathsf{SgSign}_{\mathrm{BBS+}}(pk, sk, (m_1, \ldots, m_L))$ Choose $y, z \in_R \mathbb{Z}_p$, compute $B = (g_1^{m_1} \cdots g_L^{m_L} g_{L+1}^y g)^{\frac{1}{X+z}}$, and output a signature (B, y, z).

$\mathsf{SgVerify}_{\mathrm{BBS+}}(pk, (B, y, z))$ Check whether $e(B, Yg^z) \stackrel{?}{=} e(g_1^{m_1} \cdots g_L^{m_L} g_{L+1}^y, g)$.

In our RPKE scheme, to restrictive message space, we apply BBS+ signature with $L = 3$ whose signatures are represented as $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_0, m_1)$, $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_1, m_2)$, $\ldots$, $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_r, m_{r+1})$, where $(m_1, m_2, \ldots, m_r)$ are prohibited messages, $(m_0, m_{r+1}) = (0, N+1)$, and t is the serial number. To prove the knowledge of a BBS+ signature $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_j, m_{j+1}) := (B_j, y_j, z_j)$ is as follows: Let $g_5 \in \mathbb{G}$. Choose $\alpha \in \mathbb{Z}_p$ and compute $C = B_j g_5^\alpha$. Then, C satisfies the relation:

$$\frac{e(g_5, Y)^\alpha e(g_5, g)^\zeta e(g_1, g)^t e(g_2, g)^{m_j} e(g_3, g)^{m_{j+1}} e(g_4, g)^{y_j}}{e(C, g)^{z_j}} = \frac{e(C, Y)}{e(g, g)}$$

where $\zeta = \alpha z_j$. Therefore, we prove the knowledge of DLs α, ζ, m_j, m_{j+1}, y_j, and z_j to prove the knowledge of $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_j, m_{j+1})$. Note that proving of the knowledge of t is not necessary, since t is just used as a serial number in our RPKE scheme. This relation is appeared in the REnc algorithm of our RPKE scheme for the relation of C_1.

2.9.3 Groth-Sahai Proofs

Groth and Sahai [GS08] introduced a framework for very efficient non-interactive proof for the satisfiability of relations in bilinear groups, including pairing product equations. The proof system consists of algorithms $(K_{\mathrm{NI}}, P, V, X)$. The algorithm $K_{\mathrm{NI}}(gk)$ takes a group parameter gk as input and outputs (crs, xk), where crs is a common reference string and xk is a trapdoor extraction key for extracting a witness from a proof. The algorithm $P(crs, x, w)$

outputs a proof π for an equation described by x whose witness is w. A proof π is verified by running $V(crs, x, \pi)$. The algorithm $X_{xk}(x, \pi)$ extracts a witness from the proof π which passes the verification algorithm.

There are two types of Groth-Sahai proof systems, $(K_{\mathrm{NI}}, P_{\mathrm{NIWI}}, V_{\mathrm{NIWI}}, X)$ and $(K_{\mathrm{NI}}, P_{\mathrm{NIZK}}, V_{\mathrm{NIZK}}, X)$, which respectively provide witness-indistinguishability and zero-knowledge. The two types of proof systems have identical common reference string generation algorithms, and can share a single common reference string. Furthermore, there are two types of reference strings: one yields perfect soundness, and the other yields perfect witness indistinguishability or perfect zero-knowledge, depending on the type of proof system. For further details see [GS08].

The proof system has two types of the common reference string, the soundness string and the witness-indistinguishability string. For both types the string consists of three vectors $\vec{f_1}$, $\vec{f_2}$, and $\vec{f_3}$ of $\mathbb{G}^3$, in which $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$ with random f_1, $f_2 \in \mathbb{G} \setminus \{1\}$ for both types. For the soundness string, the last vector $\vec{f_3}$ is set to $\vec{f_3} = \vec{f_1}^{\zeta_1} \vec{f_2}^{\zeta_2}$, whereas for the witness-indistinguishability string, it is set to $\vec{f_3} = \vec{f_1}^{\zeta_1} \vec{f_2}^{\zeta_2}(1, 1, g)^{-1}$. On the soundness string, the Groth-Sahai proof system provides *perfect soundness* of the proof system, while on the witness-indistinguishability string the proof system can provide a *zero-knowledge* simulation for certain types of a statement (that include the statement that we used in this paper).

Instead, the common reference string can be seen as eight group elements $crs = (F, H, U, V, W, U', V', W')$. It should be noted that F and H essentially serve as a public key of a linear encryption scheme [BBS04]. This property is exploited in the Groth group signature scheme (and therefore also in our modification of that scheme). For further details see [GS08].

2.9.4 Σ-protocol [Dam, Cra96]

Let $R \subset \{0,1\}^* \times \{0,1\}^*$ be a binary relation. For $(x, \omega) \in R$, we call ω is a witness of x. We assume that the following 3-round form, where x is common input of a prover P and a verifier V, and ω (such that $(x, \omega) \in R$)) is private input to P. First, P sends a message a to V. V sends a random bit string e'. Finally, P sends a reply z, and V decides whether the proof is accepted or not. We say that a 3-round protocol $\langle P, V \rangle$ is a Σ-protocol for relation R if the following hold:

Completeness: If P and V follow the protocol, then V always accepts.

Special soundness: From any common input x and any pair of accepting conversations on input x, (a, e', z) and (a, e'', z'') where $e' \neq e''$, one can efficiently compute ω such that $(x, \omega) \in R$.

Special honest verifier zero-knowledge: There exists a polynomial-time simulator, which on input x and a random challenge string e', outputs an accepting conversation of the form (a, e', z), with the same probability distribution as conversations between the honest P, V on input x.

In our RPKE construction, we convert the underlying Σ-protocol into NIZK proof of knowledge by applying Fiat-Shamir heuristic [FS87]. Therefore, we require random oracles in our construction. We denote such a converted proof as $NIZK\{\omega : (x, \omega) \in R\}$ where x is an instance of the language and ω is its witness.

Part II

Applying Public-key Encryption with Non-interactive Opening to Secure Group Signature and Threshold Encryption

Chapter 3

Opening Soundness of Dynamic Group Signature Schemes

In this chapter, we propose a new security notion for the group signature primitive, which we call *opening soundness*. As discussed below, this security notion captures practical threats that will occur in typical use cases of group signature.

The contribution of this chapter falls into the category (II) of our contribution discussed in Sect. 1.1.3. Namely, the opening soundness notion is obtained by translating the *committing* notion for PKENO to a notion for group signature, through a generic construction of group signature using public-key encryption (together with other primitives).

3.1 Introduction

Group signatures, introduced by Chaum and van Heyst [CvH91], allow a group member to anonymously sign a message on behalf of the group. More specifically, anyone will be able to verify that a signature originates from a group member, but the signature does not reveal the identity of the signer, not even to other members of the group. Group membership is controlled by an authority called the issuer, who handles enrollment of users through an interactive join protocol. To prevent misuse of the signing capabilities obtained by group members, another authority called the opener can revoke the anonymity of a signature and identify the signer of the message.

Following the introduction of group signatures, a series of different security requirements were proposed for this primitive, each of which aims at addressing a specific security concern by augmenting or refining previous notions, e.g. unforgeability, exculpability, traceability, coalition resistance, framing resistance, anonymity and unlinkability. These security notions were later consolidated in the security model proposed by Bellare, Micciancio, and Warinschi [BMW03] who introduce two strong security requirements, full-anonymity and full-traceability, which imply all of the previously proposed notions of security.

However, a drawback of the model by Bellare, Micciancio, and Warinschi [BMW03] is that only *static* group signature schemes are considered i.e. the set of group members is fixed, and the private key material of each group member is generated in the setup phase of the scheme. Furthermore, the authority controlling the group (which acts as both the issuer and opener) is considered to be fully trusted. To address this, Bellare, Shi, and Zhang [BSZ05] extended the model of [BMW03] to capture *dynamic* group signature schemes in which a user can dynamically join the group by engaging in a join protocol with the issuer. Furthermore, to reduce trust in the opener, the model adopts the approach by Camenisch and Michels [CM98], and requires that the opener produces a non-interactive and publicly verifiable proof that a given signature was produced by a given signer. The model introduces three formal security notions: anonymity, traceability, and non-frameability. The former two notions are adaptations of the full-anonymity and full-traceability notions to the dynamic group signature setting. The latter notion, non-frameability, requires that even if a malicious opener and issuer collude, they cannot frame an honest user by producing a signature and corresponding opening which identify the honest user as the signer, when the honest user did not produce the signature in question.

Limitations of Non-Frameability. While non-frameability is a strong security notion, it only partly covers the security properties one would intuitively expect to gain when the opener is required to produce a non-interactive and publicly verifiable proof of an opening. More specifically, the non-frameability notion only ensures that the opener cannot frame an *uncorrupted* user by constructing a proof that the user is the signer of a signature he did not produce. However, no guarantee is given regarding an opening involving a *corrupted* user. This leaves open the possibility that an opening showing that a malicious or corrupted user is the signer of a signature produced by an honest user, can be constructed. Furthermore, this might not require the opener to be corrupted or malicious, in which case a malicious user might be able to independently forge a proof showing that he is the signer of any signature of his choice.

Depending on the concrete scenario in which a dynamic group signature scheme is used, the ability to forge an opening proof might become a real security concern. We highlight several potential threats that this ability gives rise to:

- ***Signer impersonation.*** The most obvious threat is signer impersonation. This is a problem if a group signature scheme is used for an anonymous auction as suggested in [ACJT00]. In this scenario, the bidders correspond to group members, and when submitting a bid, a group member will attach a group signature on his bid. The opener serves as the auctioneer, and will make the opening of the signature on the highest bid public. This will enable anyone to verify who the winner of the auction is. However, a malicious bidder may forge a proof of ownership of the signature on the highest bid and may insist that he/she is the winner.
  A similar situation occurs if a dynamic group signature scheme is used to implement an authentication scheme with identity escrow [KP98]. In this case, a malicious group member can claim to be the user who authenticated himself to a server (and provide a proof thereof) when this is not the case.
- ***Proxy confession.*** The ability to open a group signature is introduced to keep the group members accountable of the messages signed on behalf of the group. However, assume that a signature on some message causes a dispute, but the real signer wants to avoid being blamed for this. Then the real signer asks (or intimidates) another group member to forge a proof of ownership of the signature and take the blame.
- ***Key exposure.*** Consider the case in which a group member's private key is exposed and falls into the hands of a malicious user. This will not only allow the malicious user to construct future signatures on any message of this choice, but will furthermore

allow him to claim (and prove) that the original user is the signer of any *previously generated* signature.

Our Contribution. We highlight the above described potential weakness of the security guarantee provided by the formal model of Bellare, Shi, and Zhang [BSZ05]. Furthermore, we show that this is not only a property of the security model, but that the most efficient dynamic group signature schemes enable a malicious group member to forge a proof of ownership of a signature.

To address this, we propose a new security notion for dynamic group signatures which we denote *opening soundness*. We consider two variants of this notion, weak opening soundness and (ordinary) opening soundness. The former is intended to address the above highlighted security threats in an intuitive and straightforward manner, and will rule out the possibility that a malicious group member can produce a proof of ownership of a signature generated by an honest user. The latter considers a stronger adversary who has access to the private key of the opener, and who is only required to produce two different openings of a maliciously constructed signature. The notion of opening soundness implies the notion of weak opening soundness.

As a positive result, we prove that the generic construction of a dynamic group signature scheme by Bellare, Shi, and Zhang [BSZ05] achieves opening soundness. We furthermore propose a modification of the scheme by Groth [Gro10] which allows us to prove opening soundness of the modified scheme. In contrast, we show that the original scheme does not provide weak opening soundness. In addition, we briefly discuss opening soundness of the random oracle scheme [FI05, BCN$^+$10]. A summary of our results regarding opening soundness of the above mentioned schemes can be seen in Table 3.1.

Related Work. Since the first proposal of group signature by Chaum and van Heyst, many efficient constructions have been proposed, most of which are relying on the random oracle model [ACJT00, BBS04, CL04, KY05, FI05, DP06, BCN$^+$10]. Many initial schemes were based on the strong-RSA assumption. The first group signature schemes based on assumptions of the discrete-logarithm type were achieved independently by Camenisch and

Table. 3.1 Summary of the results. The mark "?" means it is an open question whether the scheme has the given property or not. The rightmost column denotes the section in which the security of the corresponding scheme is discussed.

	Opening Soundness		
	(Ordinary)	Weak	
Our Variant of [Gro10]	Yes	Yes	(§3.4.1)
Bellare-Shi-Zhang [BSZ05]	Yes	Yes	(§3.3)
Furukawa-Imai [FI05]	No	?	(§3.3)
Bichsel et al. [BCN+10]	No	?	(§3.3)
Groth (full version) [Gro10]	No	No	(§3.3)

Lysyanskaya [CL04], and Boneh, Boyen, and Shacham [BBS04]. The former scheme is based on the LRSW assumption, while the latter is based on the q-strong Diffie-Hellman assumption. Kiayias, Tsiounis, and Yung proposed the notion of traceable signature [KTY04], which can be seen as an extension of group signature with additional anonymity-revocation functionalities. One of these functionalities is that of allowing a group member to claim the authorship of a signature, however, its security requirement does not care about the possibility in which a malicious member falsely claims the authorship of an honestly generated signature by another.

Constructions which are provably secure without random oracles were only recently achieved. Besides the generic construction relying on non-interactive zero-knowledge (NIZK) proofs for general NP languages, Groth constructed the first concrete group signature scheme with constant signature size by exploiting the properties of bilinear groups [Gro06], though signatures are extremely large. Boyen and Waters proposed group signature schemes [BW06, BW07] whose signature sizes are quite compact. In particular the latter scheme has signatures consisting only of six group elements of a composite order group. The drawback of these schemes is that they only achieve weaker security guarantees, that is, they only provide so called CPA-anonymity in the security model of Bellare, Micciancio, and Warinschi [BMW03]. Groth proposed another group signature scheme [Gro07, Gro10] which has constant signature size (roughly one or two kilobytes) and which is provably secure in the dynamic group signature model of Bellare, Shi, and Zhang [BSZ05] without relying on random oracles.

3.2 Opening Soundness

In this section we give a formal definition of opening soundness. Specifically, we introduce two variants of opening soundness, weaker and stronger definitions.

The weaker definition, named weak opening soundness, is intended to address the security concerns discussed in the introduction in a straightforward manner, and will rule out the possibility that a malicious user can claim ownership of a signature produced by an honest user by forging an opening proof. The definition is as follows:

Definition 3.1. A group signature scheme is said to have *weak opening soundness* if

$$\begin{aligned}&\Pr[(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k); (m, i, i^*, s) \leftarrow \mathcal{A}^{\mathsf{AddU}(\cdot)}(gpk);\\ &\Sigma \leftarrow \mathsf{GSig}(gpk, gsk_i, m); \tau^* \leftarrow \mathcal{A}^{\mathsf{AddU}(\cdot)}(s, \Sigma, gsk_{i^*})\\ &\qquad\qquad : i \neq i^* \wedge i, i^* \in \mathsf{HU} \wedge \mathsf{Judge}(gpk, i^*, upk_{i^*}, m, \Sigma, \tau^*) = 1]\end{aligned}$$

is negligible for all polynomial time adversaries $\mathcal{A}$, where the oracle AddU is defined as follows:

AddU**:** On a query $i \in \mathbb{N}$, the oracle runs $(upk_i, usk_i) \leftarrow \mathsf{UKg}(gpk)$, then executes the protocol $(gsk_i, \mathsf{reg}_i) \leftarrow \langle \mathsf{Join}(gpk, upk_i, usk_i), \mathsf{Issue}(gpk, ik)\rangle$, adds i to a set $\mathcal{HU}$, and lastly returns upk_i.

Note that the adversary is only allowed to receive the secret signing key of a single user i^*. Hence, this definition will not rule out attacks involving a corrupted opener, and therefore cannot contribute towards reducing trust in this entity.

In contrast, the stronger definition, named opening soundness, is intended to rule out the possibility that an adversary can produce two different openings of a signature, even if he is allowed to corrupt the opener and all the users in the system, and furthermore generate the signature in question maliciously. The definition is as follows:

Definition 3.2. A group signature scheme is said to have *opening soundness* if

$$\begin{aligned}&\Pr[(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^k); (m, \Sigma, i_1, \tau_1, i_2, \tau_2) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{WReg}}(gpk, ok, ik)\\&: \mathsf{GVf}(gpk, m, \Sigma) = 1 \land i_1 \neq i_2 \land \mathsf{Judge}(gpk, i_1, upk_{i_1}, m, \Sigma, \tau_1) = 1\\&\qquad\qquad\qquad\qquad\qquad\qquad \land \mathsf{Judge}(gpk, i_2, upk_{i_2}, m, \Sigma, \tau_2) = 1]\end{aligned}$$

is negligible for all polynomial time adversaries $\mathcal{A}$, where the oracle $\mathsf{CrptU}(i, M)$ sets the user public key of the user i to be M, and the oracle $\mathsf{WReg}(i, M)$ sets $\mathsf{reg}[i]$ to M.

While the weaker definition provides a minimum level of protection against the type of attacks described in the introduction, we believe that, when applied to the scenarios mentioned in the introduction, any dynamic group signature scheme should provide (ordinary) opening soundness to prevent any type of attack which exploits ambiguity of openings, or involves a corrupted opener. Furthermore, we will show that this level of security can be achieved efficiently by showing that our modified version of the scheme by Groth provides opening soundness (See Sect. 3.4 for details).

3.3 Opening Soundness of Existing Schemes

We will now take a closer look at some of the existing dynamic group signature schemes, and highlight the level of opening soundness (ordinary, weak or none) achieved by these. Note that since the Bellare-Shi-Zhang security model for dynamic group signatures does not considers opening soundness, a security proof in this model will not allow us to make any conclusions regarding the opening soundness of existing schemes.

In this section, we will focus on the standard model scheme by Groth described in [Gro10] (note that the updated scheme in [Gro10] is slightly different from the scheme described in [Gro07]) and the generic construction of a dynamic group signature scheme by Bellare, Shi, and Zhang [BSZ05]. More specifically, we will show that the scheme by Groth does not have weak opening soundness whereas the generic construction by Bellare, Shi and Zhang has opening soundness. We further show that the random oracle model schemes by Furukawa and Imai [FI05] and Bichsel et al. [BCN$^+$10] do not have opening soundness. Interestingly, while these schemes do not provide opening soundness, there seems to be no obvious attack against the weak opening soundness of these.

The Groth Scheme. Figure 3.1 shows a description of the Groth scheme. Below, we will expand on the description given in the figure. However, before discussing the implementation

```
GKg($1^k$):
$gk \leftarrow \mathcal{G}(1^k)$; $\mathcal{H} \leftarrow \mathrm{HashGen}(1^k)$
$(f, h, z) \leftarrow \mathbb{G}$; $T = e(f, z)$
$(crs, xk) \leftarrow K_{\mathrm{NI}}(gk)$
$(F, H, U, V, W, U', V', W') \leftarrow crs$
$K, L \leftarrow G$; $pk \leftarrow (F, H, K, L)$
$(gpk, ik, ok)$
    $\leftarrow ((gk, \mathcal{H}, f, h, T, crs, pk), z, xk)$

Join/Issue(User $i$: $gpk$; Issuer: $gpk, ik$):
Run the coin-flipping protocol in [Gro10]
    The user obtains $v_i = g^{x_i}$ and $x_i$
    and the issuer obtains $v_i$
Issuer: $r \leftarrow \mathbb{Z}_p$
        $(a_i, b_i) \leftarrow (f^{-r}, (v_i h)^r z)$
        set reg[$i$] $\leftarrow v_i$
        send $(a_i, b_i)$ to the user
User: If $e(a_i, hv_i)e(f, b_i) = T$,
        set $gsk_i \leftarrow (x_i, a_i, b_i)$

Open($gpk, ok$, reg, $m, \Sigma$):
$(b, v, \sigma)$
    $\leftarrow X_{xk}(crs, (gpk, a, \mathcal{H}(vk_{\mathrm{sots}})), \pi)$
Return $(i, \sigma)$ if there is $i$ so $v =$ reg[$i$],
else return $(0, \sigma)$

GSig($gpk, gsk_i, m$):
$(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow \mathrm{KeyGen}_{\mathrm{sots}}(1^k)$
    (Repeat until $\mathcal{H}(vk_{\mathrm{sots}}) \neq -x_i$)
$\rho \leftarrow \mathbb{Z}_p$; $a \leftarrow a_i f^{-\rho}$; $b \leftarrow b_i(hv_i)^\rho$
$\sigma \leftarrow g^{1/(x_i+\mathcal{H}(vk_{\mathrm{sots}}))}$
$\pi \leftarrow P_{\mathrm{NIWI}}(crs, (gpk, a, \mathcal{H}(vk_{\mathrm{sots}})), (b, v_i, \sigma))$
$y \leftarrow E_{pk}(\mathcal{H}(vk_{\mathrm{sots}}), \sigma)$
$\psi \leftarrow P_{\mathrm{NIZK}}(crs, (gpk, y, \pi), (r, s, t))$
$\sigma_{\mathrm{sots}} \leftarrow \mathrm{Sign}_{sk_{\mathrm{sots}}}(vk_{\mathrm{sots}}, m, a, \pi, y, \psi)$
Return $\Sigma = (vk_{\mathrm{sots}}, a, \pi, y, \psi, \sigma_{\mathrm{sots}})$

GVf($gpk, m, \Sigma$):
Return 1 if the following holds:
    $1 = \mathrm{Ver}_{vk_{\mathrm{sots}}}((vk_{\mathrm{sots}}, m, a, \pi, y, \psi), \sigma_{\mathrm{sots}})$,
    $1 = V_{\mathrm{NIWI}}(crs, (gpk, a, \mathcal{H}(vk_{\mathrm{sots}})), \pi)$,
    $1 = V_{\mathrm{NIZK}}(crs, (gpk, \pi, y), \psi)$, and
    $1 = \mathrm{ValidCiphertext}(pk, \mathcal{H}(vk_{\mathrm{sots}}), y)$,
else return 0

Judge($gpk, i$, reg[$i$], $m, \Sigma, \sigma$):
Return 1 if
    $i \neq 0 \wedge e(\sigma, v_i g^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$,
else return 0
```

Figure. 3.1 The Groth group signature scheme [Gro10].

details of the Groth scheme, we note that the scheme diverge slightly from the description of a dynamic group signature scheme given in the Bellare-Shi-Zhang model [BSZ05]. Specifically, in [BSZ05], a user is assumed to independently generate a public/private key pair (upk_i, usk_i), and then afterwards obtain a group signing key gsk_i by interacting with the issuer in the Join protocol. In the Groth scheme [Gro10], on the other hand, a user generates a public/private key pair *jointly* with the issuer in the Join protocol. The public key for user

i will be stored in $reg[i]$ by the issuer, and the corresponding private key will be the group signing key gsk_i of user i. This intuitively corresponds to a scheme in which the user key generation algorithm UKg is merged with the Join protocol. Note that in this setup it is assumed that user i and the issuer agree upon the content of $\mathsf{reg}[i]$. To ensure this, it is suggested in [Gro10] that the user signs the content of $\mathsf{reg}[i]$, using a separate signing key, and that an entry in reg is only considered to be valid if the content is signed by the corresponding user.

To model the security of this type of scheme, a few minor changes are required to the security model presented in Sect. 2.3. Specifically, the public key upk_i of user i is simply defined as the content of $\mathsf{reg}[i]$, and we no longer consider the write-registration-table oracle WReg in the security definitions, but only the corrupt oracle CrptU which allows the adversary to set the public key upk_i (i.e., the content of $\mathsf{reg}[i]$) to a given value, i.e., the WReg oracle is simply removed from the security definitions. Furthermore, since the issuer is the only party which can insert the public key of a user in reg, and the issuer will only do so upon successful completion of the Join protocol, we no longer consider the corrupt oracle CrptU in the traceability security definition, but only SndToI which allows the adversary to interact with the (honest) issuer in the Join protocol, i.e., the CrptU oracle is removed from the definition. Lastly, the oracles AddU and $\mathsf{SendToU}$ will no longer run the algorithm UKg since this algorithm is not defined for the scheme. With these changes, we obtain a security model equivalent to the model presented by Groth [Gro10].

We will now return to the implementation details of the Groth scheme. In the group key generation algorithm GKg, the elements f, h, T correspond to a verification key of the Zhou-Lin signature scheme [ZL06], whereas z corresponds to the signing key. Furthermore, pk is a public key of Kiltz's tag-based encryption scheme. Note that the first two elements of pk and the common reference string crs for the non-interactive Groth-Sahai proofs are identical.

In the group signing algorithm GSig, a group member constructs two non-interactive Groth-Sahai proofs. The first proof π, constructed via P_{NIWI}, shows knowledge of a signature σ, a verification key v and a part b of a (re-randomized) certificate (a, b) which satisfy $e(a, hv)e(f, b) = T \wedge e(\sigma, vg^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$. The first part a of the certificate can be safely revealed as part of the group signature since it does not leak any information about the identity of the member due to the re-randomization. The second proof ψ, constructed via P_{NIZK}, demonstrates that the plaintext of y is same as the witness σ used in π. More specifically, the tag-based encryption y has the form $(y_1, y_2, y_3, y_4, y_5) = (F^{r_y}, H^{s_y}, g^{r_y+s_y}\sigma, (g^{\mathcal{H}(vk_{\text{sots}})}K)^{r_y}, (g^{\mathcal{H}(vk_{\text{sots}})}L)^{s_y})$, while the Groth-Sahai proof π contains a commitment $c = (c_1, c_2, c_3) = (F^{r_c}U^t, H^{s_c}V^t, g^{r_c+s_c}W^t\sigma)$. The proof demonstrates that there exists (r, s, t) such that $(c_1 y_1^{-1}, c_2 y_2^{-1}, c_3 y_3^{-1}) = (F^r U^t, H^s V^t, g^{r+s}W^t)$. When y and c encrypt the same message, there exists (r, s, t) that satisfies above equation, but if y and c encrypt different messages, no such tuple (r, s, t) exists.

The verification algorithm GVf will, in addition to the verification of the two non-interactive proofs and the one-time signature, verify that the ciphertext y is a valid ciphertext, using the algorithm ValidCiphertext. This algorithm is easily implemented for the tag-based encryption scheme by Kiltz (see Sect. 2.9.1 for details).

We will now show how a malicious group member can forge a opening proof which shows that he is the signer of any signature Σ produced by user i. As described above, an opening proof consists of a certified signature σ on vk_{sots} which is part of Σ. To verify the opening proof, it is only verified that σ is a valid signature on vk_{sots} under the verification key v_i of the user in question.

Hence, a malicious user i' who wants to impersonate the signer of the group signature Σ on m, simply uses his own private signing key $x_{i'}$ to construct a new signature σ' on vk_{sots}, and publicizes this as an opening proof together with his own identity i'. This proof will be accepted by the Judge algorithm since σ' is a valid signature in vk_{sots}.

We formally state this as a theorem:

Theorem 3.3. *The Groth group signature scheme does not provide weak opening soundness.*

Proof. We describe an algorithm for producing a forged proof: When the adversary receives the security parameter 1^ℓ and a group public key gpk, it firstly issues two queries AddU(1) and AddU(2) in order to add two members 1 and 2 the group. The adversary then requests the challenge by outputting $(i, i^*, m) = (1, 2, 0^\ell)$, and receives a tuple (Σ, gsk_2), where $\Sigma = (vk_{\text{sots}}, a, \pi, y, \psi, \sigma_{\text{sots}})$ and $gsk_2 = (x_2, a_2, b_2)$. The adversary forges a proof of ownership by computing $\sigma^* = g^{1/(x_2+\mathcal{H}(vk_{\text{sots}}))}$ and outputs σ^* (Notice that vk_{sots} is taken from the group signature Σ).

One can easily verify that $\mathsf{Judge}(gpk, \mathsf{reg}, 2, m, \Sigma, \sigma^*)$ actually outputs 1, which means that the algorithm successfully breaks the opening soundness. $\square$

The Bellare-Shi-Zhang Scheme. Below, we will give an intuitive description of the generic construction of a dynamic group signature scheme by Bellare, Shi, and Zhang [BSZ05].

In the Bellare-Shi-Zhang construction, each group member i has a key pair (vk_i, sk_i) of an EUF-CMA secure signature scheme. The issuer also possesses his own key pair (ak, ck) of the signature scheme. The issuer signs the message $\langle i, vk_i\rangle$ to obtain the signature $cert_i$, and sends $cert_i$ to the user i. A group signature on a message m by the user i is a pair (C, π): here C is an encryption of $\langle i, vk_i, cert_i, s\rangle$, s is a signature on m under the key pair (vk_i, sk_i), and the NIZK proof π proves that the plaintext encrypted in C is of the form $\langle i, vk, cert, s\rangle$ and that $cert$ and s are a valid certificate on vk_i and a valid signature on the message in question, respectively. The opener attributes a group signature $\Sigma = (C, \pi)$ to the user i by providing an NIZK proof τ for another statement (i.e., different from that of π), which shows the existence of a decryption key that corresponds to the opener's public key and that under that key C is decrypted to $\langle i, vk_i, cert_i, s\rangle$.

This simple scheme provides opening soundness. Intuitively, this is due to the correctness of the public key encryption used to encrypt the signature and the certificate, and the soundness of the NIZK proof system for τ. The correctness condition of public key encryption ensures that given a public key pk and a ciphertext C, the decryption of C is determined uniquely. Now, let us assume that an adversary of the opening soundness game outputs a tuple $(m, \Sigma, i_1, \tau_1, i_2, \tau_2)$ where $\Sigma = (C, \pi)$ and wins the game. The proof τ_1 proves that C decrypts to $\langle i_1, vk, cert, s\rangle$ for some vk, $cert$, and s, whereas τ_2 proves that C decrypts to a different plaintext $\langle i_2, vk', cert', s'\rangle$ for some vk', $cert'$, and s'. However, this should not be possible since the decryption of C under a fixed public key is unique. Hence, the adversary breaks the soundness of the NIZK proof system.

We show the detailed description of the BSZ scheme in Fig. 3.2. The construction is a generic construction from a EUF-CMA secure signature scheme $(\mathsf{SKg}, \mathsf{Sign}, \mathsf{Ver})$, a IND-CCA secure public-key encryption scheme $(\mathsf{EKg}, \mathsf{Enc}, \mathsf{Dec})$, a simulation-sound zero-knowledge non-interactive proof system (K_1, P_1, V_1), and a zero-knowledge non-interactive proof system (K_2, P_2, V_2). The proof system (K_1, P_1, V_1) is for the relation

$$
\begin{aligned}
&((pk, ak, m, C), (i, vk', cert, \sigma, r)) \in R_1 \\
&\qquad \iff \mathsf{Ver}_{ak}(\langle i, vk' \rangle, cert) = 1 \wedge \mathsf{Ver}_{vk'}(m, s) = 1 \wedge \mathsf{Enc}_{pk}(\langle i, vk', cert, s \rangle; r) = C,
\end{aligned}
$$

while the proof system (K_2, P_2, V_2) is for the relation

$$
\begin{aligned}
&((pk, C, i, vk, cert, \sigma), (dk, R)) \in R_2 \\
&\qquad\qquad \iff \mathsf{EKg}(1^k; R) = (pk, dk) \wedge \mathsf{Dec}(dk, C) = \langle i, vk, cert, \sigma \rangle.
\end{aligned}
$$

The theorem on the opening soundness is stated bellow.

Theorem 3.4. *The Bellare-Shi-Zhang construction (Fig. 3.2) provides opening soundness, assuming that the non-interactive proof systems (P_1, V_1) and (P_2, V_2) provide soundness with negligible soundness error.*

Proof. Let (GKg, UKg, Join, Issue, GSig, GVf, Open, Judge) be the Bellare-Shi-Zhang construction. Let us consider an adversary $\mathcal{A}$ that is run in the environment of the opening soundness experiment, and let succ be the event that $\mathcal{A}$ breaks the opening soundness of the scheme.

We will show that the probability Pr[succ] is negligible. Toward this end we define three events invalid, non-trace$_1$, and non-trace$_2$. The event invalid is that $\mathcal{A}$ outputs $(M, \Sigma, i, \tau, i', \tau')$ such that the group signature $\Sigma = (c, \pi)$ contains a ciphertext c that has no corresponding plaintext m and randomness r which satisfy $c = E_{pk}(m; r)$. The event non-trace$_1$ denotes that, for the ciphertext c output by $\mathcal{A}$, there exists no decryption key dk that satisfies $pk = G(1^k, dk)$ and $D_{dk}(c) = \langle i_1, vk, cert, s \rangle$ for some vk, $cert$, and s, and finally non-trace$_2$ denotes the same event for i_2.

```
$\mathsf{GKg}(1^k)$:
$crs_1 \leftarrow K_1(1^\ell)$; $crs_2 \leftarrow K_2(1^\ell)$
$R \leftarrow \{0,1\}^k$; $(pk, dk) \leftarrow \mathsf{EKg}(1^k; R)$
$(ak, ck) \leftarrow \mathsf{SKg}(1^k)$
$gpk := (1^k, crs_1, crs_2, pk, ak)$
$ok := (dk, R)$; $ik := ck$
Return $(gpk, ok, ik)$
```

```
$\mathsf{UKg}(1^k)$:
$(upk, usk) \leftarrow \mathsf{SKg}(1^k)$
Return $(upk, usk)$
```

```
$\mathsf{Join}/\mathsf{Issue}$ :
$\mathsf{Join}(gpk, upk_i, usk_i)$:
  $(vk_i, sk_i) \leftarrow \mathsf{SKg}(1^k)$; $s_i \leftarrow \mathsf{Sign}_{usk_i}(vk_i)$
  Send $(vk_i, s_i)$ to the issuer
$\mathsf{Issue}(gpk, upk_i, ik)$:
  If $\mathsf{Ver}_{upk_i}(vk_i, s_i) = 1$ then
    $cert_i \leftarrow \mathsf{Sign}_{ck}(\langle i, vk_i \rangle)$
    $\mathsf{reg}[i] := (vk_i, s_i)$,
  Else $cert_i := \varepsilon$
  Send $cert_i$ to the user
User:
  $gsk_i := (i, vk_i, sk_i, cert_i)$
```

```
$\mathsf{GSig}(gpk, gsk_i, m)$:
Parse $gpk$ as $(1^k, crs_1, crs_2, pk, ak)$
Parse $gsk_i$ as $(i, vk_i, sk_i, cert_i)$
$\sigma \leftarrow \mathsf{Sign}_{sk_i}(m)$
$r \leftarrow \{0,1\}^k$; $C \leftarrow \mathsf{Enc}_{pk}(\langle i, vk_i, cert_i, \sigma \rangle; r)$
$\pi_1 \leftarrow P_1(1^k, (pk, ak, m, C),$
                 $(i, vk_i, cert_i, \sigma, r), crs_1)$
Return $\Sigma := (C, \pi)$;
```

```
$\mathsf{GVf}(gpk, \mathsf{reg}, m, \Sigma)$:
Parse $gpk$ as $(1^k, crs_1, crs_2, pk, ak)$
Parse $\Sigma$ as $(C, \pi_1)$
Return $V_1(1^k, (pk, ak, m, C), \pi_1, crs_1)$
```

```
$\mathsf{Open}(gpk, ok, \mathsf{reg}, m, \Sigma)$:
Parse $gpk$ as $(1^k, crs_1, crs_2, pk, ak)$
Parse $ok$ as $(dk, R)$
Parse $\Sigma$ as $(C, \pi_1)$
$M \leftarrow \mathsf{Dec}_{dk}(C)$
Parse $M$ as $\langle i, vk, cert, \sigma \rangle$
If $\mathsf{reg}[i] \neq \varepsilon$ then
  Parse $\mathsf{reg}[i]$ as $(vk_i, s_i)$
Else $vk_i := \varepsilon$; $s_i := \varepsilon$
$\pi_2 \leftarrow P_2(1^k, (pk, C, i, vk, cert, \sigma), (dk, R), crs_2)$
If $V_1(1^k, (pk, ak, m, C), \pi_1, crs_1) = 0$ then
  Return $(0, \varepsilon)$
If $vk \neq vk_i$ or $\mathsf{reg}[i] = \varepsilon$ then
  Return $(0, \varepsilon)$
$\tau := (vk_i, s_i, i, vk, cert, \sigma, \pi_2)$
Return $(i, \tau)$
```

```
$\mathsf{Judge}(gpk, \mathsf{reg}, i, upk_i, m, \Sigma, \tau)$:
Parse $gpk$ as $(1^k, crs_1, crs_2, pk, ak)$
Parse $\Sigma$ as $(C, \pi_1)$
If $(i, \tau) = \varepsilon$ then
  Return $V_1(1^k, (pk, ak, m, C), \pi_1, crs_1) = 0$
Parse $\tau$ as $(\bar{vk}, \bar{s}, i', vk', cert', \sigma', \pi_2)$
If $V_2(1^k, (C, i', vk', cert', \sigma'), \pi_2, crs_2) = 0$ then
  Return 0
If $i = i'$ and $\mathsf{Ver}_{upki}(\bar{vk}, \bar{s}) = 1$
  and $\bar{pk} = pk'$ then
  Return 1
Else Return 0
```

Figure. 3.2 The Bellare-Shi-Zhang group signature scheme [BSZ05].

By the union bound, we obtain an upper bound for Pr[succ] as

$$\begin{aligned}\Pr[\mathsf{succ}] \leq{} & \Pr[\mathsf{succ} \wedge \neg\mathsf{invalid} \wedge \neg\mathsf{non\text{-}trace}_1 \wedge \neg\mathsf{non\text{-}trace}_2] \\ & + \Pr[\mathsf{succ} \wedge \mathsf{invalid}] + \Pr[\mathsf{succ} \wedge \mathsf{non\text{-}trace}_1] + \Pr[\mathsf{succ} \wedge \mathsf{non\text{-}trace}_2].\end{aligned}$$

The last three terms $\Pr[\mathsf{succ}\wedge\mathsf{invalid}]$, $\Pr[\mathsf{succ}\wedge\mathsf{non\text{-}trace}_1]$, and $\Pr[\mathsf{succ}\wedge\mathsf{non\text{-}trace}_2]$ are all negligible due to the soundness of the underlying zero-knowledge proof systems which are assumed to have negligible soundness error i.e. if the event $\mathsf{invalid}$ occurs, it is straightforward to construct an algorithm which breaks the soundness of (K_1, P_1, V_1), and likewise, if either of the events $\mathsf{non\text{-}trace}_1$, or $\mathsf{non\text{-}trace}_2$ occur, it is straightforward to construct algorithms which break the soundness of (K_2, P_2, V_2).

The remaining part is to show that $\Pr[\mathsf{succ} \wedge \neg\mathsf{invalid} \wedge \neg\mathsf{non\text{-}trace}_1 \wedge \neg\mathsf{non\text{-}trace}_2]$ is negligible. This term is in fact exactly equal to zero, due to the correctness of the public key encryption scheme used. The condition $\neg\mathsf{invalid} \wedge \neg\mathsf{non\text{-}trace}_1 \wedge \neg\mathsf{non\text{-}trace}_2$ means that there are two different decryption keys dk_1 and dk_2 that correspond to the same public key pk (i.e., there are random tapes ρ_1 and ρ_2 such that $(pk, dk_1) = \mathsf{EKg}(1^k;\rho_1)$ and $(pk, dk_2) = \mathsf{EKg}(1^k;\rho_2)$) but which produce different decryption results for a single valid ciphertext c. The correctness condition requires that if a ciphertext c is honestly generated under a public key pk, two decryption keys which are different but correspond to the same public key pk, produce the same decryption results. Since the above situation contradicts this requirement, the probability $\Pr[\mathsf{succ} \wedge \neg\mathsf{invalid} \wedge \neg\mathsf{non\text{-}trace}_1 \wedge \neg\mathsf{non\text{-}trace}_2]$ is equal to zero. □

The Furukawa-Imai Scheme. The Furukawa-Imai group signature scheme [FI05] does not have opening soundness, which we will show in the following.

The scheme makes use of a group $\mathbb{G}$ (with generator g) in which the decisional Diffie-Hellman assumption holds, in addition to bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In the scheme, each group member i has a public key $Q_i = g^{x_i}$ and the corresponding secret key x_i. The public key Q_i is encrypted in a group signature with (a kind of) ElGamal encryption. Let $(R, V) = (Q_i g^r, S^r)$ be the ciphertext that appears in a group signature, where $S = g^s$ is the public key of the ElGamal encryption. The opener possesses the decryption key s, and identifies the signer by decrypting the ciphertext. An opening contains a proof of knowledge of w such that $Q_i = R/V^{1/w}$, where Q_i is the public key of the specified member (The opener uses s as the witness for the above equation).

If the adversary corrupts the opener and two different members i and j, the adversary can construct two different openings of a single signature, each of which attributes the signature to user i and user j, respectively. The adversary proceeds as follows: At first the signature is honestly generated by the user i. Let $(R, V) = (g^{x_i+r}, S^r)$ be the ciphertext contained in this signature. The first opening is also honestly generated by the opener to attribute the signature to i. The second proof is generated by computing a proof of knowledge w that satisfies $Q_j = R/V^{1/w}$ with the witness $w = sr/(x_i + r - x_j)$. This proof attributes the signature to the user j. Note that the randomness r for the encryption is reused to forge the second proof. This is the reason why the adversary needs to corrupt the user i, not only the user j and the opener.

The Bichsel et al. Scheme. In the Bichsel et al. scheme [BCN+10], a group member receives a Camenisch-Lysyanskaya signature on a random message ξ from the issuer. To generate a group signature, the member rerandomizes this certificate and computes a "signature of knowledge" of ξ on the message m in question. This rerandomized certificate on ξ and the signature of knowledge of ξ on m constitute the group signature.

The issuer should not know the random message ξ, because otherwise non-frameability is compromised. For this reason, in the group-joining protocol, ξ is jointly generated by the user and the issuer as follows: The user i chooses a random exponent τ_i and sends $\tilde{r} = \tilde{x}^{\tau_i}$ to the issuer, while the issuer also chooses a random κ_i and computes $\tilde{w} = \tilde{r} \cdot \tilde{x}^{\kappa_i} = \tilde{x}^{\tau_i+\kappa_i}$. This $\tau_i + \kappa_i$ will be used as the random message ξ mentioned above. To establish a publicly verifiable connection between this ξ and the user i, the user i generates an (ordinary) signature on $k_i = e(g, \tilde{r})$ with a key pair which is previously registered in a public key infrastructure.

To open a signature, the opener uses $\tilde{w}$ to identify the user which corresponds to the rerandomized certificate in the group signature, which is a Camenisch-Lysyanskaya signature on the user's ξ. However, since $\tilde{w}$ makes the Camenisch-Lysyanskaya signature publicly verifiable, it cannot be used as an opening. Instead, the opener produces a non-interactive zero-knowledge proof of $\tilde{w}$ and κ_i such that $k_i = e(g, \tilde{w})/e(g, \tilde{x})^{\kappa_i}$ and provides the signature on k_i. To verify this opening, a third party simply verifies the non-interactive zero-knowledge proof and the signature.

Unfortunately this scheme does not satisfy opening soundness. Assume a malicious signer obtains a group signature by an honest user, and further obtains an honestly generated opening of the signature. The proof of ownership contains k_i and a signature on this by the honest user. The malicious signer replaces the signature on k_i with his own signature on k_i. This forged opening passes the verification.

3.4 Achieving Opening Soundness

In this section we present a variant of the Groth scheme, which provides opening soundness (besides anonymity, non-frameability, and traceability).

3.4.1 The Modified Groth Scheme

The High-Level Idea. Let us first consider a general approach for achieving opening soundness.

The opener, who has the secret opening key, will always be able to determine the correct opening of a group signature. To provide opening soundness, the opener needs to convince others that a given opening is correct. The easiest way to do that is to make the opening key public, but this will compromise the anonymity of the scheme. Instead, the opener can provide an NIZK proof of the correctness of an opening, to convince any third party. This is, in fact, the approach used in the Bellare-Shi-Zhang construction.

If the opening algorithm essentially corresponds to a "decryption" of a ciphertext contained in the group signature (this is the case for many existing schemes), we might be able to take a different and more efficient approach. In particular, if the encryption scheme provides randomness recovering, the opener can simply release the randomness used for the ciphertext in question instead of an expensive zero-knowledge proof. Any third party will then be able to verify the correctness of an opening by re-encrypting the relevant information with the randomness provided by the opener, and then confirm that the resulting ciphertext is the same as the one contained in the signature.

In the Groth scheme, an opening essentially corresponds to the decryption of a linear encryption scheme. While linear encryption is not randomness-recovering, the opener is able to release related values which, together with the use of a bilinear map, will allow a third party to confirm that the decryption was done correctly. This property will allow us to add opening soundness to the original scheme. More specifically, in our variant of the Groth scheme, the opener, given a ciphertext $(c_1, c_2, c_3) = (F^r, H^s, vg^{r+s})$, reveals g^r and g^s as a part of an opening. Using the properties of the bilinear map, these values can replace the exact randomness r and s when checking the correspondence between a ciphertext and a decryption: If a third party, given g^r and g^s, wants to check the correspondence between a ciphertext (c_1, c_2, c_3) and a decryption v, he simply checks whether the equations $e(F, g^r) = e(c_1, g)$, $(H, g^s) = e(c_2, g)$, and $v = c_3/(g^r g^s)$ hold. If this is the case, he accepts the decryption as valid.

The above described modification to the Groth scheme will ensure that a verifier running the Judge algorithm is able to verify that the public user key v_i, given as part of an opening, is the same as the public user key used in the proof π which is contained in the group signature. This will ensure that two different openings containing different public user keys cannot both be accepted as valid for a single group signature. While this property is very close to opening soundness, it will not address the possibility that two different user have the same public key. To rule this out, we make the following additional change to the Groth scheme: we let both the verification algorithm GVf and the judge algorithm Judge take the registration table reg as input i.e. we assume that reg is made public (note that this is allowed in the original scheme [Gro10]). With this change, the Judge algorithm can simply check whether the public key, given as part of an opening, corresponds to the public key of more than one user, and reject the opening if this is the case. However, to ensure that the scheme remains traceable, the verification algorithm will have to implement a similar check. Hence, we will simply reject any signature or opening in the case the registration table reg contains repeated public keys. Note that to preserve correctness, this change also requires us to ensure that no honest execution of the Join protocol generates repeated public keys.

We note that the used approach to the verification of a decryption result is essentially the same as that used by Galindo et al. [GLF+10] in the context of public key encryption with non-interactive opening (PKENO). Furthermore, we note that in [GLF+10], the application of PKENO schemes to group signature is briefly discussed as a mechanism for simplifying the construction of an opening. Here, we will show that this approach is able to ensure the opening soundness of group signature schemes.

Description of our variant. The Groth scheme can achieve opening soundness with the small modification shown in Fig. 3.3.

```
Join/Issue(User $i$: $gpk$; Issuer: $gpk, ik$):
Run the coin-flipping protocol in [Gro10]
  The user obtains $v_i = g^{x_i}$ and $x_i$
  and the issuer obtains $v_i$
  (Repeat until $v_i \neq \mathsf{reg}[j]$ for all $j$)
Issuer: $r \leftarrow \mathbb{Z}_p$
      $(a_i, b_i) \leftarrow (f^{-r}, (v_i h)^r z)$
      set $\mathsf{reg}[i] \leftarrow v_i$
      send $(a_i, b_i)$ to the user
User: If $e(a_i, hv_i)e(f, b_i) = T$
      set $gsk_i \leftarrow (x_i, a_i, b_i)$
```

```
GVf($gpk$, reg, $m, \Sigma$):
Return 1 if the following holds:
  $1 = \mathrm{Ver}_{vk_{\mathrm{sots}}}((vk_{\mathrm{sots}}, m, a, \pi, y, \psi), \sigma_{\mathrm{sots}})$,
  $1 = V_{\mathrm{NIWI}}(crs, (gpk, a, \mathcal{H}(vk_{\mathrm{sots}})), \pi)$,
  $1 = V_{\mathrm{NIZK}}(crs, (gpk, \pi, y), \psi)$,
  $1 = \mathrm{ValidCiphertext}(pk, \mathcal{H}(vk_{\mathrm{sots}}), y)$,
  and $\mathsf{reg}[i] \neq \mathsf{reg}[j]$ for all $i \neq j$
else return 0
```

```
Open($gpk, ok$, reg, $m, \Sigma$):
If GVf($gpk$, reg, $m, \Sigma$) = 0, return $(0, \perp)$
$(b, v, \sigma) \leftarrow X_{xk}(crs, (gpk, a, \mathcal{H}(vk_{\mathrm{sots}})), \pi)$
$(d_F, d_H) \leftarrow xk$; $(y_1, \ldots, y_5) \leftarrow y$
$\tau_F := y_1^{1/d_F}$; $\tau_H := y_2^{1/d_H}$
Return $(i, (\sigma, \tau_F, \tau_H))$
  if there is $i$ so $v = \mathsf{reg}[i]$,
else $(0, \perp)$
```

```
Judge($gpk, i$, reg, $m, \Sigma, (\sigma, \tau_F, \tau_H)$):
$v_i \leftarrow \mathsf{reg}[i]$
Return 1 if the following holds:
  GVf($gpk$, reg, $m, \Sigma$) = 1,
  $i \neq 0$, $e(\sigma, v_i g^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$,
  $e(F, \tau_F) = e(y_1, g)$, $e(H, \tau_H) = e(y_2, g)$,
  and $\sigma \tau_F \tau_H = y_3$,
else return 0
```

Figure. 3.3 The proposed modification of the Groth group signature scheme. The algorithms that do not appear in the figure are exactly the same as in Fig. 3.1.

Theorem 3.5. *The modified Groth scheme shown in Fig. 3.3 provides opening soundness.*

Proof. Let us consider the game in Definition 3.2, and let gpk be the group public key in the game, where the key is parsed as $(F, H, \cdots)$, and let $(m, \Sigma, i, \tau, i', \tau')$ be the output of the adversary. Furthermore, let Σ, τ, and τ' be parsed as follows: $\Sigma = (vk_{\mathrm{sots}}, a, \pi, y, \psi, \sigma_{\mathrm{sots}})$ in which $y = (y_1, y_2, y_3, y_4, y_5)$, $\tau = (\sigma, \tau_F, \tau_H)$ and $\tau' = (\sigma', \tau'_F, \tau'_H)$.

We hereafter show that given a fixed Σ, it must hold that $i = i'$: Given a fixed Σ (in particular y_1, y_2, and y_3), the verification equations

$$e(F, \tau_F) \stackrel{?}{=} e(y_1, g) \wedge e(H, \tau_H) \stackrel{?}{=} e(y_2, g) \wedge \sigma \tau_F \tau_H \stackrel{?}{=} y_3$$

uniquely determine τ_F, τ_H, and σ. Since both $\tau = (\sigma, \tau_F, \tau_H)$ and $\tau' = (\sigma', \tau'_F, \tau'_H)$ are accepted by Judge and hence must satisfy the verification equations, we must have that $(\sigma, \tau_F, \tau_H) = (\sigma', \tau'_F, \tau'_H)$. Now, since $\sigma = \sigma'$ and the equation $e(\sigma, vg^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$ uniquely determines v given fixed σ and $\mathcal{H}(vk_{\mathrm{sots}})$, that v_i and $v_{i'}$ satisfy $e(\sigma, v_i g^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$ and $e(\sigma, v_{i'} g^{\mathcal{H}(vk_{\mathrm{sots}})}) = e(g, g)$ respectively, must imply that $v_i = v_{i'}$. Hence, since $v_i = \mathsf{reg}[i] \neq \mathsf{reg}[j] = v_j$ for all $i \neq j$, we conclude that $i = i'$. □

The changes shown in Fig. 3.3 yields a scheme which is secure in the Bellare-Shi-Zhang model i.e. the anonymity, the non-frameability, and the traceability of the original Groth scheme are maintained. This will be shown in the following.

Theorem 3.6. *The modified Groth scheme provides anonymity if the decisional linear assumption holds in $\mathbb{G}$, the one-time signature scheme is strongly unforgeable, and the hash function is target collision-resistant.*

Proof. Let $\mathcal{A}$ be an adversary that have the advantage ε in the anonymity game. To bound the probability ε we gradually modify the game played by $\mathcal{A}$. In the following S_i denotes the event that the adversary $\mathcal{A}$ successfully guesses the bit $b = b'$ interacting with the environment of Game i.

Game 0. Game 0 is identical to the game in the definition of anonymity. In this game we have that $\Pr[S_0] = 1/2 + \varepsilon$.

Game 1. We modify the behavior of the Open oracle as follows: If the Open oracle receives a valid signature which reuses the verification key vk^*_{sots} from the challenge Σ^*, then the game aborts. Due to the strong unforgeability of the one-time signature scheme $(\mathrm{KeyGen}_{\mathrm{sots}}, \mathrm{Sign}_{\mathrm{sots}}, \mathrm{Ver}_{\mathrm{sots}})$, this modification does not change the success probability of $\mathcal{A}$ with more than a negligible amount, that is, we have that $|\Pr[S_0] - \Pr[S_1]|$ is negligible.

Game 2. We further modify the Open oracle to abort when a queried signature contains a one-time signature verification key vk_{sots} that, when applying the hash function $\mathcal{H}$, collides with the challenge verification key vk^*_{sots} i.e. $\mathcal{H}(vk_{\mathrm{sots}}) = \mathcal{H}(vk^*_{\mathrm{sots}})$. This causes at most a negligible change in the probability in which $\mathcal{A}$ successfully guess the challenge bit due to the collision resistant property of $\mathcal{H}$.

Game 3. We then modify how the Open oracle obtains a signer identity i: When the Open oracle is required to open a group signature, it first extracts a witness (b, v, σ) from the proof π using the extraction key xk. However, in Game 3, instead of then searching for i such that $\mathsf{reg}[i] = v$ (which is done until Game 2), the Open oracle searches for i such that

$$e(\sigma, v_i g^{\mathcal{H}(vk_{\text{sots}})}) = e(g, g)$$

that is, σ is a valid signature on vk_{sots} under v_i. Note that the above verification equation uniquely defines v_i given a signature σ and a message $\mathcal{H}(vk_{\text{sots}})$. Furthermore, since the perfect soundness of π guarantees that σ is a valid signature on $\mathcal{H}(vk_{\text{sots}})$ under the extracted v, the v_i identified in the above procedure must be identical to v, and hence, the user identity i returned by the oracle does not vary between Game 2 and Game 3.

Game 4. We now modify how the Open oracle obtains the signature σ: Specifically, in Game 4, the Open oracle obtains σ by decrypting y with xk, instead of extracting σ from the proof of knowledge π. Due to the perfect soundness of ψ, this modification produces the same σ as in Game 3.

Game 5. Now we change how (σ, τ_F, τ_H) is computed. Instead of decrypting y with xk (recall that xk consists of $\log_g F$ and $\log_g H$), we proceed as follows: In the generation of the public key of the tag-based encryption, K and L are constructed as $K := F^\kappa$ and $L := H^\lambda$. The Open oracle then uses κ and λ to compute (σ, τ_F, τ_H) as $\tau_F := (y_4/y_1^\kappa)^{1/\mathcal{H}(vk_{\text{sots}})}$, $\tau_H := (y_5/y_2^\lambda)^{1/\mathcal{H}(vk_{\text{sots}})}$, and $\sigma := y_3/\tau_F\tau_H$. As shown in Lemma 3.7, this will not change the behavior of the oracle.

Game 6. In this game we switch the common reference string from a string providing perfect soundness to a string providing perfect witness-indistinguishability and perfect zero-knowledge, respectively, for the two types of proof systems used in the scheme. Since to two types of reference strings are computationally indistinguishable under the decisional linear assumption, the success probability of the adversary will not change by more than a negligible amount. Note that this change is possible because the Open oracle no longer needs the extraction key xk. Furthermore, in this game, all proofs ψ are simulated with the zero-knowledge trapdoor.

Game 7. Finally we change the component y_3 in the challenge to a random element in $\mathcal{G}$. As shown in Lemma 3.8, this will not introduce more than a negligible change in the success probability of the adversary assuming the decisional linear assumption holds.

In Game 7 we can conclude that $\Pr[S_7] = 1/2$, because the view of the adversary is independent from the challenge bit b. Specifically, the challenge $(vk^*_{\text{sots}}, a, \pi, y, \psi, \sigma^*_{\text{sots}})$ contains no information on b. Indeed, vk^*_{sots} is independently generated in the setup, a is distributed uniformly due to rerandomization, the perfectly witness-indistinguishable proof π distributes independently from the witness and hence the bit b, y is merely a random encryption, ψ does not contain the information on b since it is computed from y and the zero-knowledge trapdoor, and finally σ^*_{sots} is a signature on $\langle vk^*_{\text{sots}}, m, a, \pi, y, \psi\rangle$, which are all independent of b as seen above. The oracles (Open, SndToU, WReg, USK and CrptU) also behave independently of b.

Finally we prove that the changes in Game 5 and Game 7 will only introduce a negligibly change in the success probability of the adversary.

Lemma 3.7. $\Pr[S_4] = \Pr[S_5]$.

Proof (of Lemma 3.7). We will show that the response of the Open oracle does not change between Game 4 and Game 5.

Consider a group signature $\Sigma = (vk_{\text{sots}}, a, \pi, y, \psi, \sigma_{\text{sots}})$ submitted to the Open oracle. If the ciphertext y, which is a part of Σ, does not pass the validity check ValidCiphertext, the oracles in both games simply outputs $\perp$.

Hence, we consider the case in which the ciphertext y passes the validity check. In this case we can assume that there exist r and s in $\mathbb{Z}_p$ such that $y_1 = F^r$, $y_2 = H^s$, $y_4 = (g^{\mathcal{H}(vk_{\text{sots}})}K)^r$, and $y_5 = (g^{\mathcal{H}(vk_{\text{sots}})}L)^s$. We now show that the three equations $\tau_F = g^r$, $\tau_H = g^r$ and $\sigma = y_3/g^{r+s}$ hold in both games, and hence, the openings (τ_F, τ_H, σ) returned by Open in Game 4 and Game 5 are identical.

Consider the first two equations. In Game 4, τ_F and τ_H are computed as $\tau_F := y_1^{1/d_F}$ and

$\tau_H := y_2^{1/d_H}$. Since $F = g^{d_F}$ and $H = g^{d_H}$, $\tau_F = y_1^{1/d_F} = (F^r)^{1/d_F} = g^r$ and $\tau_H = y_2^{1/d_H} = (H^s)^{1/d_H} = g^s$ hold. In Game 5, τ_F and τ_H are computed as $\tau_F := (y_4/y_1^{\kappa})^{1/\mathcal{H}(vk_{\text{sots}})}$ and $\tau_H := (y_5/y_2^{\lambda})^{1/\mathcal{H}(vk_{\text{sots}})}$, where $K = F^{\kappa}$ and $L = H^{\lambda}$. Thus

$$\tau_F = \left(\frac{y_4}{y_1{}^{\kappa}}\right)^{1/\mathcal{H}(vk_{\text{sots}})} = \left(\frac{(g^{\mathcal{H}(vk_{\text{sots}})}K)^r}{(F^r)^{\kappa}}\right)^{1/\mathcal{H}(vk_{\text{sots}})} = g^r$$

and

$$\tau_H = \left(\frac{y_5}{y_2{}^{\lambda}}\right)^{1/\mathcal{H}(vk_{\text{sots}})} = \left(\frac{(g^{\mathcal{H}(vk_{\text{sots}})}L)^s}{(H^s)^{\lambda}}\right)^{1/\mathcal{H}(vk_{\text{sots}})} = g^s.$$

Lastly, consider the third equation $\sigma = y_3/g^{r+s}$. Note that σ is computed as $\sigma := y_3/y_1^{1/d_F}y_3^{d_H}$ in Game 4, whereas it is computed as $\sigma := y_3/(y_4/y_1^{\kappa})^{1/\mathcal{H}(vk_{\text{sots}})}(y_5/y_2^{\lambda})^{1/\mathcal{H}(vk_{\text{sots}})}$ in Game 5. Since we have already established that $y_1^{1/d_F} = (y_4/y_1^{\kappa})^{1/\mathcal{H}(vk_{\text{sots}})} = g^r$ and $y_2^{1/d_H} = (y_5/y_2^{\lambda})^{1/\mathcal{H}(vk_{\text{sots}})} = g^s$, we can conclude that the two computations yield the same value y_3/g^{r+s}. □

Lemma 3.8. $|\Pr[S_6] - \Pr[S_7]|$ *is negligible if the decisional linear assumption holds.*

Proof (of Lemma 3.8). To see this we construct a simulator that distinguishes a linear tuple from a random tuple, given that $|\Pr[S_6] - \Pr[S_7]|$ is non-negligible for some $\mathcal{A}$. The simulator receives the description of bilinear groups gk and a tuple (F, H, g, F^r, H^s, R) where R is g^{r+s} or a random group element, and simulates either Game 6 or Game 7, respectively.

Given gk and (F, H, g, F^r, H^s, R), the simulator constructs a witness-indistinguishable common reference string on the top of g, F, H together with a zero-knowledge trapdoor, which can be done because the trapdoor consists of only the discrete logarithms of U', V', W' with respect to F, H, and g. Then the simulator sets up K, L as $K := F^{c_1}g^{-\mathcal{H}(vk^*_{\text{sots}})}$, $L := H^{c_2}g^{-\mathcal{H}(vk^*_{\text{sots}})}$ where c_1, c_2 are randomly chosen from $\mathbb{Z}_p$. The rest of the public verification key gpk is honestly generated, and the adversary $\mathcal{A}$ is run with input gpk and ik.

When the adversary $\mathcal{A}$ issues an oracle query, the simulator responds as follows: User joining queries, both corrupted and uncorrupted, is dealt with by simply following the real protocol. The challenge request (i_0, i_1, m) is handled by picking a random bit b, computing a and π correctly from the signing key x_{i_b} of user i_b, computing a ciphertext y as $(y_1, y_2, y_3, y_4, y_5) :=$

$(F^r, H^s, R\sigma, (F^r)^{c_1}, (H^s)^{c_2})$, generating a simulated proof ψ from the zero-knowledge trapdoor, and generating a one-time signature σ_{sots} on $(vk^*_{\text{sots}}, m, a, \pi, y, \psi)$. When the simulator receives an open query $(vk_{\text{sots}}, a, \pi, y, \psi, \sigma_{\text{sots}})$, the simulator first verifies the signature, and if the signature does not pass the verification, it returns $\perp$. In the case the signature is valid, the simulator computes

$$\tau_F := (y_1^{c_1}/y_4)^{1/(\mathcal{H}(vk^*_{\text{sots}})-\mathcal{H}(vk_{\text{sots}}))}, \quad \tau_H := (y_2^{c_2}/y_5)^{1/(\mathcal{H}(vk^*_{\text{sots}})-\mathcal{H}(vk_{\text{sots}}))}, \quad \sigma := y_3/\tau_F\tau_H,$$

finds i for which σ is a valid signature on the message vk_{sots} under $v_i = \mathsf{reg}[i]$, and outputs $(i, (\sigma, \tau_F, \tau_H))$. If no such i is found, output $(0, \perp)$.

Finally the adversary outputs a bit b' and halts. The simulator outputs 1 if $b = b'$, and outputs 0 if $b \neq b'$.

In the above simulation, if R in the tuple given to the simulator is equal to g^{r+s}, the simulated oracle response is identical to that of Game 6. On the other hand, if R is randomly chosen, the simulation is identical Game 7. Hence if $|\Pr[S_6] - \Pr[S_7]|$ is non-negligible, the simulator's advantage in distinguishing linear tuples is also non-negligible. □

These two lemmas complete the proof of Theorem 3.6. □

Non-frameability and traceability can be proven more easily since these security notions do not require simulation of the Open oracle. For non-frameability, once an opening of the modified scheme that compromises the non-frameability notion is produced, one can obtain an opening for the original scheme (by simply dropping the extra components of τ_F and τ_H) which will compromise the non-frameability of the original scheme. The proof of the following theorems are essentially identical to the original proofs given in [Gro10], and are therefore not given here.

Theorem 3.9. *The modified Groth scheme provides non-frameability assuming the q-SDH assumption [BB08] holds, the one-time signature scheme is existentially unforgeable under a weak chosen message attack, and that the hash function is collision resistant.*

Theorem 3.10. *The modified Groth scheme provides traceability assuming the q-U assumption [Gro10] holds.*

3.5 Conclusion

We have identified an overlooked security concern for dynamic group signatures, namely, the possibility that a false opening proof can be produced by a corrupt user. To address this concern, we defined (two variants of) a new security notion denoted opening soundness, and furthermore discussed the opening soundness of several existing schemes. As a result, we have shown that the Bellare-Shi-Zhang construction [BSZ05] provides opening soundness as it is, and that small modifications to the Groth scheme (of the full version) [Gro10] allow this scheme to provide opening soundness as well. We have also briefly discussed the opening soundness of some of the random oracle schemes [FI05, BCN+10], but leave further investigation of these schemes as future work.

Chapter 4

On Necessity of Public-key Encryption with Non-interactive Opening for Group Signature Schemes

In this chapter, we proposed a generic construction of public-key encryption with non-interactive opening (PKENO) from any group signature scheme satisfying the opening soundness notion.

This contribution is the type (I) of our contribution (Sect. 1.1.3). Namely, by showing the generic construction, this contribution clarifies a necessity condition for constructing a group signature scheme satisfying opening soundness. This contribution evidences that for obtaining a group signature scheme with opening soundness it is necessary to use PKENO as a building block or to use a complexity theoretic assumption strong enough to construct a PKENO scheme.

4.1 Introduction

4.1.1 Background

Group signatures, introduced by Chaum and Heyst [CvH91], are a popular type of anonymous signatures. In a group signature (GS) scheme, a group manager (GM) issues a group signing key to each member in the group, and by using this key, a member can generate a *group signature* on behalf of the group. This signature is similar to an ordinary digital signature in that it is publicly verifiable using the public key of the group manager, but a verifier cannot identify the member within the group who constructed the signature. Hence, whereas it can be verified that a signature originates from a group member, the actual signer will remain anonymous. To prevent misuse, the GM is able to revoke this anonymity and identify the group member who constructed a given signature. Besides being interesting from a theoretical point of view, GS schemes provides functionalities which are applicable in many practical scenarios, which have led to a rigorous study of both the GS primitive and its applications in the literature. For example, in a biometric-based authentication scheme [BCPZ08], a user can be anonymously authenticated by using a user's biometric trait as a secret key of a GS scheme. In an identity management scheme for outsourcing business [IMS$^+$06], with the help of GS scheme, the outsourcee does not have to manage the list of identities of users. In an anonymous survey system [NS03], the dealer can collect statistical information without revealing the identity of users by applying a GS scheme.

However, due to its sophisticated functionalities, designing practical GS schemes is generally not easy, and hence, only a limited number of such constructions are known [ACJT00, BBS04, DP06, FI06, Gro07]. In principle, a GS scheme can be constructed from any enhanced trapdoor permutation [BMW03, BSZ05], but this fact does not immediately imply that it is possible to construct a *practical* GS scheme from such a cryptographic primitive. Similar gaps exist for ordinary digital signatures and pseudorandom generators, for which there are well-known generic constructions based on any one-way function [Gol01, Gol04, Rom90, BMG07]. However, these constructions are far from efficient, and stronger assumptions are required to construct *practical* schemes. Hence, investigating the difficulty of designing practical GS schemes goes beyond determining the theoretical minimal assumption on which group signatures can be built.

The motivation of the present work is to clarify the relative strength of the GS primitive by investigating the relationship of GS with another primitive of public-key encryption with non-interactive opening (PKENO). Since PKENO is recognized to be a very powerful cryptographic tool, a close relationship with these will highlight the difficulty of constructing group signature schemes.

4.1.2 Our Contribution

In this paper, we analyze the difficulty of constructing group signatures by showing an implication result. More specifically, we show that PKENO [DHKT08, Gal09] can be constructed from an arbitrary GS scheme which is secure in the dynamic group setting and provides opening soundness [SSE$^+$12a]. While opening soundness is not part of the security model defined by Bellare et al. [BSZ05], it has recently been introduced in [SSE$^+$12a] as an arguably essential security requirement for group signatures when considering security against a potentially malicious group manager(s), as done in [BSZ05][*1]. Our result implies that this type of GS is a very strong cryptographic primitive, since PKENO is already a stronger primitive than standard public key encryption (PKE), which itself is recognized as a very powerful cryptographic tool. Moreover, our transformation is relatively practical, as the resulting ciphertext consists of only a small number of group signatures, assuming the message space of the PKENO scheme is restricted to short messages. This shows that constructing an efficient

[*1] Intuitively, opening soundness guarantees that, for a given message/signature pair (m, σ), the group manager cannot convince a verifier that σ was constructed by one signer while, at the same time, being able to convince another verifier that σ was constructed by a *different* signer. We note that similar security requirements are considered for other types of signature schemes providing signer anonymity (e.g. partial signatures [BD09] and convertible undeniable signatures [PKO10]), and that the generic construction of a GS scheme presented in [BSZ05] provides opening soundness. Note, however, that not all GS scheme which are secure in the model of [BSZ05] provides this property e.g. [Gro07] does not. See Section 2.3.2 for a formal definition of opening soundness.

group signature (in terms of signature size) is as hard as constructing a PKENO scheme with small ciphertext overhead and a message space consisting of short messages. Furthermore, in our transformation, not all functionalities of a GS scheme are utilized to construct a PKENO scheme. This provides further evidence that GS is stronger than PKENO (We stress, however, that our result does not imply impossibility of designing practical GS schemes.)

From a technical point of view, we do not merely rewrite the functionality of a GS scheme to that of a PKENO scheme, but develop a dedicated multiple encryption technique for our conversion which simultaneously enhances efficiency and security. Specifically, in our approach, we obtain as an intermediate result a PKE scheme with single-bit plaintexts, and need to extend the plaintext space of this scheme to support sufficiently large message. To resolve this issue, we make use of our specific multiple encryption technique. We notice that the existing multiple encryption techniques cannot be applied to our approach since, for example, the Dodis-Katz multiple encryption technique [DK05] requires that the component encryption scheme already has a sufficiently large plaintext space, and the approach by Zhang-Hanaoka-Shikata-Imai [ZHSI04] requires random oracles which are known be to problematic [CGH98]. Furthermore, the Myers-Shelat technique [Ms09] cannot be applied to extend the plaintext space of our intermediate encryption scheme as the converted encryption scheme by [Ms09] loses special properties of the intermediate scheme which are important for constructing PKENO. However, our multiple encryption technique is based on the specific functionality of GS, and therefore, can only be applied in limited situations (like our conversions). More specifically, our multiple encryption technique exploits that a collection of group signatures can easily be bound to a tag by including the tag as part of the message being signed by each signature. This property plays a crucial role in achieving a secure encryption scheme with a larger message space while maintaining a reasonable level of efficiency. However, while the property follows straightforwardly from the functionality of GS, not many cryptographic primitives provide a similar property, which limits the applications of our technique.

4.1.3 Related Works

The relationship between GS and PKE has previously been studied in the literature. More specifically, Abdalla and Warinschi [AW04] gave a generic construction of chosen-plaintext (CPA) secure PKE (with multi-bit plaintext space) [AW04] from a GS scheme. This result was extended to generic constructions of chosen-ciphertext (CCA) secure PKE from GS with an appropriate level of security [CG05, OFHO09]. Our construction are obtained by extending these results and combining them with our multiple encryption technique to obtain the functionality of PKENO.

The concept of PKENO was first proposed by Damgård, Hofheinz, Kiltz, and Thorbek [DHKT08]. This type of scheme allows a receiver of a ciphertext to prove that the decryption result corresponds to a given message, without compromising his decryption key. The functionality of a PKENO can, for example, be used to construct a secure authenticated message transmission system with non-repudiation (introduced in [GLF$^+$10]). If a standard PKE is used, then there is no way to provide a non-repudiable proof of the origin of a received message unless the receiver reveals his own decryption key. By replacing PKE with PKENO, the receiver can provide such a proof.

Concrete PKENO constructions have also been proposed [DHKT08, Gal09, GLF$^+$10, LDLK10]. A generic construction of PKENO based on identity-based encryption (IBE) has been proposed [DHKT08] by following the IBE-to-PKE transformation by Canetti, Halevi, and Katz [CHK04]. Another generic construction proposed by Galindo et al. [GLF$^+$10] is based on (robust) threshold encryption scheme. However, unlike group signatures, the building blocks for these constructions (identity-based encryption and threshold encryption, respectively) are widely recognized as very powerful cryptographic primitives. Furthermore, these constructions will not allow us to draw any conclusion about the difficulty of constructing efficient group signature schemes.

4.2 Public Key Encryption with Non-interactive Opening from Group Signatures

In this section, we propose a generic construction of a PKENO scheme based on a GS scheme which is secure in the BSZ model and provides opening soundness.

4.2.1 Proposed GS-based PKENO

We will now present our GS-based PKENO construction. The basic idea behind our construction is similar to that of Abdalla and Warinschi [AW04] and Ohtake et al. [OFHO09]. More specifically, we let a ciphertext consist of a collection of group signatures. Each group signature will correspond to single bit of the plaintext and is constructed using one of two signing keys, which will be part of the public key, depending on whether the plaintext bit is 0 or 1. The receiver will then use the opening key to determine which signing key was used to construct the signature and will thereby learn the corresponding bit of the plaintext. Furthermore, due to the functionality of the group signature scheme, the receiver will also obtain a publicly verifiable proof of this correspondence, which will be used to implement the non-interactive opening property of the PKENO. To avoid malleability, the group signatures in a ciphertext will all be signatures on a verification key vk_{sots} of a one-time signature scheme, and the corresponding sk_{sots} will be used to construct a one-time signature which binds the group signatures together. However, this approach requires a somewhat counter-intuitive measure to ensure that the scheme provides the strong correctness requirement that a receiver can provide a publicly verifiable proof of the decryption result of *any* ciphertext. More specifically, if a ciphertext in the above construction outlined above contains a group signature which verifies but cannot be traced to one of the two signer keys in the public key, the ciphertext will be invalid, but the receiver will not obtain a publicly verifiable proof of this fact. In this case, we let the receiver reveal his private key as a proof of the invalidity of the ciphertext. Note, however, that if the underlying GS scheme satisfies traceability, such group signature will be computationally hard to construct, and this measure will not harm the security of the scheme.[*2] The message space of our PKENO construction is assumed to be $\mathcal{M}_{PKENO} = \{0,1\}^t$. The construction is defined as follows:

$\mathsf{NOKg}(1^\lambda)$: Given a security parameter 1^λ ($\lambda \in \mathbb{N}$), run $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\lambda)$. For $i = 1, 2$, run $(upk_i, usk_i) \leftarrow \mathsf{UKg}(1^\lambda)$, and the interactive algorithms $\mathsf{Join}(gpk, upk_i, usk_i)$ and $\mathsf{Issue}(gpk, ik)$ to obtain gsk_i and $\mathsf{reg}[i]$. Output an encryption key $pk = (gpk, gsk_1, gsk_2, upk_1, upk_2)$ and a decryption key $sk = (ok, \mathsf{reg})$.

$\mathsf{NOEnc}(pk, m)$: For a t-bit plaintext m, let $m_i \in \{0,1\}$ be the i-th bit of m. Generate a key pair $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow \mathsf{SgKg}_{\mathrm{sots}}(1^\lambda)$. For all $i \in [1, t]$, run $\sigma_i \leftarrow \mathsf{GSig}(gpk, gsk_{m_i+1}, vk_{\mathrm{sots}})$, compute $\sigma_{\mathrm{sots}} \leftarrow \mathsf{SgSign}_{sk_{\mathrm{sots}}}((\sigma_1, \ldots, \sigma_t))$, and output $C := (\sigma_{\mathrm{sots}}, vk_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t))$. Note that σ_i is a GS of the signed message vk_{sots} under the signing key gsk_{m_i+1}.

$\mathsf{NODec}(pk, sk, C)$: Parse $C = (\sigma_{\mathrm{sots}}, vk_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t))$. If $\mathsf{SgVerify}_{vk_{\mathrm{sots}}}(\sigma_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t)) \neq 1$, then output $\perp$. Otherwise, for all $i \in [1, t]$, if there exists σ_i such that $0 \leftarrow \mathsf{GVf}(gpk, vk_{\mathrm{sots}}, \sigma_i)$, then output $\perp$. Otherwise, run $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\mathrm{sots}}, \sigma_i, \mathsf{reg})$. If $j_i \notin \{1, 2\}$ or $\mathsf{Judge}(gpk, m_i + 1, upk_{m_i+1}, vk_{\mathrm{sots}}, \sigma_i, \tau_i) = 0$ for any $i \in [1, t]$, then output $\perp$. Otherwise, set $m_i = j_i - 1$, and output $m = m_1 \| \cdots \| m_t$.

$\mathsf{NOProve}(pk, sk, C)$: Parse C as $(\sigma_{\mathrm{sots}}, vk_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t))$. If $\mathsf{SgVerify}_{vk_{\mathrm{sots}}}(\sigma_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t)) \neq 1$ or if $\mathsf{GVf}(gpk, vk_{\mathrm{sots}}, \sigma_i) \neq 1$ for any $i \in [1, t]$, then output the proof $\pi = \emptyset$ indicating an invalid ciphertext. Otherwise, run $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\mathrm{sots}}, \sigma_i, \mathsf{reg})$ for all $i \in [1, t]$. If $j_i \notin \{1, 2\}$ or $\mathsf{Judge}(gpk, m_i + 1, upk_{m_i+1}, vk_{\mathrm{sots}}, \sigma_i, \tau_i) = 0$ for any i, the ciphertext is invalid, but this cannot be publicly verified. In this case, output $sk = (ok, \mathsf{reg})$ as a proof. Otherwise, output the proof $\pi = (\tau_1, \ldots, \tau_t)$.

[*2] One might think that it is sufficient to output a special symbol, e.g. $\perp$, to indicate that (some of) the signatures are untraceable. However, this is not the case. In a construction where this approach is taken, a malicious receiver will be able to claim that a ciphertext, which is actually valid, is invalid by outputting $\perp$. Since a verifier cannot distinguish between traceable and untraceable signatures, he will not be able to detect that the claim made by the receiver is incorrect, and if $\perp$ is accepted as a valid proof, the verifier would be convinced that the ciphertext in question is invalid when this might not the case.

$\mathsf{NOVerify}(pk, C, m, \pi)$: Parse C as $(\sigma_{\text{sots}}, vk_{\text{sots}}, (\sigma_1, \ldots, \sigma_t))$ and consider the following cases:

$m = \perp, \pi = \emptyset$: If $\mathsf{SgVerify}_{vk_{\text{sots}}}(\sigma_{\text{sots}}, (\sigma_1, \ldots, \sigma_t)) \neq 1$ or if $\mathsf{GVf}(gpk, vk_{\text{sots}}, \sigma_i) \neq 1$ for any $i \in [1, t]$, then output 1. Otherwise, output 0.

$m = \perp, \pi = sk$: Compute $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_i, \mathsf{reg})$ for all $i \in [1, t]$. If $j_i \notin \{1, 2\}$ or $\mathsf{Judge}(gpk, j_i, upk_{j_i}, vk_{\text{sots}}, \sigma_i, \tau_i) = 0$ for any i, output 1. Otherwise, output 0.

$m = \perp, \pi = (\tau_1, \ldots, \tau_t)$ **or** $m \neq \perp, \pi \neq (\tau_1, \ldots, \tau_t)$: Output 0.

$m \neq \perp, \pi = (\tau_1, \ldots, \tau_t)$: Parse $m \rightarrow m_1 \| \cdots \| m_t$. If $\mathsf{Judge}(gpk, m_i + 1, upk_{m_i+1}, vk_{\text{sots}}, \sigma_i, \tau_i) = 1$ for all $i \in [1, t]$, output 1. Otherwise, output 0.

Note that in the above construction, the dynamic aspects of the group signature scheme is actually not required since a user will run GKg, UKg, and the (interactive) Join, Issue protocol by himself as part of the NOKg algorithm. However, the functionality provided by Open and Judge, which is normally not defined for a static group signature scheme [BMW03], is crucial for the construction.

4.2.2 Security Analysis

In the following, we prove that our GS-based PKENO construction satisfies the functionality and security requirements outlined in above. The correctness of our PKENO construction easily follows from the correctness of the underlying group signature scheme, so we leave out the details of this observation.

Theorem 4.1. *Our GS-based PKENO scheme satisfies correctness.*

Proof. This can be seen by considering the output produced by NODec and NOProve when given a ciphertext with different properties, and how NOVerify will respond to this.

Firstly, consider the case in which the ciphertext $C = (\sigma_{\text{sots}}, vk_{\text{sots}}, (\sigma_1, \ldots, \sigma_t))$ has the property that

$$\mathsf{SgVerify}_{vk_{\text{sots}}}(\sigma_{\text{sots}}, (\sigma_1, \ldots, \sigma_t)) \neq 1$$

or

$$\mathsf{GVf}(gpk, vk_{\text{sots}}, \sigma_i) \neq 1$$

for at least one $i \in [1, t]$. In this case, C corresponds to an invalid ciphertext (which can be publicly verified), NODec will output $m = \perp$, NOProve will output $\pi = \emptyset$, and hence $\mathsf{NOVerify}(pk, C, m, \pi)$ will output 1. In the following cases, we assume $\mathsf{SgVerify}_{vk_{\text{sots}}}(\sigma_{\text{sots}}, (\sigma_1, \ldots, \sigma_t)) = 1$ and $\mathsf{GVf}(gpk, vk_{\text{sots}}, \sigma_i) = 1$ for all $i \in [1, t]$.

Secondly, consider the case in which C has the property that there exists an index $i^* \in [1, t]$ for which $(j_{i^*}, \tau_{i^*}) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_{i^*}, \mathsf{reg})$ and either

$$j_{i^*} \notin \{1, 2\}$$

or

$$\mathsf{Judge}(gpk, j_{i^*}, upk_{j_{i^*}}, vk_{\text{sots}}, \sigma_{i^*}, \tau_{i^*}) = 0$$

holds. This implies that C is an invalid ciphertext (although this cannot be publicly verified) and NODec will output $\perp$. Furthermore, NOProve will output (sk, reg) as a proof, and NOVerify will run Open and Judge to confirm the invalidity of C, and lastly return 1 since $m = \perp$.

Lastly, consider the case in which $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_i, \mathsf{reg})$ and both

$$j_i \in \{1, 2\}$$

and

$$\mathsf{Judge}(gpk, m_{i^*} + 1, upk_{m_{i^*}+1}, vk_{\text{sots}}, \sigma_{i^*}, \tau_{i^*}) = 1$$

hold for all $i \in [1, t]$. Note that in this case, C is a valid ciphertext, NODec will always output a message $m \neq \perp$ and NOProve will always output a proof of the form $\pi = (\tau_1, \ldots, \tau_t)$. Furthermore, since Open and Judge are deterministic and are executed with the same input, if we run NODec, NOProve, and NOVerify, the output of NOVerify will be 1.

Since a ciphertext C must fall in one of the above described cases, we conclude that our proposed PKENO scheme must be correct. □

Theorem 4.2. *Our GS-Based PKENO scheme is chosen-ciphertext secure if the underlying GS scheme satisfies anonymity and the one-time signature scheme is OT-sUF-CMA secure.*

Proof. The roadmap of our proof is as follows. For a fixed pair of challenge messages, m_0 and m_1, we define the sequences of games [Sho04] $G_0, G_1, \ldots, G_{t'}$, where $t' := |\{i \mid m_{0,i} \neq m_{1,i}\}|$ is the number of different bits between m_0 and m_1. Let $(i_1, i_2, \ldots, i_{t'})$ be the set of the positions such that $m_{0,i_\ell} \neq m_{1,i_\ell}$ ($\ell \in [1, t']$). The first game, G_0, is defined to be identical to $\mathrm{Exp}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(1^\lambda)$ with an exception that the challenge ciphertext is computed using m_0 as a plaintext. In G_ℓ, the i_ℓ-th bit of m_0 is changed to m_{1,i_ℓ}, and the challenge ciphertext is computed using this plaintext. In other words, $G_{\ell-1}$ and G_ℓ are identical except the i_ℓ-th bit of the challenge message is different. The last game, $G_{t'}$, corresponds to $\mathrm{Exp}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(1^\lambda)$ in which the challenge ciphertext is computed using m_1 as the plaintext. We claim that for every $1 \leq \ell \leq t'$, if there exists an adversary $\mathcal{A}$ that can distinguish between playing game $G_{\ell-1}$ and game G_ℓ with non-negligible probability, then we can construct an algorithm that can break anonymity of the underlying GS scheme. If this is the case, we can conclude that the output of any chosen-ciphertext adversary $\mathcal{A}$ will only be different with negligible probability if $\mathcal{A}$ is given an encryption of m_1 instead of an encryption of m_0 as a challenge ciphertext. This implies that the GS scheme is chosen-ciphertext secure.

Let $\mathcal{A}$ be an chosen-ciphertext adversary playing either game $G_{\ell-1}$ or game G_ℓ, and let $\epsilon_\mathcal{A}$ denote the difference between the probability that $\mathcal{A}$ outputs 1 in game $G_{\ell-1}$ and in game G_ℓ. We assume that $\epsilon_\mathcal{A}$ is non-negligible. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that breaks the anonymity of the underlying GS scheme.

$\mathcal{B}$ interacts in the anonymity experiment for the GS scheme, and initially receives the public group key gpk and the issuer key ik. First $\mathcal{B}$ queries the identities 1 and 2 to the SndToU oracle, interacts with the oracle running $\mathsf{Issue}(gpk, ik)$, and also queries $\mathsf{USK}(1)$ and $\mathsf{USK}(2)$ to obtain (upk_1, usk_1, gsk_1) and (upk_2, usk_2, gsk_2). In addition, $\mathcal{B}$ generates $(vk^*_{\mathrm{sots}}, sk^*_{\mathrm{sots}}) \leftarrow \mathsf{SgKg}_{\mathrm{sots}}(1^\lambda)$. $\mathcal{B}$ then runs $\mathcal{A}$ with input $pk = (gpk, gsk_1, gsk_2, upk_1, upk_2)$.

When a decryption query $C = (\sigma_{\mathrm{sots}}, vk_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t))$ is submitted by $\mathcal{A}$, $\mathcal{B}$ answers as follows: $\mathcal{B}$ checks whether all (one-time and group) signatures are valid or not. If there is an invalid signature, then $\mathcal{B}$ returns $\perp$. Otherwise, we consider the following two cases:

$\boldsymbol{vk_{\text{sots}} = vk^*_{\text{sots}}}$**:** We call this case the forge event. $\mathcal{B}$ outputs a random bit, and aborts.

$\boldsymbol{vk_{\text{sots}} \neq vk^*_{\text{sots}}}$**:** For all $i \in [1, t]$, $\mathcal{B}$ sends $(vk_{\text{sots}}, \sigma_i)$ to the opening oracle, and obtains (j_i, τ_i). If

$$j_i \notin \{1, 2\}$$

or

$$\mathsf{Judge}(gpk, j_i, upk_{j_i}, vk_{\text{sots}}, \sigma_i, \tau_i) = 0,$$

then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ sets $m_i = j_i - 1$, and returns $m := m_1 \| \cdots \| m_t$.

When a proof query $C = (\sigma_{\text{sots}}, vk_{\text{sots}}, (\sigma_1, \ldots, \sigma_t))$ is submitted by $\mathcal{A}$, $\mathcal{B}$ responds as in a decryption query, except that when the indices and proofs (j_i, τ_i) are obtained, $\mathcal{B}$ checks that $j_i \in \{1, 2\}$ and $\mathsf{Judge}(gpk, j_i, upk_{j_i}, vk_{\text{sots}}, \sigma_i, \tau_i) = 1$ for all $i \in [1, t]$. If this is not the case, $\mathcal{B}$ aborts, and we denote this event non-trace. Otherwise, $\mathcal{B}$ returns $\pi = (\tau_1, \ldots, \tau_t)$.

In the challenge phase, $\mathcal{A}$ submits two challenge messages (m_0, m_1). $\mathcal{B}$ responds as follows:

Recall that, in game G_ℓ, the bit at index i_ℓ in the plaintext used to construct the challenge ciphertext is changed from m_{0,i_ℓ} to m_{1,i_ℓ}. For all $k \in [1, i_\ell - 1]$, $\mathcal{B}$ computes $\sigma^*_k \leftarrow \mathsf{GSig}(gpk, gsk_{m_{1,k}+1}, vk^*_{\text{sots}})$. For $k = i_\ell$, $\mathcal{B}$ submits the identities m_{0,i_ℓ} and m_{1,i_ℓ}, and the message vk^*_{sots} as his challenge values, and obtains $\sigma^*_k \leftarrow \mathsf{GSig}(gpk, gsk_{m_{b,i_\ell}+1}, vk^*_{\text{sots}})$ (note that $m_{0,i_\ell} \neq m_{1,i_\ell}$ and that the bit b is unknown to $\mathcal{B}$). For all $k \in [i_\ell + 1, t]$, $\mathcal{B}$ computes $\sigma^*_k \leftarrow \mathsf{GSig}(gpk, gsk_{m_{0,k}+1}, vk^*_{\text{sots}})$. Lastly, $\mathcal{B}$ computes $\sigma^*_{\text{sots}} \leftarrow \mathsf{SgSign}_{sk^*_{\text{sots}}}((\sigma^*_1, \ldots, \sigma^*_t))$, and sends $C^* := (\sigma^*_{\text{sots}}, vk^*_{\text{sots}}, (\sigma^*_1, \ldots, \sigma^*_t))$ as the challenge ciphertext to $\mathcal{A}$.

After the challenge phase, $\mathcal{A}$ can ask additional decryption and proof queries which $\mathcal{B}$ responds to as above (note that since $\mathcal{B}$ aborts when $vk_{\text{sots}} = vk^*_{\text{sots}}$, $\mathcal{B}$ will never submit the illegal query $(vk^*_{\text{sots}}, \sigma^*_{i_\ell})$ to the opening oracle).

At some point, $\mathcal{A}$ will output a bit b' which $\mathcal{B}$ forwards as his own guess in the anonymity experiment of the underlying GS scheme. Note that if $\mathcal{B}$'s challenge bit b is 0, then $\mathcal{B}$ will be simulating game $G_{\ell-1}$ to $\mathcal{A}$, whereas if $b = 1$, $\mathcal{B}$ will be simulating game G_ℓ to $\mathcal{A}$. Since the simulation is perfect assuming $\mathcal{B}$ does not abort, $\mathcal{B}$ breaks the anonymity of the GS scheme with probability at least

$$\mathrm{Adv}^{\mathrm{anon}}_{\mathsf{GS},\mathcal{B}} \geq |\epsilon_{\mathcal{A}} - \Pr[\mathsf{forge}] - \Pr[\mathsf{non\text{-}trace}]|$$

Recall that $\epsilon_{\mathcal{A}}$ is assumed to be non-negligible. To complete the proof, we show that $\Pr[\mathsf{forge}]$ and $\Pr[\mathsf{non\text{-}trace}]$ are negligible assuming the used one-time signature scheme is OT-sUF-CMA secure and the GS scheme satisfies traceability. This implies that $\mathcal{B}$ will break the anonymity of the GS scheme with non-negligible probability.

We first show that $\Pr[\mathsf{forge}]$ is negligible by constructing an algorithm $\mathcal{B}'$ which interacts with $\mathcal{A}$ and breaks the OT-sUF-CMA security of the one-time signature scheme if forge occurs. $\mathcal{B}'$ is constructed as follows.

Initially, $\mathcal{B}'$ is given a verification key vk from the OT-sUF-CMA experiment, which $\mathcal{B}'$ will use as vk^*_{sots} in the interaction with $\mathcal{A}$. Firstly, $\mathcal{B}'$ generates $(pk, sk) \leftarrow \mathsf{NOKg}(1^\lambda)$ and runs $\mathcal{A}$ with input pk. Note that since $\mathcal{A}$ knows sk, all decryption and proof queries can trivially be answered. If forge occurs before the challenge phase, $\mathcal{A}$ will submit a decryption or proof query $C = (\sigma_{\mathrm{sots}}, vk_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t))$ for which $vk_{\mathrm{sots}} = vk^*_{\mathrm{sots}}$ and σ_{sots} is a valid signature on $(\sigma_1, \ldots, \sigma_t)$ under vk^*_{sots}. Hence, $\mathcal{B}'$ outputs σ_{sots} and $m = (\sigma_1, \ldots, \sigma_t)$, and breaks the OT-sUF-CMA security of the one-time signature scheme.

Otherwise, $\mathcal{A}$ will submit two challenge messages, m_0 and m_1. $\mathcal{B}'$ picks $b \leftarrow \{0, 1\}$, computes $\sigma^*_i \leftarrow \mathsf{GSig}(gpk, gsk_{m_{b,i}+1}, vk^*_{\mathrm{sots}})$ for $i \in [1, t]$, and submits $(\sigma^*_1, \ldots, \sigma^*_t)$ to his one-time signing oracle to obtain σ^*_{sots}. Lastly, $\mathcal{B}'$ forwards $C^* = (\sigma^*_{\mathrm{sots}}, vk^*_{\mathrm{sots}}, (\sigma^*_1, \ldots, \sigma^*_t))$ to $\mathcal{A}$. If, after the challenge phase, forge occurs, $\mathcal{B}'$ breaks the OT-sUF-CMA security as above. Note that the forgery output by $\mathcal{B}'$ will be a valid forgery since $C \neq C^*$ implies that $(\sigma_{\mathrm{sots}}, (\sigma_1, \ldots, \sigma_t)) \neq (\sigma^*_{\mathrm{sots}}, (\sigma^*_1, \ldots, \sigma^*_t))$.

Hence, $\mathcal{B}'$ will break the OT-sUF-CMA security of the one-time signature scheme whenever forge occurs.

Lastly, we show that $\Pr[\mathsf{non\text{-}trace}]$ is negligible by constructing an algorithm $\mathcal{B}''$ which interacts with $\mathcal{A}$ and breaks the traceability of the GS scheme whenever $\mathsf{non\text{-}trace}$ occurs. $\mathcal{B}''$ is constructed as follows.

Initially, $\mathcal{B}''$ is given (gpk, ok) as input and have access to the oracles $\mathcal{O} = \{\mathsf{SndToU}(\cdot,\cdot),$ $\mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot)\}$. $\mathcal{B}$ makes queries $\mathsf{AddU}(1)$, $\mathsf{AddU}(2)$, $\mathsf{RReg}(1)$,

$\mathsf{RReg}(2)$, $\mathsf{USK}(1)$, and $\mathsf{USK}(2)$ to its oracles to obtain (upk_1, usk_1, gsk_1), (upk_2, usk_2, gsk_2) and reg. Then $\mathcal{B}$ sets $pk \leftarrow (gpk, gsk_1, gsk_2, upk_1, upk_2)$, $sk \leftarrow (ok, \mathsf{reg})$, and runs $\mathcal{A}$ with input pk. Note that $\mathcal{B}''$ can trivially answer all decryption and proof queries since $\mathcal{B}''$ knows sk. If $\mathsf{non\text{-}trace}$ occurs, $\mathcal{A}$ submits a decryption or proof query $C = (\sigma_{\text{sots}}, vk_{\text{sots}}, (\sigma_1, \ldots, \sigma_t))$ such that if we compute $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_i, \mathsf{reg})$, then exists an i such that either $j_i \notin \{1, 2\}$ or $\mathsf{Judge}(gpk, m_i + 1, upk_{m_i+1}, vk_{\text{sots}}, \sigma_i, \tau_i) = 0$. This is exactly the winning condition in the traceability experiment, and $\mathcal{B}''$ returns $m = vk^*_{\text{sots}}$ and σ_i to break the traceability of the GS scheme.

This completes the proof. □

Theorem 4.3. *Our GS-Based PKENO scheme satisfies proof soundness if the underlying GS scheme satisfies opening soundness and correctness, and the one-time signature scheme satisfies correctness.*

Proof. Let $\mathcal{A}$ be an adversary who breaks proof soundness of our GS-based PKENO scheme. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that breaks opening soundness of the underlying GS scheme. $\mathcal{B}$ is constructed as follows.

Initially, $\mathcal{B}$ receives gpk, ok, and ik from the opening soundness experiment of the underlying GS scheme. $\mathcal{B}$ then makes (upk_1, usk_1, gsk_1), (upk_2, usk_2, gsk_2), and reg using ik, and makes four queries $\mathsf{CrptU}(1, upk_1)$, $\mathsf{CrptU}(2, upk_2)$, $\mathsf{WReg}(1, \mathsf{reg}[1])$, and $\mathsf{WReg}(2, \mathsf{reg}[2])$. Lastly, $\mathcal{B}$ sets $pk = (gpk, gsk_1, gsk_2, upk_1, upk_2)$ and $sk = (ok, \mathsf{reg})$, and runs $\mathcal{A}$ with input (pk, sk).

At some point, $\mathcal{A}$ sends a challenge message m to $\mathcal{B}$ which parses m as $m := m_1 \| \cdots \| m_t$. $\mathcal{B}$ then generates $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow \mathsf{SgKg}_{\text{sots}}(1^\lambda)$, and computes $\sigma_i \leftarrow \mathsf{GSig}(gpk, gsk_{m_i+1}, vk_{\text{sots}})$ for all $i \in [1, t]$. Lastly, $\mathcal{B}$ computes $\sigma_{\text{sots}} \leftarrow \mathsf{SgSign}_{sk_{\text{sots}}}((\sigma_1, \ldots, \sigma_t))$, and sends the ciphertext $C := (\sigma_{\text{sots}}, vk_{\text{sots}}, (\sigma_1, \ldots, \sigma_t))$ to $\mathcal{A}$.

After receiving C, $\mathcal{A}$ will output a message m' and a proof π'. The definition of proof soundness requires that the output of a successful adversary satisfies $\mathsf{NOVerify}(pk, C, m', \pi') = 1$ and $m \neq m'$. We first consider the case in which $(m', \pi') = (\perp, \emptyset)$. Note that due to the correctness of the one-time signature scheme and the group signature scheme, it must be the case that $\mathsf{SgVerify}_{sk_{\text{sots}}}(\sigma_{\text{sots}}, (\sigma_1, \ldots, \sigma_t)) = 1$ and $\mathsf{GVf}(gpk, vk_{\text{sots}}, \sigma_i) = 1$ for all $i \in [1, t]$. Hence, due to the definition of $\mathsf{NOVerify}$, $\mathcal{A}$ will have advantage 0 in this case. Likewise, in the case $(m, \pi') = (\perp, sk)$[*3], where $sk = (ok, \mathsf{reg})$, we note that the correctness of the group signature scheme implies that computing $(j_i, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_i, \mathsf{reg})$ yields $j_i \in \{1, 2\}$ and $\mathsf{Judge}(gpk, j_i, upk_{j_i}, vk_{\text{sots}}, \sigma_i, \tau_i) = 1$ for all $i \in [1, t]$, and hence, $\mathcal{A}$ will have advantage 0 due to the definition of $\mathsf{NOVerify}$.

This leaves the case $m' \neq \perp$ and $\pi' \neq \{\emptyset, sk\}$. In this case, $\mathcal{B}$ parses m' and π' as $m' := m'_1 \| \cdots \| m'_t$ and $\pi' := (\tau'_1, \ldots, \tau'_t)$. Recall that the definition of proof soundness requires that the output of a successful adversary satisfies $m \neq m'$. Hence, there must exist at least one index $i \in [1, t]$ such that $m'_i \neq m_i$. For such an i, $\mathcal{B}$ computes $(j, \tau_i) \leftarrow \mathsf{Open}(gpk, ok, vk_{\text{sots}}, \sigma_i)$. Due to the correctness of the GS scheme, it must hold that $\mathsf{Judge}(gpk, m_i + 1, upk_{m_i+1}, vk_{\text{sots}}, \sigma_i, \tau_i) = 1$. Furthermore, since $\mathsf{NOVerify}(pk, C, m', \pi') = 1$, it must also hold that $\mathsf{Judge}(gpk, m'_i + 1, upk_{m'_i+1}, vk_{\text{sots}}, \sigma_i, \tau'_i) = 1$. Hence, by returning the message vk_{sots}, the signature σ_i, the identities $m_i + 1$ and $m'_i + 1$, and the proofs τ_i and τ'_i, $\mathcal{B}$ breaks the opening soundness of the underlying GS scheme.

□

4.2.3 Concrete Implementations

A generic construction of a GS scheme secure in the BSZ model was introduced in [BSZ05] based on an existential unforgeable digital signature, a CCA-secure PKE, and a simulation-sound non-interactive zero-knowledge proof system. In addition, there are several GS schemes secure in the BSZ model such as e.g., the Delerablée-Pointcheval scheme [DP06] and the Groth schemes [Gro06, Gro07]. The Delerablée-Pointcheval scheme is efficient but only secure in the random oracle model. Although the first scheme by Groth [Gro06] is secure in the standard model, each group signature consists of a large number of group elements. The second scheme by Groth [Gro07] provides a reasonable constant-size group

[*3] Notice that in this case, outputting the secret key sk as a proof will play a crucial role in proving the soundness of the proposed scheme. See the first paragraph of Section 4.2.1 for details.

signature, and is secure in the standard model. As for the latter Groth scheme [Gro07], although the original scheme does not provide opening soundness as it is, it will provide opening soundness with a slight modification [GLF$^+$10, SSE$^+$12a][*4].

4.3 Conclusion

In this paper, we have shown that PKENO can be constructed from group signatures which are secure in the BSZ model and provide opening soundness. These results imply that this type of group signatures are a stronger primitive than PKENO which itself is already a very strong primitive compared to ordinary CCA-secure public key encryption. Furthermore, assuming the used group signature scheme is efficient, the PKENO scheme derived from our constructions are also relatively efficient. Hence, we can interpret our result as evidence of the thesis that designing group signatures is significantly harder than designing many other ordinary cryptographic primitives.

[*4] Actually the Groth GS scheme (and its variant by Sakai et al. [SSE$^+$12a]) adopt a different syntax from the BSZ model, thus the security definitions under which the security of the schemes are proven also need to be modified from the BSZ model (See [SSE$^+$12b] for further discussion). Fortunately these security definitions are sufficiently strong to instantiate our generic constructions with the Groth scheme.

Chapter 5

Threshold Encryption with Decryption Consistency from Public-key Encryption with Non-interactive Opening

In this chapter, we propose a generic construction of threshold public-key encryption with decryption consistency from any PKENO scheme.

This contribution is understood as a contribution of type (I) in Sect. 1.1.3. Combining the result by Galindo et al. [GLF$^+$10], the contribution shows equivalence of between existence of a PKENO scheme and a threshold encryption scheme. It shows that for designing a threshold encryption scheme it is promising to design a PKENO scheme and then extend it to support threshold decryption.

5.1 Introduction

5.1.1 Dynamic threshold encryption.

Threshold public-key encryption (TPKE) [CG99, DF90, DSDFY94, SG02] is an extension of ordinary public-key encryption which distributes the secret key among several (say, n) decryption servers such that arbitrary k servers are needed to cooperate to successfully decrypt a ciphertext. In this paper, we consider TPKE schemes to be non-interactive, which means that each decryption server is able to produce its "decryption share" without interacting with other parties, and any (honestly generated) k decryption shares from k different servers can be combined successfully to produce the correct plaintext.

In addition to (a threshold variant of) the chosen-ciphertext security, TPKE schemes are required to satisfy decryption consistency [BBH06, SG02]. The decryption consistency requires that even if a sender and the decryption servers collude, they cannot create two different sets of k decryption shares which respectively produce different plaintexts when honestly combined. This property forces a sender to commit to the message being encrypted. More specifically, decryption consistency prevents a malicious sender from creating "equivocal" ciphertexts essentially corresponding to the encryption of two different messages, and then, at a later stage, deciding what message the ciphertext should decrypt to by forcing a specific set of servers to participate in the decryption process.

Many TPKE schemes have a limitation that restricts the set of authorized decryption servers (i.e. the servers allowed to participate in the decryption process) and the threshold to be fixed at the setup of the scheme. In addition, decryption servers cannot join the system after the system is set up. This restricts the flexibility of TPKE schemes, and potentially limits the applications of TPKE.

To address these restrictions, Delerablée and Pointcheval proposed dynamic TPKE [DP08]. A dynamic TPKE scheme allows a decryption server to join the system after the setup, and also allows a sender to choose the threshold k and the authorized set of servers, among which any k servers can successfully decrypt the ciphertext when they cooperate. Delerablée and

Pointcheval [DP08] formalized syntax and a security notion of dynamic TPKE, and proposed a concrete dynamic TPKE scheme from an assumption called the multi-sequence of exponent Diffie-Hellman (MSE-DDH) assumption.

The Delerablée-Pointcheval scheme is currently the only known dynamic TPKE scheme, and thus, until now, there have been no dynamic TPKE schemes that avoid q-type assumptions[*1]. Furthermore, their scheme depends on a random oracle to achieve decryption consistency, and hence there is no known dynamic TPKE schemes in the standard model that provides decryption consistency regardless of the underlying assumption.

5.1.1.1 Our contribution.

To overcome these drawbacks, we propose new dynamic TPKE schemes supporting decryption consistency without depending on any q-type assumptions or random oracles. More precisely we propose two constructions of dynamic TPKE, both of which use public-key encryption with non-interactive opening (PKENO) as a core component of the constructions. PKENO [DHKT08] is an extension of the ordinary public-key encryption that allows the receiver to prove the validity of the decryption result without revealing the decryption key.

The first scheme uses a PKENO scheme in a purely black-box manner, it is a generic, or more precisely black-box, construction of a dynamic TPKE scheme from PKENO. The construction combines the multiple encryption technique [DK05] and a technique of verifiable secret sharing [BKP11, BOGW88] to ensure the decryption consistency. However, this generic construction archives relatively weaker notion of decryption consistency, compared with several previous (non-dynamic) TPKE schemes (More concretely, the definition of decryption consistency that the first scheme satisfies is slightly weaker than the definition that, for example, the Boneh-Boyen-Halevi scheme [BBH06] satisfies).

The second proposed scheme overcomes this weakness of the first proposed scheme, by deviating from being a generic construction. This scheme combines a specific PKENO

[*1] A q-type assumption is an assumption for which the size of the instance is parameterized by a polynomial q in the security parameter. Usually the polynomial q bounds the number of an adversary's queries the scheme can resists. Also note that q-type assumptions allow a more efficient generic attack on the underlying problem [Che06, SHI$^+$12] than static (non q-type) assumptions.

construction and the Groth-Sahai proof [GS08], to achieve the stronger decryption consistency than the first scheme, which is in precise as strong as that of the Boneh-Boyen-Halevi scheme [BBH06]. Furthermore, this specific construction archives asymptotically shorter ciphertext overhead than the first generic construction, as the first scheme has the ciphertext overhead proportional to n^2 in which n is the number decryption servers involved in the ciphertext, while the second scheme has the overhead of proportional to n.

Our results highlight usefulness of PKENO to construct threshold PKE schemes, as both of the above two result make use of that as a central building block. We highlight that the usability of PKENO for constructing TPKE has been conjectured by Galindo et al. [GLF+10], but this conjecture is not investigated in detail.

We revisit their conjecture, and show that there exists some subtlety in decryption consistency. In particular, as Galindo et al. suggested, our construction uses multiple encryption of PKENO. In addition, we show that for achieving decryption consistency we need another technique to detect a maliciously generated ciphertext. In the first construction we use a technique of verifiable secret sharing for this purpose. However, it cannot achieve the highest notion of decryption consistency, as mention above. If we admit deviating from a black-box construction, we can obtain as strong decryption consistency as achieved by several previous works, as shown in the second construction.

To further study the Galindo et al. conjecture, we lastly investigate the possibility of a black-box construction of TPKE from PKENO keeping the higher notion of decryption consistency. In fact, introducing another technique to ensure the validity of a ciphertext, we provide a affirmative answer by a black-box construction of TPKE scheme that provides strong decryption consistency. Drawbacks of this scheme is that it is no longer dynamic TPKE, and that the number of decryption servers the scheme can support is only logarithmic in the security parameter, rather than an arbitrary polynomial.

5.1.1.2 Our technique.

Our approach is based on the Dodis-Katz multiple encryption technique and further enhancing the ability to detect malicious behavior of both a sender and decryption servers. This is because, the Dodis-Katz multiple encryption scheme already can serve as dynamic TPKE except decryption consistency, as each decryption servers generate their own key pair by their own to join the system, and to specify the authorized set dynamically a sender simply picks the public keys of the servers that the sender wants to be authorized. Thus, to construct a dynamic TPKE scheme with decryption consistency, we need to improve the Dodis-Katz scheme to provide decryption consistency.

One of the possible (and simple) approach is to replace the underlying secret sharing scheme with a more enhanced scheme with some kind of cheating detection. However, this approach is not straightforward because most of the secret sharing scheme with cheating detection assumes that the shares are distributed honestly, and in such a case it is able to detect malicious share holders. In contrast, our setting of decryption consistency, even the shares are generated maliciously, as the shares are generated by a potentially malicious sender.

Instead, we need another mechanism to ensure the consistency between the multiple shares. Further difficulty is that these shares are encrypted as the Dodis-Katz scheme does. In this case, any single decryption servers cannot verify the consistency of the shares, as the server can only see a single share which is directed to that server. In particular, if we combine Shamir's k-out-of-n scheme with the Dodis-Katz multiple encryption, encrypted n shares should consist of degree-$(k-1)$ curve, otherwise different k shares result in different decryption results, and thus the decryption consistency will be violated. For decryption consistency, we need to extend the Dodis-Katz scheme to allow decryption servers to detect such a maliciously generated ciphertext.

To this end, we take three different approaches for each proposed scheme.

The first scheme combines the Dodis-Katz scheme with a technique from *verifiable secret sharing* [BOGW88, BKP11]. This is a classical technique to provide consistency between shares of Shamir's secret sharing scheme, and extensively studied mainly in the context of multiparty computations. We bring this technique to the context of non-interactive TPKE to construct a TPKE scheme with decryption consistency.

The second scheme is fairly simple. We use an non-interactive zero-knowledge proof to ensure consistency between the encrypted shares. This simplicity will be obtained at the cost of the non-black-box construction or quite restricted efficient instantiations. That is, the only known efficient instantiation of non-interactive zero-knowledge proof is restricted to the bilinear groups (especially the Groth-Sahai proof [GS08]), or at least this construction is no longer a black-box construction, as we need to employ the non-interactive zero-knowledge proof for general NP-languages.

The third scheme takes slightly different approach. This scheme uses the technique of *multiple-assignment secret sharing* [ISN93], which is originally developed for constructing secret sharing schemes supporting general access structure. Interestingly, as shown in this paper, this technique is also useful to ensure decryption consistency of a TPKE scheme. The key technique is instantiating a multiple-assignment scheme with n-out-of-n secret sharing. A useful property of n-out-of-n sharing is that any combination of n shares can be a set of honestly generated shares, whereas in k-out-of-n sharing ($k < n$) there are invalid, and thus potentially dangerous for decryption consistency, combinations of n shares. If we want to use k-out-of-n sharing for constructing a TPKE scheme, we need to exclude such a potentially dangerous combination of shares with some additional mechanism. In contrast, if we only use an n-out-of-n sharing scheme, no such dangerous combination exists, thus we have no need to manage such mechanism for detecting dangerous ciphertexts anymore.

5.1.1.3 Related work.

Our first proposed scheme includes a commitment in the ciphertext to achieve certain kind of decryption consistency. Similar techniques of including a commitment in the ciphertext are often proposed in the literature, for various purpose. Abdalla, Bellare, and Neven proposed robustness notion of the ordinary public-key encryption [ABN10], which requires that a single ciphertext should not be decrypted successfully decrypted by two or more decryption keys, and used a similar technique to achieve this robustness. This technique is reminiscent of the improvement of the Canetti-Halevi-Katz transformation by Boneh and Katz [BK05], which uses an encapsulation scheme (a commitment scheme to a random string) instead of a normal commitment scheme.

Shoup and Gennaro formalized decryption consistency and proposed two schemes that achieve this notion [SG02]. These schemes are respectively based on the computational and decision Diffie-Hellman assumption, together with using random oracles. Boneh, Boyen, and

Halevi proposed a TPKE scheme with decryption consistency which is no longer relying on random oracles [BBH06]. Their scheme is proved under the decision bilinear Diffie-Hellman assumption.

Dodis and Katz proposed the multiple encryption technique that preserves chosen-ciphertext security of the underlying encryption scheme, and applies this technique to construct TPKE scheme [DK05]. They formalized several notions of message privacy named MCPA, wMCCA, MCCA, and sMCCA. These four security notions are in fact concentrated on the secrecy of the plaintext, rather than resilience of decryption process against maliciously behaved sender and receivers (decryption servers), and thus are independent notions from decryption consistency. In the same paper the authors also discuss decryption robustness. This notion is more related to decryption consistency, however, it ensures that under the assumption that a ciphertext is a honestly generated encryption of M, how many honestly derived decryption shares are sufficient to recover the M successfully, even when that honest shares are mixed with maliciously generated shares. In contrast, decryption consistency requires that even when a ciphertext and its decryption shares are generated maliciously, the result of combining shares should be uniquely determined.

Emura, Hanaoka, and Sakai [EHS10] claimed that group signature can be transformed to PKENO and TPKE. However, their first approach is not quite matured, and is revised by the same authors. They finally showed that the underlying group signature scheme needs to have an additional property called opening soundness [SSE$^+$12a], and gave a rigorous proof of the fact that any group signature scheme with opening soundness can be transformed to a PKENO scheme [EHSS13].

5.2 New Definitions

In this section we introduce new definitions used in the rest of this chapter. The definitions includes a dynamic extension of the threshold encryption, a weaker variant of decryption consistency, and a labeled variant of the PKENO primitive.

5.2.1 Threshold Public-Key Encryption

First we define an extension of the threshold encryption primitive that allows dynamic joining of decryption servers. We then define an weaker variant of decryption consistency which allows a decryption server to claim, in a publicly verifiable way, the ciphertext is partially invalid so that it is impossible to provide a decryption share which can be securely combined.

Intuitively this weaker decryption consistency allows each decryption servers to be unable to verify the *entire* ciphertext. In fact we give two different variant of decryption consistency, namely the weak and strong decryption consistency. While the "strong" notion exactly follows the definition in [BBH06], the "weak" notion is introduced to capture the decryption consistency achieved by one of our proposed scheme. More detailed discussion on this weak notion is given in the paragraphs after Definition 5.1.

To define the relaxed version of decryption consistency, at first we allow the share verification algorithm to have a ternary output (rather than the binary $\top$ and $\bot$).

$\mathsf{ThVerify}$**.** The share verification algorithm $\mathsf{ThVerify}$ takes as input the public key pk, the verification key vk, a ciphertext C, and a decryption share μ. It outputs either $\top_{\text{valid}}$, $\top_{\text{invalid}}$, or $\bot$.

The second correctness condition is also modified correspondingly: for all $\lambda \in \mathbb{N}$, any integers n and k ($1 \leq k \leq n$), any $(pk, vk, (sk_i)_{i\in[n]}) \leftarrow \mathsf{ThKg}(1^\lambda, n, k)$, it holds that

- for any C and any $\iota \in [n]$, $\mathsf{ThVerify}(pk, vk, C, \mathsf{ThDec}(pk, \iota, sk_\iota, C))$ outputs either $\top_{\text{valid}}$ or $\top_{\text{invalid}}$.

The definition of chosen-ciphertext security is unchanged.

Finally, the definition of decryption consistency will be changed with replacing the winning condition of the adversary as follows. After outputting a tuple $(C, ((\iota_i, \hat{\mu}_i))_{i\in[k]}, ((\iota'_i, \hat{\mu}'_i))_{i\in[k]})$, the adversary is declared to win the game if one of the following two conditions holds:

- The following three conditions holds (i) ι_1, ..., ι_k are mutually distinct, and

$\mathsf{ThVerify}(pk, vk, C, (\iota_i, \hat{\mu}_i)) = \top_{\text{valid}}$; (ii) the same holds for $((\iota'_i, \hat{\mu}'_i))_{i\in[k]}$; and (iii) $\mathsf{ThCombine}(pk, vk, C, ((\iota_i, \hat{\mu}_i))_{i\in[k]}) \neq \mathsf{ThCombine}(pk, vk, C, ((\iota'_i, \hat{\mu}'_i))_{i\in[k]})$.

- There exists $\iota, \hat{\mu}, \hat{\mu}'$ that satisfy $(\iota, \hat{\mu}), (\iota, \hat{\mu}') \in \{(\iota_1, \hat{\mu}_1), \ldots, (\iota_k, \hat{\mu}_k), (\iota'_1, \hat{\mu}'_1), \ldots, (\iota'_k, \hat{\mu}'_k)\}$, $\mathsf{ThVerify}(pk, vk, C, (\iota, \hat{\mu})) = \top_{\text{valid}}$, and $\mathsf{ThVerify}(pk, vk, C, (\iota, \hat{\mu}')) = \top_{\text{invalid}}$.

We denote this modified experiment by $\text{Exp}^{(k,n)\text{-wConsistent}}(\lambda)$.

Definition 5.1. A threshold encryption scheme ($\mathsf{ThKg}, \mathsf{ThEnc}, \mathsf{ThDec}, \mathsf{ThVerify}, \mathsf{ThCombine}$) (with the extended syntax) has weak decryption consistency if for any integer k and n ($0 \leq k \leq n$) and any probabilistic polynomial-time adversary $\mathcal{A}$ the advantage $\text{Adv}_{\mathcal{A}}^{(k,n)\text{-wConsistent}}(\lambda) = \Pr[\text{Exp}^{(k,n)\text{-wConsistent}}(\lambda) = 1]$ is negligible in λ.

Note that in the above definition, the adversary is provided with all the secret keys of the decryption servers, rather than those of only k servers. This stronger type of definition was introduced by Galindo et al. [GLF+10], in order to prove a generic construction of PKENO from TPKE. Our result can be seen as the converse of [GLF+10].

The $\mathsf{ThVerify}$ algorithm has three possible outputs $\top_{\text{valid}}$, $\top_{\text{invalid}}$, and $\perp$ rather than just binary values $\top$ and $\perp$. In particular, if the ciphertext is not publicly verifiable, it might be the case that some servers receive a valid share, while the other servers are unable to obtain any valid shares. Furthermore, due to the lack of public verifiability of the ciphertext, a server that receives a valid share cannot convince himself that the other servers also receive a valid share, and a server that does not receive a valid share cannot convince himself that the other servers also do not receive a valid share. This situation make it harder to agree on the validity/invalidity of the entire ciphertext among the servers.

Our definition of the weak decryption consistency tries to relax this difficulty by allowing each servers to claim the validity/invalidity of their received shares individually. That is, a server that receives a valid share is required to produce a publicly verifiable proof for the validity of the share, while a server that receives an invalid share is required to produce a publicly verifiable proof for the invalidity of the share. The $\mathsf{ThVerify}$ algorithm verifies these proofs and outputs $\top_{\text{valid}}$ if the server's claim of validity is (considered to be) true, $\top_{\text{invalid}}$ if the server's claim of invalidity is true $\perp$ if the server's claim is simply false. The definition of the weak decryption consistency requires that if servers' claim of validity are verified as correct, any combination of such shares should be agree on the decryption result. However, if a server claims that the received share is invalid (and the claim is verified by $\mathsf{ThVerify}$), we no longer require this server to produce any shares that can be securely combined.

We further introduce a stronger definition of decrytption consistency called *strong decrytption consistency*. It requires the $\mathsf{ThVerify}$ algorithm to output either $\top_{\text{valid}}$ or $\perp$ but never output $\top_{\text{invalid}}$. The strong decrytption consistency is actually equivalent to the decrytption consistency defined by Boneh, Boyen, and Halevi [BBH06] except for the "known secret key" extension. The formal definition is as follows.

Definition 5.2. A TPKE scheme is said to have *strong decryption consistency* if in addition to the weak decryption consistency it satisfies the following condition: for any $(pk, vk, (sk_i)_{1\leq i\leq n}) \leftarrow \mathsf{ThKg}(1^\lambda, n, k)$, any ciphertext $C \in \{0,1\}^*$, and any decryption shares μ, $\mathsf{ThVerify}(pk, vk, C, (\mu_i)_{1\leq i\leq k})$ outputs either $\top_{\text{valid}}$ or $\perp$.

5.2.2 Dynamic Threshold Public-key Encryption

We then give a formal definition of the dynamic TPKE. The definition basically follows the definition given by Delerablée and Pointcheval [DP08] again with the known-secret-keys extension. However the description of the security game is modified to the public-key setting rather than the identity-based setting.

A dynamic TPKE scheme consists of the following probabilistic polynomial-time algorithms:

DSetup. The setup algorithm DSetup takes as input the security parameter 1^λ and outputs the public parameter pk and the master secret key mk. The public parameter pk implicitly specifies the message space $\mathcal{M}_{pk}$, which determines the set of plaintext that can be encrypted under that public key.

DJoin. The join algorithm takes as input the public parameter pk and the master secret key mk, and outputs a pair (upk, usk) of the public and secret keys for a new user. We assume that the user public key upk is made publicly available.

DEnc. The encryption algorithm DEnc take as input the public parameter pk, n public keys upk_1, ..., upk_n, the threshold k, and the plaintext m to be encrypted and outputs a ciphertext C.

DDec. The partial decryption algorithm DDec takes as input the public parameter pk, the public and secret keys of some user, and a ciphertext C and outputs the decryption share (upk, μ).

$\mathsf{DVerify}$. The verification algorithm $\mathsf{DVerify}$ takes as input the public parameter pk, the public key upk of some user, a ciphertext C, and its decryption share μ by (the owner of) upk and outputs either $\top_{\text{valid}}$, $\top_{\text{invalid}}$, or $\perp$.

$\mathsf{DCombine}$. The combining algorithm $\mathsf{DCombine}$ takes as input the security parameter pk, a ciphertext C and k decryption shares $\mu_1, \ldots, \mu_k$ and outputs a plaintext m or $\perp$.

We require dynamic TPKE schemes to satisfy the following correctness conditions: for any integer n and k ($n \geq k$), any honestly generated $(pk, mk) \leftarrow \mathsf{DSetup}(1^\lambda)$, and any n honestly generated users' key pair $(upk_1, usk_1) \leftarrow \mathsf{DJoin}(pk, mk)$, ..., $(upk_n, usk_n) \leftarrow \mathsf{DJoin}(pk, mk)$, it is required that

- for any plaintext m, honestly generated ciphertext $C \leftarrow \mathsf{DEnc}(\pi, upk_1, \ldots, upk_n, k, m)$, and any size-$k$ subset $\{\iota_1, \ldots, \iota_k\} \subset [n]$, if one honestly computes decryption shares as $\mu_i \leftarrow \mathsf{DDec}(pk, upk_{\iota_i}, usk_{\iota_i}, C)$ $(1 \leq i \leq k)$, then we have that $\mathsf{DCombine}(pk, C, \mu_1, \ldots, \mu_k) = m$, and
- for an arbitrary ciphertext C and any $1 \leq i \leq n$, if one honestly computes a decryption share $\mu \leftarrow \mathsf{DDec}(pk, upk_i, usk_i, C)$, then we have that $\mathsf{DVerify}(pk, upk, C, \mu)$ is either $\top_{\text{valid}}$ or $\top_{\text{invalid}}$.

The security requirements for dynamic TPKE are defined by extending those of (non-dynamic) TPKE. We firstly describe the secrecy requirement by the following game between a challenger and an adversary:

```
$\mathrm{Exp}^{\text{dynamic-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$:
  $b \leftarrow \{0,1\}$
  $(pk, mk) \leftarrow \mathsf{DSetup}(1^\lambda)$
  $(upk_i, usk_i) \leftarrow \mathsf{DJoin}(pk, mk)\ (i \in [N])$
  $(\tilde{upk}_i, \tilde{usk}_i) \leftarrow \mathsf{DJoin}(pk, mk)\ (i \in [\tilde{N}])$
  $(m_0, m_1, S, state) \leftarrow \mathcal{A}^{\mathsf{DDec}(pk, upk_\cdot, usk_\cdot, \cdot)}(pk, (upk_i)_{i\in[N]}, (\tilde{upk}_i, \tilde{usk}_i)_{i\in[\tilde{N}]})$
    where $S = \{upk^*_1, \ldots, upk^*_n\} \subset \{upk_1, \ldots, upk_n, \tilde{upk}_1, \ldots, \tilde{upk}^*_n\}$
          and $S$ include (at most) $k-1$ corrupted keys from $\{\tilde{upk}_1, \ldots, \tilde{upk}_{\tilde{N}}\}$
  $C^* \leftarrow \mathsf{DEnc}(pk, upk^*_1, \ldots, upk^*_n, k, m_{b)})$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{DDec}(pk, upk_\cdot, usk_\cdot, \cdot)}(C^*, state)$
  return $(b = b')$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2)$ is required to output m_0 and m_1 with $|m_0| = |m_1|$, Furthermore, $\mathcal{A}_2$ not to submit any query (i, C^*) with arbitrary i.

Definition 5.3. A dynamic TPKE scheme ($\mathsf{DSetup}, \mathsf{DJoin}, \mathsf{DEnc}, \mathsf{DDec}, \mathsf{DVerify}, \mathsf{DCombine}$) is chosen-ciphertext secure if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$ and any N, $\tilde{N} \in \mathbb{N}$ the advantage $\mathrm{Adv}^{\text{dynamic-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}^{\text{dynamic-CCA}}_{(\mathcal{A}_1,\mathcal{A}_2)}(\lambda) = 1] - 1|$ is negligible in λ.

We then define the decrytption consistency requirement by the following game.

```
$\mathrm{Exp}^{\text{dynamic-Consistent}}_{\mathcal{A}}(\lambda)$:    $(pk, mk) \leftarrow \mathsf{DSetup}(1^\lambda)$
  $(\tilde{upk}_i, \tilde{usk}_i) \leftarrow \mathsf{DJoin}(pk, mk)\ (i \in [\tilde{N}])$
  $(C, ((\iota_j, \hat{\mu}_j))_{j\in[k]}, ((\iota'_j, \hat{\mu}'_j))_{j\in[k]}) \leftarrow \mathcal{A}(pk, (\tilde{upk}, \tilde{usk})_{i\in[\tilde{N}]})$
  return 1 if one of the following two conditions holds.
```

Here the winning condition is defined by the following two condition:

1. Both S and S' consists of k decryption shares from k distinct servers, in which we assume the threshold associated with a ciphertext can be publicly determined and denote this by k, S and S' are not equal to each other as sets, all shares $\mu \in S \cup S'$ satisfy $\mathsf{DVerify}(pk, upk, C, \mu) = \top_{\text{valid}}$ where upk is the user public key of the corresponding decryption servers of μ, and $\mathsf{DCombine}(pk, C, S) \neq \mathsf{DCombine}(pk, C, S')$.
2. There exists μ, $\mu' \in S \cup S'$ which are both attributed to the same decryption server, $\mathsf{DVerify}(pk, upk, C, \mu) = \top_{\text{valid}}$, and $\mathsf{DVerify}(pk, upk, C, \mu') = \top_{\text{invalid}}$, in which upk is

the user public key of the decryption server of μ and μ'.

The adversary wins if one of the above two conditions hold.

Definition 5.4. A dynamic TPKE scheme (DSetup, DJoin, DEnc, DDec, DVerify, DCombine) is weak decryption consistency if for all probabilistic polynomial-time adversary $\mathcal{A}$ and any $\tilde{N} \in \mathbb{N}$ the advantage $\mathrm{Adv}_{\mathcal{A}}^{\text{dynamic-Consistency}}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}_{\mathcal{A}}^{\text{dynamic-Consistency}}(\lambda) = 1] - 1|$ is negligible in λ.

As in the non-dynamic version of decryption consistency, this definition is given in the "known secret keys" manner. Also as in the previous, we can define the strong decryption consistency of a dynamic TPKE scheme. The exact definition of the strong decryption consistency is given quite similarly to the non-dynamic version, hence we omit the formal definition.

5.2.3 Public-Key Encryption with Non-Interactive Opening

As mentioned in the introduction, our generic construction is based on public-key encryption with non-interactive opening. Actually, we require the underlying scheme to support labels (or to be tag-based) [MRY04, Kil06], whose formal definition is as follows.

We define syntax and security of public-key encryption with non-interactive opening. A public-key encryption scheme with non-interactive opening consists of the following five algorithms.

NOKg. The key-generation algorithm NOKg takes as input a security parameter 1^λ and outputs a pair (ek, dk) of the encryption key and the decryption key. The public parameter pk implicitly specifies the message space $\mathcal{M}_{pk}$, which determines the set of plaintext that can be encrypted under that public key.

NOEnc. The encryption algorithm NOEnc takes as input the encryption key ek, a label L, and a plaintext m. It outputs a ciphertext c.

NODec. The decryption algorithm NODec takes as input the decryption key dk, a label L, and a ciphertext c. It outputs a plaintext m or a special symbol $\perp$.

NOProve. The proof algorithm NOProve takes as input the decryption key dk, a label L, and a ciphertext c. It outputs a proof π.

NOVerify. The verification algorithm NOVerify takes as input the encryption key ek, a label L, a ciphertext c, a plaintext m, and a proof π, and outputs a bit 1 or 0, indicating the proof is respectively *valid* or *invalid*.

As the correctness condition, the labeled PKENO scheme is required to satisfy the following conditions.

- For any $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$, any plaintext $m \in \mathcal{M}_{pk}$ and any label $L \in \{0, 1\}^*$, it holds that $\mathsf{NODec}(dk, L, \mathsf{NOEnc}(pk, L, m)) = m$.
- For any $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$, any ciphertext $c \in \{0, 1\}^*$, and any label $L \in \{0, 1\}^*$, it holds that $\mathsf{NOVerify}(ek, L, c, \mathsf{NODec}(dk, L, c), \mathsf{NOProve}(dk, L, c)) = 1$.

Notice that in the latter conditions the ciphertext c is not restricted to the legitimate output of the encryption algorithm $\mathsf{NOEnc}(ek, L, m)$ with some L and m, and hence $\mathsf{NODec}(dk, L, c)$ potentially would be $\perp$.

We require the labeled PKENO scheme to be *selective-label weak chosen-ciphertext secure* and *strongly committing*. Although the former requirement for PKENO schemes has not been formally stated in the literature, it is a straightforward adoption of the similar requirement for ordinary (tag-based) public-key encryption schemes formalized by Kiltz [Kil06]. The latter requirement is originally formalized by Galindo et al. [GLF+10]. More precisely our definition is a slightly weaker variant than that of Galindo et al.[GLF+10], as our definition requires the target key pair to be generated honestly. It is also worth noting that the requirement of *proof soundness*, which is defined by Damgård et al., is implied by our definition.

The requirement of selective-label weak chosen-ciphertext security is defined by the following game between challenger and the adversary $\mathcal{A}$.

```
$\mathrm{Exp}^{\text{PKENO-sLabel-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda)$:
  $b \leftarrow \{0,1\}$
  $(L^*, state_1) \leftarrow \mathcal{A}_1(1^\lambda)$
  $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$
  $(m_0, m_1, state_2) \leftarrow \mathcal{A}_1{}^{\mathsf{NODec}(dk,\cdot,\cdot),\mathsf{NOProve}(dk,\cdot,\cdot)}(state_1, ek)$
  $c^* \leftarrow \mathsf{NOEnc}(ek, L^*, m_b)$
  $b' \leftarrow \mathcal{A}_2{}^{\mathsf{PDec}(dk,\cdot,\cdot),\mathsf{NOProve}(dk,\cdot,\cdot)}(state_2, c^*)$
  return $b = b'$
```

In the experiment, the adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ is required to output m_0 and m_1 with $|m_0| = |m_1|$, and not to submit (L^*, c) to its both oracles regardless of c throughout the experiment.

Definition 5.5. A PKENO scheme $(\mathsf{NOKg}, \mathsf{NOEnc}, \mathsf{NODec}, \mathsf{NOProve}, \mathsf{NOVerify})$ is selective-label weakly chosen-ciphertext secure if for all probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ the advantage $\mathrm{Adv}^{\text{PKENO-sLabel-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda) = |2 \cdot \Pr[\mathrm{Exp}^{\text{PKENO-sLabel-wCCA}}_{(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)}(\lambda) = 1] - 1|$ is negligible in λ.

The committing requirement is defined by the following game.

```
$\mathrm{Exp}^{\text{PKENO-Label-Commit}}_{\mathcal{A}}(\lambda)$:
  $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$
  $(c, L, m, \pi, m', \pi') \leftarrow \mathcal{A}(ek, dk)$
  return $(\mathsf{NOVerify}(ek, c, L, m, \pi) = \top) \wedge (\mathsf{NOVerify}(ek, c, L, m', \pi') = \top)$
           $\wedge\ (m \neq m')$
```

In the experiment, the plaintexts m and m' are required to be in $\mathcal{M}_{ek} \cup \{\perp\}$.

Definition 5.6. A PKENO scheme $(\mathsf{NOKg}, \mathsf{NOEnc}, \mathsf{NODec}, \mathsf{NOProve}, \mathsf{NOVerify})$ is committing if for all probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{PKENO-Label-Commit}}_{\mathcal{A}}(\lambda) = \Pr[\mathrm{Adv}^{\text{PKENO-Label-Commit}}_{\mathcal{A}}(\lambda) = 1]$ is negligible in λ.

5.3 Generic Construction of Dynamic Threshold PKE

In this section we present the first proposed construction, which can provide the dynamic TPKE functionality.

First we revisit the Galindo et al. conjecture [GLF+10], on which the first proposed scheme is based. The conjecture is that TPKE with decryption consistency can be constructed from the Dodis-Katz transformation with labeled PKENO. However, below we explain that it is not enough to achieve the decrytption consistency. Let m be a plaintext to be encrypted, and $f(x)$ be a $k-1$ degree polynomial with the constant term m. Then, a sender computes n PKENO ciphertexts of $f(i)$ for all $i \in [n]$ whose label is a verification key of an one-time signature. Finally, the sender makes a signature whose signed messages are all ciphertexts. By using the Prove algorithm of PKENO, the $\mathsf{TVerify}$ algorithm can be implemented. So, this construction seems to have decryption consistency, since m can be obtained from any k of ciphertexts. One problem of this construction is, however, that a sender may not encrypt $f(i)$. That is, it is not guaranteed that all corresponding plaintexts are on the same polynomial f. A trivial attack is as follows. Let $k = 2$ and $n = 3$ for the simplicity. Then, a sender chooses two degree-1 polynomial f and g where $f(2) = g(2)$, computes PKENO ciphertexts for $f(1)$, $f(2)$, and $g(3)$, respectively. As in the case that all ciphertexts, say C_1, C_2, C_3, are correctly generated, all PKENO ciphertexts are valid in the sense of the $\mathsf{TVerify}$ algorithm. However, the $\mathsf{TCombine}$ algorithm outputs $f(0)$ from (C_1, C_2), and outputs $g(0)$ from (C_2, C_3), which contradicts decryption consistency. Namely, even if all decryption shares are valid, there is no guarantee that any combinations of k decryption shares yield the same plaintext.

The problem explained above is that a sender may not encrypt plaintexts which are on the same polynomial. In our construction, we adopt a technique from verifiable secret sharing [BKP11, BOGW88] to resolve this problem.

$\mathsf{DSetup}(1^\lambda)$. Generate the commitment parameter ck as $ck \leftarrow \mathsf{ComKg}(1^\lambda)$, set $pk = ck$ and $mk = \emptyset$, and output (pk, mk).

$\mathsf{DJoin}(pk, mk)$. Generate the public and secret keys (ek, dk) of the PKENO scheme by running $(ek, dk) \leftarrow \mathsf{NOKg}(1^\lambda)$. Set $(upk, usk) = (ek, dk)$ and output (upk, usk).

$\mathsf{DEnc}(pk, upk_1, \ldots, upk_n, k, m)$. Let ck be pk and ek_i be upk_i for all $i \in [n]$, and proceed as follows:

- Generate a key pair $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow \mathsf{SgKg}(1^\lambda)$ for the one-time signature scheme.
- Choose a random bivariate polynomial $f(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{i,j} x^i y^j$ of degree $k-1$ with $a_{0,0} = m$ and $f(i, j) = f(j, i)$ for all i and j.

- Compute commitments and their decommitments of $f(i,j)$ for $1 \leq i \leq j \leq n$ as $(c_{i,j}, r_{i,j}) \leftarrow \mathsf{Commit}(ck, f(i,j))$.
- Let $c_{j,i} = c_{i,j}$ and $r_{j,i} = r_{i,j}$ for $1 \leq i < j \leq n$.
- Compute PKENO ciphertext as $C_i = \mathsf{NOEnc}_{pk_i}(vk_{\mathrm{sots}}, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n}\rangle)$ for $i = 1, \ldots, n$.
- $\sigma_{\mathrm{sots}} \leftarrow \mathsf{SgSign}_{sk_{\mathrm{sots}}}(\langle k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}\rangle)$.
- Output $C = (vk_{\mathrm{sots}}, k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}, \sigma_{\mathrm{sots}})$.

$\mathsf{DDec}(pk, upk, usk, C)$**.** Parse C as $(vk_{\mathrm{sots}}, k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}, \sigma_{\mathrm{sots}})$ and find i such that $upk_i = upk$. If no such i is found, output $(upk, \perp)$. Otherwise, proceed as follows.

- Output $(upk, \perp)$ if $\mathsf{SgVerify}_{vk_{\mathrm{sots}}}(\langle k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}\rangle) = 0$.
- Decrypt C_i as $\hat{m} \leftarrow \mathsf{NODec}_{usk}(vk_{\mathrm{sots}}, C_i)$.
- Compute a proof π as $\pi \leftarrow \mathsf{NOProve}_{usk}(vk_{\mathrm{sots}}, C_i)$.
- Output $\mu_i = (upk, (\hat{m}, \pi))$.

$\mathsf{DVerify}(pk, vk, C, \mu)$**.** Parse C as $(vk_{\mathrm{sots}}, k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}, \sigma_{\mathrm{sots}})$ and parse μ as $(upk, \hat{\mu})$. Find i satisfying $upk = upk_i$. If no such i exists, output $\perp$ immediately. If such i exists, run $\mathsf{SgVerify}(vk_{\mathrm{sots}}, \langle k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}\rangle, \sigma_{\mathrm{sots}})$ to verify the one-time signature σ_{sots} and proceeds as follows.

- If the one-time signature is invalid and $\hat{\mu} = \perp$, output $\top_{\mathrm{valid}}$.
- If the one-time signature is valid, $\hat{\mu}$ is parsed as $(\hat{m}, \pi)$, and $\mathsf{NOVerify}(upk_i, C_i, \hat{m}, \pi) = 1$, further verify the following three conditions:
  - $\hat{m}$ is parsed as $\langle f_1, \ldots, f_n, r_1, \ldots, r_n\rangle$,
  - $\mathsf{ComVerify}(ck, c_{i,j}, f_j, r_j) = 1$ (or $\mathsf{ComVerify}(ck, c_{j,i}, f_j, r_{j,i}) = 1$ for $j < i$) for all $j \in [n]$, and
  - $(f_1, \ldots, f_n)$ defines a degree-$(k-1)$ polynomial.

  If all of the three conditions holds, output $\top_{\mathrm{valid}}$. Otherwise output $\top_{\mathrm{invalid}}$.
- If neither two conditions hold, output $\perp$.

$\mathsf{DCombine}(pk, vk, C, \mu_1, \ldots, \mu_k)$**.** Parse C as $(vk_{\mathrm{sots}}, k, (upk_i)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}, \sigma_{\mathrm{sots}})$ and μ_i as $(\widehat{upk}_i, \hat{\mu}_i)$ for all $1 \leq i \leq k$.

- If there is at least one μ_i that is $\perp$, output $\perp$.
- Otherwise if all $\hat{\mu}_i$ are parsed as $((f_{i,1}, \ldots, f_{i,1}, r_{i,1}, \ldots, r_{i,n}), \pi_i)$, proceed as follows:
  - find t_i satisfying $upk_{t_i} = \hat{upk}_i$ for all i,
  - interpolate $(t_1, f_{t_i,t_1}), \ldots, (t_k, f_{t_i,t_k})$ to obtain a polynomial $g_{t_i}(x)$ for all i,
  - interpolate $(t_1, g_{t_1}(0)), \ldots, (t_k, g_{t_k}(0))$ to obtain a polynomial $g(y)$, and
  - output $g(0)$.

Security. Security of the above scheme is described as follows.

Theorem 5.7. *The construction is chosen-ciphertext secure if the PKENO scheme is selective-label weakly chosen-ciphertext secure, the commitment scheme is (computationally) hiding, and the one-time signature scheme is strongly unforgeable.*

Proof. Let $\mathcal{A}$ be an adversary of having the advantage ε in the chosen-ciphertext security game. We bound this advantage to be negligible by gradually changing the game as follows. In the following description, we denote by S_i the event in which the adversary correctly guess the bit in Game i.

Game 0. This is exactly same as in the original chosen-ciphertext security game.

Game 1. In this game, the decryption oracle is modified to reject any decryption query which reuses the one-time signature verification key from the challenge ciphertext regardless of validity of the one-time signature in the queries.

Game 2. Then we modify how the challenge ciphertext is computed, by replacing the uncorrupted ciphertexts with garbage. Let $C^* = (vk^*_{\text{sots}}, (upk^*_i)_{1 \le i \le n}, (c^*_{i,j})_{1 \le i \le j \le n}, (C^*_i)_{1 \le i \le n}, \sigma^*_{\text{sots}})$ be the challenge ciphertext. In this game, the components C^*_i are computed as follows:

$$C_i^* \leftarrow \begin{cases} \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}^*, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n}\rangle) \\ \qquad\qquad\qquad\qquad \text{if } upk_i^* \in \{\tilde{upk}_1, \ldots, \tilde{upk}_{\tilde{N}}\}, \\ \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}^*, \langle 0, \ldots, 0, 0^{|r_{i,1}|}, \ldots, 0^{|r_{i,n}|}\rangle) \\ \qquad\qquad\qquad\qquad \text{if } upk_i^* \in \{upk_1, \ldots, upk_N\}. \end{cases}$$

With this modification the advantage of the adversary only changes negligibly. See Lemma 5.9 for details.

Game 3. Finally we replace the commitments in the challenge ciphertext with garbage. More concretely we replace the commitments whose decommitment are not included in the corrupted ciphertexts. This modification does not introduce more than negligible difference on the advantage of the adversary. See Lemma 5.10 for detail.

Lemma 5.8. *If the one-time signature scheme is strongly unforgeable,* $|\Pr[S_0] - \Pr[S_2]|$ *is negligible.*

Proof (of Lemma 5.8). It can be shown by a standard argument using the difference lemma. The two games differs only when a ciphertext $C = (vk_{\mathrm{sots}}, k, (upk_i)_{1\le i\le n}, (c_{i,j})_{1\le i\le j\le}, (C_i)_{1\le i\le n}, \sigma_{\mathrm{sots}})$ with the following property is queried: (1) it reuses the verification key vk_{sots}^* from the challenge ciphertext, and (2) the one-time signature σ_{sots}^* is valid. Whenever we receive such a query from the adversary, we can obtain a strong forgery of the underlying one-time signature scheme. This fact proves that the difference $|\Pr[S_0] - \Pr[S_2]|$ is bounded by the probability that a polynomial-time algorithm outputs such a forgery, which is negligible by the assumption of the lemma. □

Lemma 5.9. *If the labeled PKENO scheme is selective-label weakly chosen-ciphertext secure,* $|\Pr[S_1] - \Pr[S_2]|$ *is negligible.*

Proof (of Lemma 5.9). The proof further proceeds with a sequence of (sub)games $G_{1,0}$, $G_{1,1}$, ..., $G_{1,N}$, in which $G_{1,0}$ is identical to Game 1 and $G_{1,N}$ is identical to Game 2. In the game $G_{1,l}$, the ciphertexts with public key $upk_1, \ldots$, upk_{l-1}, or upk_l are replaced with garbage, while these with upk_{l+1}, …, upk_{N-1}, upk_N are kept untouched. More formally, when the adversary submits the set $S = \{upk_1^*, \ldots, upk_n^*\}$ (together with the threshold k and two messages m_0 and m_1), then in the game $G_{1,l}$ the ciphertext C_i in the challenge ciphertext is computed as:

$$C_i \leftarrow \begin{cases} \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n} \rangle) \\ \hfill \text{if } upk_i^* \in \{\tilde{upk}_1, \ldots, \tilde{upk}_{\tilde{N}}\}, \\ \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}, \langle 0, \ldots, 0, 0^{|r_{i,1}|}, \ldots, 0^{|r_{i,n}|} \rangle) \\ \hfill \text{if } upk_i^* \in \{upk_1, \ldots, upk_l\}, \\ \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n} \rangle) \\ \hfill \text{if } upk_i^* \in \{upk_{l+1}, \ldots, upk_N\}. \end{cases}$$

Noticing that $G_{1,0}$ is equivalent to Game 1 and $G_{1,n-k+1}$ is to Game 2, we can conclude Lemma 5.9 by showing that for all $l \in \{1, \ldots, N\}$, $|\Pr[S_{1,l-1}] - \Pr[S_{1,l}]|$ is negligible.

Now we will show that $|\Pr[S_{1,l-1}] - \Pr[S_{1,l}]|$ is negligible for all $l \in \{1, \ldots, N\}$. To prove this we construct a simulator $\mathcal{B}_l$ which runs in the chosen-ciphertext security game of the PKENO scheme. The description of $\mathcal{B}$ is as follows:

Setup. The simulator $\mathcal{B}$ first chooses a key-pair $(vk_{\mathrm{sots}}^*, sk_{\mathrm{sots}}^*)$ of the one-time signature scheme. Then it outputs vk_{sots}^* as the label for the challenge. The simulator receives an encryption key ek of the PKENO scheme. To set up the TPKE scheme, the simulator $\mathcal{B}$ sets $upk_l = ek$. For all the other upk_i $(i \notin [N] \setminus \{l\})$, it generates $(ek_i, dk_i) \leftarrow \mathsf{NOKg}(1^\lambda)$ for all $i \neq \iota_l$ and sets $pk_i = ek_i$. Furthermore, for all $\tilde{upk}_i$ $(i \in [\tilde{N}])$ the simulator again generates $(\tilde{ek}_i, \tilde{dk}_i) \leftarrow \mathsf{NOKg}(1^\lambda)$ and sets $(\tilde{upk}_i, \tilde{usk}_i) = (\tilde{ek}_i, \tilde{dk}_i)$. The simulator also runs $\mathsf{ComKg}(1^\lambda)$ to get a commitment key ck. Finally it sets $pk = ck$ and sends $(pk, (upk_i)_{i\in[N]}, (\tilde{upk}_i, \tilde{usk}_i)_{i\in[\tilde{N}]})$ to the adversary $\mathcal{A}$.

Query I. When the adversary $\mathcal{A}$ issues a decryption query (upk', C), the simulator responds as follows. First it parses C as $(vk_{\mathrm{sots}}, k, (upk_i')_{1\le i\le n}, (c_{i,j})_{1\le i\le j\le n}, (C_i)_{1\le i\le n}, \sigma_{\mathrm{sots}})$.

- If $vk_{\mathrm{sots}} = vk_{\mathrm{sots}}^*$ or $\mathsf{SgVerify}(vk_{\mathrm{sots}}, \langle k, (c_{i,j})_{1\le i\le j\le}, (C_i)_{1\le i\le n} \rangle, \sigma_{\mathrm{sots}}) = 0$, the simulator responds with $(upk', \perp)$.
- Let $i' \in [n]$ be the integer that satisfies $upk' = upk_{i'}'$. The simulator obtains the

decryption of $C_{i'}$ as follows. If $upk' = upk_l$, to which the simulator embeds the target PKENO scheme, the simulator issues a decryption query $(vk_{\text{sots}}, C_{i'})$ to its own decryption-and-prove oracle and receives a pair (m, π) of a message and a proof. The simulator responds with $(upk', (m, \pi))$.

- If $upk' = upk_i$ $(i \neq l)$, the simulator runs $\mathsf{NODec}(dk_i, vk_{\text{sots}}, C_i)$ to obtain m and runs $\mathsf{NOProve}(dk_i, vk_{\text{sots}}, C_i)$ to obtain π. The simulator responds with $(upk', (m, \pi))$.

Challenge. When the adversary $\mathcal{A}$ requests the challenge ciphertext by submitting two messages m_0 and m_1, an authorized set $\mathcal{S} = \{upk_1^*, \ldots, upk_n^*\}$, and a threshold k, the simulator $\mathcal{B}$ chooses a random bit b, choose a random symmetric bivariate polynomial $f(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{i,j} x^i y^j$ of degree $k - 1$ with $a_{0,0} = m_b$, computes commitments and decommitments $(c_{i,j}, r_{i,j}) \leftarrow \mathsf{Commit}(ck, f(i, j))$ for $1 \leq i \leq j \leq n$, and sets $c_{j,i} = c_{i,j}$ and $r_{j,i} = r_{i,j}$ for $1 \leq i < j \leq n$. If $upk_l = upk_{i^*}^*$ with some i^*, then the simulator submits two plaintexts $M_0 = \langle f(i^*, 1), \ldots, f(i^*, n), r_{i^*,1}, \ldots, r_{i^*,n} \rangle$ and $M_1 = \langle 0, \ldots, 0, 0^{|r_{i^*,1}|}, \ldots, 0^{|r_{i^*,n}|} \rangle$ to the challenger of the PKENO game. Receiving the challenge ciphertext c^*, $\mathcal{B}$ computes n ciphertexts of the PKENO scheme as follows:

$$C_i^* \leftarrow \begin{cases} \mathsf{NOEnc}(upk_i^*, vk_{\text{sots}}^*, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n} \rangle) \\ \hfill \text{if } upk_i^* \in \{\tilde{upk}_1, \ldots, \tilde{upk}_{\tilde{N}}\}, \\ \mathsf{NOEnc}(upk_i^*, vk_{\text{sots}}^*, \langle 0, \ldots, 0, 0^{|r_{i,1}|}, \ldots, 0^{|r_{i,n}|} \rangle) \\ \hfill \text{if } upk_i^* \in \{upk_1, \ldots, upk_{l-1}\}, \\ c^* \hfill \text{if } upk_i^* = upk_l, \\ \mathsf{NOEnc}(upk_i^*, vk_{\text{sots}}^*, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n} \rangle) \\ \hfill \text{if } upk_i^* \in \{upk_{l+1}, \ldots, upk_N\}. \end{cases}$$

Finally $\mathcal{B}$ generates a one-time signature $\sigma_{\text{sots}}^* \leftarrow \mathsf{SgSign}(sk_{\text{sots}}^*, \langle k, (upk_i^*)_{1 \leq i \leq n}, (c_{i,j})_{1 \leq i \leq j \leq n}, (C_i)_{1 \leq i \leq n} \rangle)$. If $upk_l \notin \mathcal{S}$, the simulator $\mathcal{B}$ ignores the PKENO challenge ciphertext c^* and computes the challenge ciphertext for the TPKE scheme as in the game l (or the game $l - 1$. The choice now does not matter since $upk_l \notin \mathcal{S}$). In any case, the challenge ciphertext $C^* = (vk^*_{\text{sots}}, k, (upk^*_i)_{1\le i\le n}, (c_{i,j})_{1\le i\le j\le n}, (C_i)_{1\le i\le n}, \sigma_{\text{sots}})$ is sent to the adversary $\mathcal{A}$.

Query II. After receiving the challenge ciphertext, $\mathcal{A}$ again issues decryption queries, which are responded as in the first query phase by the simulator $\mathcal{B}$.

Guess. Finally the adversary $\mathcal{A}$ outputs a guess bit b' and halts. The simulator then outputs 1 if $b = b'$, otherwise outputs 0.

In the above simulation, if the simulator $\mathcal{B}_l$ receives a ciphertext of M_0, the view of the adversary $\mathcal{A}$ is equivalent to $G_{1,l-1}$, and otherwise it is equivalent to $G_{1,l}$. Hence we have that

$$\Pr[S_{1,l-1}] - \Pr[S_{1,l}] = \Pr[\mathcal{B}_l \to 1 | b = 0] - \Pr[\mathcal{B}_l \to 1 | b = 1].$$

The right-hand side is negligible because of the assumption that the PKENO scheme is selective-label weak chosen-ciphertext secure. □

Lemma 5.10. *Suppose that the commitment scheme is (computationally) hiding. Then* $|\Pr[S_2] - \Pr[S_3]|$ *is negligible.*

Proof (of Lemma 5.10). To prove the lemma we construct a simulator $\mathcal{B}$ which runs in the left-or-right game of the commitment scheme. The description of $\mathcal{B}$ is as follows.

Setup. The simulator $\mathcal{B}$ starts with receiving the commitment key ck form the challenger. The simulator generates N keys of the PKENO scheme as $(ek_i, dk_i) \leftarrow \mathsf{NOKg}(1^\lambda)$ for all $i \in [N]$ and $\tilde{N}$ keys of the same PKENO scheme as $(\tilde{ek}_i, \tilde{dk}_i) \leftarrow \mathsf{NOKg}(1^\lambda)$. The simulator also generates a key $(vk^*_{\text{sots}}, sk^*_{\text{sots}})$ by running $\mathsf{SgKg}(1^\lambda)$. The simulator sets $pk = ck$, $(upk_i, usk_i) = (ek_i, dk_i)$ for all $i \in [N]$, and $(\tilde{upk}_i, \tilde{usk}_i) = (\tilde{ek}_i, \tilde{dk}_i)$ for all $i \in [\tilde{N}]$, and run the adversary $\mathcal{A}$ with $(pk, (upk_i)_{i\in[N]}, (\tilde{upk}_i, \tilde{usk}_i)_{i\in[\tilde{N}]})$ as input.

Query I. To a share-decryption query (upk', C) $\mathcal{B}$ responds as described by Game 2 (or 3). It can be done since $\mathcal{B}$ knows the user secret keys $usk_1, \ldots, usk_N$ of all the uncorrupted user public keys $upk_1, \ldots, upk_N$.

Challenge. When the adversary $\mathcal{A}$ outputs two messages m_0 and m_1 together with an authorized set $\mathcal{S} = \{upk^*_1, \ldots, upk^*_n\}$ and a threshold k, the simulator $\mathcal{B}$ proceeds as follows. It first chooses a random bit b, a random symmetric bivariate polynomial

$f(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{i,j} x^i y_j$ with $a_{0,0} = m_b$. Then computes commitments $c_{i,j}$ ($1 \leq i \leq j \leq n$) as follows. For all (i, j) with either upk_i^* or upk_j^* corrupted, the simulator runs $\mathsf{Commit}(ck, f(i, j))$ to obtain the commitment $c_{i,j}$ together with its decommitment $r_{i,j}$. For all (i, j) with both upk_i^* and upk_j^* uncorrupted, the simulator submits $(M_0, M_1) = (f(i, j), 0)$ to the left-or-right oracle to obtain a commitment $c_{i,j}$ of either $f(i, j)$ or 0. The simulator further computes PKENO ciphertexts $C_1, \ldots, C_n$ as follows:

$$C_i \leftarrow \begin{cases} \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}^*, \langle f(i,1), \ldots, f(i,n), r_{i,1}, \ldots, r_{i,n} \rangle) & \text{if } upk_i^* \text{ is corrupted,} \\ \mathsf{NOEnc}(upk_i^*, vk_{\mathrm{sots}}^*, \langle 0, \ldots, 0, 0 \cdots 0, \ldots, 0 \cdots 0 \rangle) & \text{if } upk_i^* \text{ is uncorrupted.} \end{cases}$$

Finally the simulator $\mathcal{B}_l$ runs $\mathsf{SgSign}(sk_{\mathrm{sots}}^*, \langle k, (upk_i^*)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n} \rangle)$ to obtains σ_{sots}^*, sets $C^* = (vk_{\mathrm{sots}}^*, k, (upk_i^*)_{1\leq i\leq n}, (c_{i,j})_{1\leq i\leq j\leq n}, (C_i)_{1\leq i\leq n}, \sigma_{\mathrm{sots}}^*)$, and sends the adversary C^*.

Query II. In this phase the adversary again issues decryption queries, which are responded as in the Query I phase.

Guess. Finally the adversary $\mathcal{A}$ outputs a guess b' and halts. The simulator $\mathcal{B}_l$ outputs 1 if $b = b'$ and outputs 0 if $b \neq b'$.

In this simulation, if the simulator receives commitments of m_0's, the adversary's view is identical to G_2, and if the simulator receives those of m_1's, the view is identical to G_3. Hence we have that $|\Pr[S_2] - \Pr[S_3]| = |\Pr[\mathcal{B} \to 1|\text{the commitments are of } m_0\text{'s}] - \Pr[\mathcal{B} \to 1|\text{the commitments are of 0's}]|$, whose right-hand side is negligible. □

Lemma 5.11. $\Pr[S_3] = 1/2$.

Proof (of Lemma 5.11). To see the lemma, we argue that for any view of the adversary, the number of polynomials consistent to the view with the constraint $f(0, 0) = m_0$ is equal to that of polynomial consistent to the view with the constraint $f(0, 0) = m_1$. Actually there is a single polynomial for each constraints $f(0, 0) = m_0$ and $f(0, 0) = m_1$, which we will show below.

In Game 3, the information on the polynomial f included in the adversary's view is the set $\{ f_{i,j} = f(i, j) \mid i \in S \text{ or } j \in S \}$ of evaluations of f, in which $S \subset [n]$ is the set of the corrupted servers. The constraints of $f_{i,j} = f(i, j)$ and $f(0,0) = m_0$ uniquely determines a symmetric degree-$(k-1)$ polynomial as seen below. Let us denote the set S by $\{i_1, \ldots, i_{k-1}\}$. From the n equations $f_{i_1,1} = f(i_1, 1), \ldots, f_{i_1,n} = f(i_1, n)$ there exists a degree-$(k-1)$ univariate polynomial f_{i_1} such that $f(x) = f(i_1, x)$ for all x. Similar things hold for $i_2, \ldots, i_{k-1}$, that is, there exist degree-$(k-1)$ polynomials $f_{i_2}, \ldots, f_{i_{k-1}}$ such that $f_{i_2}(x) = f(i_2, x), \ldots, f_{i_{k-1}}(x) = f(i_{k-1}, x)$ for all x. Furthermore, from n equations $f_{1,i_1} = f(1, i_1), \ldots, f_{n,i_1} = f(n, i_1)$, we can determine the evaluation $f(0, i_1)$. Similar thing can be applied to $i_2, \ldots, i_{k-1}$, that is, we can determine the evaluation of $f(0, i_2), \ldots, f(0, i_{k-1})$. Together with the assumption that $f(0,0) = m_0$, we can further determine a degree-$(k-1)$ polynomial f_0 such that $f_0(x) = f(0, x)$ for all x. These k polynomials $f_{i_1}, \ldots, f_{i_{k-1}}$, and f_0 determine the entire bivariate polynomial $f(x, y)$ uniquely. The same thing holds for the case that $f(0,0) = m_1$. □

These lemmas conclude the proof of the main theorem. □

Theorem 5.12. *The construction has the weak decryption consistency if the PKENO scheme is strongly committing and that the commitment scheme is (computationally) binding.*

Proof. Let $\mathcal{A}$ be an adversary of the decryption consistency game, $(pk, (\tilde{upk}_i, \tilde{usk}_i)_{i\in[\tilde{N}]})$ be the input to $\mathcal{A}$, and (C, S, S') be the output of $\mathcal{A}$. Let C be parsed as $(vk_{\text{sots}}, k, (upk_i^*)_{1\le i\le n}, (c_{i,j})_{1\le i\le j\le n}, (C_i)_{1\le i\le n}, \sigma_{\text{sots}})$. Let succ_1 and succ_2 be the events in which $\mathcal{A}$ successfully outputs two sets S and S' that meet the conditions 1. and 2. defined in the game for decrytption consistency, respectively. Let us denote S and S' as $S = \{(upk_1, \hat{\mu}_1), \ldots, (upk_k, \hat{\mu}_k)\}$ and $S' = \{(upk'_1, \hat{\mu}'_1), \ldots, (upk'_k, \hat{\mu}'_k)\}$. To prove the theorem, we show that both $\Pr[\mathsf{succ}_1]$ and $\Pr[\mathsf{succ}_2]$ is negligible.

We first bound $\Pr[\mathsf{succ}_2]$. In this case, the output of the adversary contains two decryption share $(upk, \hat{\mu})$ and $(upk, \hat{\mu}')$ of the same server upk. From these decryption shares we can obtain two proofs (of the PKENO scheme) which violate the committing property of the PKENO scheme. This enables us to construct a reduction algorithm that internally runs the adversary $\mathcal{A}$ and breaks the committing property of the PKENO. Let $(upk, \hat{\mu})$ be verified as $\top_{\text{valid}}$ and $(upk, \hat{\mu}')$ be as $\top_{\text{invalid}}$. Regarding to $(upk, \hat{\mu})$ we have the two possibilities that $\hat{\mu} = \perp$ and that $\hat{\mu}$ is parsed as $(\hat{m}, \pi)$. However, since $(upk, \hat{\mu}')$ is verified as $\top_{\text{invalid}}$, which implies that the one-time signature included in C is valid, we can exclude the first possibility and thus we have that $\hat{\mu}$ is parsed as $(\hat{m}, \pi)$ and $\mathsf{NOVerify}(\tilde{upk}_i, vk_{\mathrm{sots}}, C_i, \hat{m}, \pi) = 1$ for some i. Regarding to $(upk, \hat{\mu}')$, since it is verified as $\top_{\mathrm{invalid}}$, we also have that $\hat{\mu}'$ is parsed as $(\hat{m}', \pi')$ and $\mathsf{NOVerify}(\tilde{upk}_i, vk_{\mathrm{sots}}, C_i, \hat{m}', \pi') = 1$ for the same i as before. Furthermore, since $(upk, \hat{\mu})$ is verified as $\top_{\mathrm{valid}}$, $\hat{m}$ satisfies all of the three conditions described by the $\mathsf{DVerify}$ algorithm. In contrast to this, $\hat{m}'$ does not satisfy at least one of the same conditions, because $(upk, \hat{\mu}')$ is verified as $\top_{\mathrm{invalid}}$. Hence $\hat{m}$ must differ from $\hat{m}'$, thus the tuple $(vk_{\mathrm{sots}}, C_i, \hat{m}, \pi, \hat{m}', \pi')$ violates the committing property of the underlying PKENO scheme. More formally, we need to construct a simulator that receives a target key pair of the PKENO scheme and outputs a tuple that breaks the committing property. Such a simulator will choose random $i^* \in [\tilde{N}]$ and sets $(\tilde{upk}_{i^*}, \tilde{usk}_{i^*})$ to be the target keys received. When the adversary $\mathcal{A}$ outputs (C, S, S'), the simulator will compute a tuple $(vk_{\mathrm{sots}}, C_i, \hat{m}, \pi, \hat{m}', \pi')$ as described above and will output it. The simulator successfully breaks the committing property with probability $\Pr[\mathsf{succ}_2]/\tilde{N}$, which is negligible due to the committing property of the PKENO scheme.

Then we will bound $\Pr[\mathsf{succ}_1]$. Let

$$I = \{i \in [n] \mid (upk_i^*, \hat{\mu}) \in S \text{ for some } \hat{\mu}\}$$

and

$$I' = \{i \in [n] \mid (upk_i^*, \hat{\mu}) \in S' \text{ for some } \hat{\mu}\}.$$

We denote $I = \{i_1, \ldots, i_k\}$ and $I' = \{i'_1, \ldots, i'_k\}$. Using this notations we can denote S as $\{(upk^*_{i_1}, \hat{\mu}_1), \ldots, (upk^*_{i_k}, \hat{\mu}_k)\}$, and S' as $\{(upk^*_{i'_1}, \hat{\mu}'_1), \ldots, (upk^*_{i'_k}, \hat{\mu}'_k)\}$. Since these decryption shares are verified as $\top_{\mathrm{valid}}$, we have two possibilities that $\hat{\mu}_1 = \cdots = \hat{\mu}_k = \hat{\mu}'_1 = \hat{\mu}'_k = \bot$ and that none of $\hat{\mu}_1$, …, $\hat{\mu}_k$, $\hat{\mu}'_1$, …, $\hat{\mu}'_k$ is $\bot$. Since in the former case the decryption consistency will never break, we assume that the latter case occurs in the following. In the latter case we can further assume that $\hat{\mu}_\ell$ is parsed as $(\langle f_{i_\ell,1}, \ldots, f_{i_\ell,n}, r_{i_\ell,1}, \ldots, r_{i_\ell,n}\rangle, \pi_\ell)$ for all $\ell \in [k]$ and $\hat{\mu}'_{\ell'}$ is parsed as $(\langle f'_{i'_{\ell'},1}, \ldots, f'_{i'_{\ell'},n}, r'_{i'_{\ell'},1}, \ldots, r'_{i'_{\ell'},n}\rangle, \pi'_{\ell'})$ for all $\ell \in [k]$.

We first define an event bad_1 by the following condition: there exist integers $\ell \in [k]$ and $\ell' \in [k]$ such that the ℓ-th decryption share in S and ℓ'-th decryption share in S' are both generated by the same server and these shares do not agree in the $k-1$ curve. More formally, there exist two integer ℓ and ℓ' such that $i_\ell = i'_{\ell'}$ and $(f_{i_\ell,1}, \ldots, f_{i_\ell,n}) \neq (f'_{i'_{\ell'},1}, \ldots, f'_{i'_{\ell'}})$.

We further define an event bad_2 as follows. Intuitively the event bad_2 captures the case that the decryption shares submitted by the adversary do not constitute a symmetric bivariate polynomial. To state this more formally we will introduce some notations. Let us denote the union $I \cup I'$ to be $I \cup I' = \{j_1, \ldots, j_t\}$. Notice that t is either equal to $2k$ or smaller than $2k$ (The latter case occurs when some servers are involved in both S and S'). If we assume the event bad_1 does not occur, we can define the following $t \times t$ matrix:

$$\begin{pmatrix} f^*_{j_1,j_1} & f^*_{j_1,j_2} & \cdots & f^*_{j_1,j_t} \\ f^*_{j_2,j_1} & f^*_{j_2,j_2} & \cdots & f^*_{j_2,j_t} \\ \vdots & \vdots & \ddots & \vdots \\ f^*_{j_t,j_1} & f^*_{j_t,j_2} & \cdots & f^*_{j_t,j_t} \end{pmatrix}, \tag{5.1}$$

where $f^*_{u,v}$ is defined as

$$f^*_{u,v} = \begin{cases} f_{u,v} & (u \in I \setminus I') \\ f'_{u,v} & (u \in I' \setminus I) \\ f_{u,v} = f'_{u,v} & (u \in I \cap I'). \end{cases}$$

We say that the event bad_2 occurs if bad_1 does not occur and the above matrix is not symmetric.

Noticing that $\Pr[\mathsf{succ}_1] = \Pr[\mathsf{succ}_1 \wedge (\mathsf{bad}_1 \vee \mathsf{bad}_2)] + \Pr[\mathsf{succ}_1 \wedge \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2)]$ it is sufficient to show that the two terms are negligible for proving the theorem. The first term $\Pr[\mathsf{succ}_1 \wedge (\mathsf{bad}_1 \vee \mathsf{bad}_2)]$ is bounded by a standard reduction argument constructing an adversary that breaks the committing property of the PKENO scheme or the binding property of the commitment scheme. The other term $\Pr[\mathsf{succ}_1 \wedge \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2)]$ is actually equal to zero, which will be discussed below.

Denote by g_j the result of the interpolation from n points $f^*_{j,1}, \ldots, f^*_{j,n}$, which is actually the share submitted by the server j. To see that $\Pr[\mathsf{succ}_1 \wedge \neg(\mathsf{bad}_1 \vee \mathsf{bad}_2)]$, it is sufficient to see that the column vector

$$\vec{g} = \begin{pmatrix} g_{j_1} \\ g_{j_2} \\ \vdots \\ g_{j_t} \end{pmatrix} \tag{5.2}$$

is degree-$(k-1)$. If this column vector is degree-$(k-1)$, any choice of k points (out of t) results in the same point through the interpolation from k points. It ensures that the adversary has no chance to satisfy the conditions that $\mathsf{DCombine}(pk, vk, C, S) \neq \mathsf{DCombine}(pk, vk, C, S')$.

To see that the vector $\vec{g}$ is degree-$(k-1)$ we proceed as follows. Observing the fact that the n points $f^*_{j,1}, \ldots, f^*_{j,n}$ is degree-$(k-1)$, we first claim that even though g_j's are defined as the interpolation from the n points $f^*_{j,1}, \ldots, f^*_{j,n}$, interpolation from k points coincides in the result of interpolation. Particularly g_j can be written as the Lagrange interpolation from the k points $f^*_{j,j_1}, \ldots, f^*_{j,j_t}$ as

$$g_j = \lambda_{j_1} f^*_{j,j_1} + \lambda_{j_2} f^*_{j,j_2} + \cdots \lambda_{j_t} f^*_{j,j_t}.$$

Furthermore, $\vec{g}$ can be represented as the following linear combination:

$$\begin{pmatrix} g_{j_1} \\ g_{j_2} \\ \vdots \\ g_{j_t} \end{pmatrix} = \lambda_{j_1} \begin{pmatrix} f^*_{j_1,j_1} \\ f^*_{j_2,j_1} \\ \vdots \\ f^*_{j_t,j_1} \end{pmatrix} + \lambda_{j_2} \begin{pmatrix} f^*_{j_1,j_2} \\ f^*_{j_2,j_2} \\ \vdots \\ f^*_{j_t,j_2} \end{pmatrix} + \cdots + \lambda_{j_t} \begin{pmatrix} f^*_{j_1,j_t} \\ f^*_{j_2,j_t} \\ \vdots \\ f^*_{j_t,j_t} \end{pmatrix}. \tag{5.3}$$

This equation shows that if all of the t column vectors in the right-hand side of the linear combination is degree-$(k-1)$, so is the vector in the left-hand side. Since the matrix in Eq. (5.1) is symmetric and every column vectors of that matrix are degree-$(k-1)$ as verified by $\mathsf{DVerify}$, every row vectors are also degree-$(k-1)$. The t vectors appear in Eq. (5.3) is actually the row vectors of the matrix in Eq. (5.1), hence the vector $\vec{g}$ is degree-$(k-1)$. □

Instantiating the Generic Construction. To instantiate this generic construction, we need to have a "labeled" scheme of PKENO. Fortunately, to extend some known PKENO scheme to support labels is relatively straightforward. In particular, the PKENO scheme proposed by Galindo et al. [GLF+10] can be easily extended to the labeled scheme. For completeness, we provide the description of the labeled scheme and its security proof, together with its variant from the decision bilinear Diffie-Hellman assumption, in Appendix 5.6.

If we instantiate the generic construction with the PKENO scheme form the decision linear assumption, we can obtain the first dynamic TPKE scheme *from a static assumption* which archives the weak decryption consistency without relying random oracles. In addition, regarding the result by Emura et al. showing that a PKENO scheme can be constructed from an arbitrary group signature scheme (with a sufficiently but reasonably strong security notion) as a generic and black-box construction [EHSS13], interestingly we can obtain an (even dynamic) TPKE with the weak decryption consistency from an arbitrary group signature scheme as a generic construction. It would be of independent interest that these apparently unrelated two primitives of group signature and threshold encryption can be related only by means of generic constructions.

However, we need to admit our generic construction of dynamic TPKE is not as efficient as the Delerablée-Pointcheval scheme, as our construction has the ciphertext size proportional to the square of the number of the authorized servers, while the Delerablée-Pointcheval scheme has a constant size ciphertext. Furthermore, our scheme only achieves the weak decryption consistency rather than the strong notion.

In the next section we present a specific construction of dynamic TPKE that provides a more shorter, linear to the number of the authorized servers, ciphertext and the strong decryption consistency. The security of this specific construction comes from only the decision linear assumption.

5.4 Dynamic Threshold PKE from the Decision Linear Assumption

In this section we present a dynamic TPKE scheme with the strong decryption consistency. Security of this scheme solely comes from the decision linear assumption, while the Delerablée-Pointcheval scheme relies on a q-type assumption which is called MSE-DDH assumption. We also remark that the strong decryption consistency of our scheme does not require random oracles.

The description of the proposed scheme is as follows.

$\mathsf{DSetup}(1^\lambda)$**.** Run $\mathcal{G}(1^\lambda)$ to set up the bilinear group parameter $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Choose a common reference string $(\vec{f_1}, \vec{f_2}, \vec{f_3}) \in (\mathbb{G}^3)^3$ for the binding setting, where $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$, and $\vec{f_3} = \vec{f_1}^{\zeta_1}\vec{f_2}^{\zeta_2}$ for random $f_1, f_2 \in \mathbb{G} \setminus \{1\}$ and random ζ_1, $\zeta_2 \in \mathbb{Z}_p$. Set $pk = (p, \mathbb{G}, \mathbb{G}_T, e, g, \vec{f_1}, \vec{f_2}, \vec{f_3})$ and $mk = \emptyset$, and output (pk, mk).

$\mathsf{DJoin}(pk, mk)$**.** Generate public and secret keys of the PKENO scheme described in Sect. 5.6.2 by choosing random $x, y \leftarrow \mathbb{Z}_p$ and random $U, V \leftarrow \mathbb{G}$ and setting $u = g^x$ and $v = g^y$. Set $upk = (u, v, U, V)$ and $usk = (x, y)$ and output (upk, usk).

$\mathsf{DEnc}(pk, upk_1, \ldots, upk_n, k, m)$**.** Parse upk_i as (u_i, v_i, U_i, V_i) for all $i \in [n]$. Generate verification and signing keys (vk, sk) for a one-time signature scheme by running $\mathsf{SgKg}(1^\lambda)$. Choose random integers $r_1, \ldots, r_n, s_1, \ldots, s_n, a_1, \ldots, a_{k-1} \leftarrow \mathbb{Z}_p$ and computes $c_{i,1} \leftarrow {u_i}^{r_i}$, $c_{i,2} \leftarrow {v_i}^{s_i}$, $c_{i,3} \leftarrow (g^{vk}U_i)^{r_i}$, $c_{i,4} \leftarrow (g^{vk}V_i)^{s_i}$, and $c_{i,5} \leftarrow g^{r_i+s_i}mg^{a_1 i + a_2 i^2 + \cdots + a_{k-1} i^{k-1}}$ for all $i \in [n]$. Then compute a Groth-Sahai proof π^{zk} which demonstrates that the equations

$$c_{i,1} = {u_i}^{r_i},$$

$$c_{i,2} = {v_i}^{s_i},$$

$$c_{i,5} = g^{r_i}g^{s_i}(g^{i^{k-1}})^{a_{k-1}} \cdots (g^{i^2})^{a_2}(g^{i})^{a_1}m$$

for all $i \in [n]$ with witness $m \in \mathbb{G}$ and $r_1, \ldots, r_n, s_1, \ldots, s_n, a_1, \ldots, a_{k-1} \in \mathbb{Z}_p$. Finally compute a one-time signature σ by running $\mathsf{SgSign}(sk, \langle (upk_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}} \rangle)$ and output $C = (vk, (upk_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}}, \sigma)$.

$\mathsf{DDec}(pk, upk, usk, C)$**.** Parse C as $(vk, (upk_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}}, \sigma)$. Find $i \in [n]$ such that $upk = upk_i$ and output $(upk, \perp)$ if no such i exists. Otherwise proceed as follows. Firstly verify that the one-time signature σ is valid, the Groth-Sahai proof π^{zk} is valid, and for all $i \in [n]$ the equations $e(u_i, c_{i,3}) = e(c_{i,1}, g^{vk}U_i)$ and $e(v_i, c_{i,4}) = e(c_{i,2}, g^{vk}V_i)$ hold. If any of the above does not holds, output $(upk, \perp)$ immediately. If all of them holds, compute $\pi^{(u)} = c_{i,1}^{1/x}$, $\pi^{(v)} = c_{i,2}^{1/y}$, and $\hat{m} = c_{i,5}/\pi^{(u)}\pi^{(v)}$, and outputs $\mu = (upk, (\hat{m}, \pi^{(u)}, \pi^{(v)}))$.

DVerify(pk, upk, C, μ). Parse C as $(vk, (upk_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}}, \sigma)$. Then verify the following conditions: no $i \in [n]$ does not satisfy $upk = upk_i$, the one-time signature σ is invalid, the Groth-Sahai proof π^{zk} is invalid, or for some $i \in [n]$ one of $e(u_i, c_{i,3}) = e(c_{i,1}, g^{vk}U_i)$ and $e(v_i, c_{i,4}) = e(c_{i,2}, g^{vk}V_i)$ does not hold. If at least one of the above does not hold and μ is parsed as $(upk, \perp)$, output $\top_{\mathrm{valid}}$. Otherwise, if all of the above do hold, μ is parsed as $(upk, (\hat{m}, \pi^{(u)}, \pi^{(v)}))$, and the three equations $e(u, \pi^{(u)}) = e(c_{i,1}, g)$, $e(v, \pi^{(v)}) = e(c_{i,2}, g)$, and $c_{i,5} = \hat{m}\pi^{(u)}\pi^{(v)}$ hold, output $\top_{\mathrm{valid}}$. In any other cases, output $\perp$.

DCombine$(pk, C, \mu_1, \ldots, \mu_k)$. Parse C as $(vk, (upk_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}}, \sigma)$. If there is at least one μ_i that is parsed as $(upk, \perp)$, output $\perp$. Otherwise, parse μ_i as $(\hat{upk}_i, (\hat{m}_i, \hat{\pi}_i^{(u)}, \hat{\pi}_i^{(v)}))$, find t_i satisfying $upk_{t_i} = \hat{upk}_i$ for all $i \in [k]$, compute $m = \hat{m}_1{}^{\lambda_1} \cdots \hat{m}_k{}^{\lambda_k}$ in which $\lambda_i = \prod_{j\in[k]\setminus\{i\}} -t_j/(t_i - t_j)$, and output m.

Security. This scheme is proven to be secure under the decision linear assumption.

Theorem 5.13. *The construction is chosen-ciphertext secure if the decision linear assumption holds on $\mathcal{G}$.*

Chosen-ciphertext security of this scheme is in fact relatively easily derived from the zero-knowledge property of the Groth-Sahai proof system and the labeled PKENO scheme presented in Sect. 5.6.2, as the proposed scheme is built on these two building blocks. However, for completeness we present a bit detailed proof for the chosen-ciphertext security.

Proof (of Theorem 5.13). Let $\mathcal{A}$ be an adversary having the advantage ε in the game of the chosen-ciphertext security. To bound the advantage ε we consider the following sequence of games:

Game 0. This is the original game defined in the definition of chosen-ciphertext security.

Game 1. In this game the common reference string set in the public parameter pk is changed to be generated by the simulation algorithm.

Game 2. In this game the Groth-Sahai proof included in the challenge ciphertext is replaced with the simulated proof.

Game 3. In this game the decryption oracle is changed to reject any query that reuse the verification key vk of the one-time signature scheme from the challenge ciphertext.

Game 4. In this game, all ciphertexts of the uncorrupted public keys in the challenge ciphertext are replaced with ciphertexts that encrypt random group elements.

In the following we denote by S_i the event in which the adversary correctly guess the bit flipped by the challenger in Game i.

Lemma 5.14. *If the decision linear assumption on $\mathcal{G}$ holds,* $|\Pr[S_0] - \Pr[S_1]|$ *and* $|\Pr[S_1] - \Pr[S_2]|$ *are negligible.*

Proof. The change introduced by Game 1 will not affect the behavior of the adversary $\mathcal{A}$, since the soundness string and the witness-indistinguishable string of the Groth-Sahai proof is computationally indistinguishable, which itself proved from the decision linear assumption on $\mathcal{G}$. The change by Game 2 is also indistinguishable, since on a witness-indistinguishable string a real proof and a simulated proof (with the trapdoor behind the string) has the same distribution, thus the change by Game 2 is perfectly indistinguishable. $\square$

Lemma 5.15. *If the one-time signature scheme is strongly unforgeable,* $|\Pr[S_2] - \Pr[S_3]|$ *is negligible.*

Proof. The proof will be done by applying the difference lemma. The game differs when the adversary queries a ciphertext that includes the same verification key to the challenge ciphertext, and is not responded with $\perp$ by the original decryption oracle. In that case the query contains a valid signature on the queried ciphertext. Furthermore, whenever the adversary issues such a query, from this query we can extract a strong forgery of the one-time signature scheme, which enable us to construct a simulator that attacks the strong unforgeability of the one-time signature scheme. $\square$

Lemma 5.16. *If the decision linear assumption on $\mathcal{G}$ holds,* $|\Pr[S_3] - \Pr[S_4]|$ *is negligible.*

Proof. The proof basically follows the proof of the Dodis-Katz transformation. We first introduce the subgames $G_{3,0}, \ldots, G_{3,N}$ of the following:

Game $G_{3,\ell}$. In this game, if the challenge ciphertext includes a PKENO ciphertexts of the keys $upk_1, \ldots, upk_\ell$ (the public keys $upk_1, \ldots, upk_N$ are the keys given to the adversary at the first phase of the game), these PKENO ciphertexts are replaced with ciphertexts of random group elements.

We also note that the game $G_{3,0}$ is identical to Game 3, and the game $G_{3,N}$ is identical to Game 4, and thus it is sufficient for proving the lemma to prove that $|\Pr[S_{3,\ell-1}] - \Pr[S_{3,\ell}]|$ is negligible.

To prove this inequality we construct an algorithm $\mathcal{B}$ that attacks the chosen-ciphertext security of the DLIN-based PKENO scheme presented in Sect. 5.6.2. We briefly describe this algorithm.

Setup. The algorithm $\mathcal{B}$ firstly generates verification/signing keys vk^* and sk^* of the one-time signature scheme, and sends the verification key vk^* as the target label to the PKENO challenger. Then $\mathcal{B}$ receives a public key (u, v, U, V) of the PKENO scheme, and uses this key as upk_ℓ. All the other public keys upk_i $(i \neq \ell)$ and $\tilde{upk}_i$ and the witness-indistinguishable common reference string $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$, together with the trapdoor for simulating the Groth-Sahai proof, are generated by $\mathcal{B}$ itself.

Query I. Decryption queries (upk', C) are responded as follows. Let $C = (vk, (upk'_i)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5})_{i\in[n]}, \pi^{\mathrm{zk}}, \sigma)$. If $vk = vk^*$, $\mathcal{B}$ returns $(upk', \perp)$ to $\mathcal{A}$. This is a valid simulation due to the change introduced in Game 3. Otherwise, if $upk' \neq upk_\ell$, $\mathcal{B}$ responds to the query by using the decryption key corresponding to upk, which is known to $\mathcal{B}$. If $upk' = upk_\ell$, $\mathcal{B}$ proceeds as follows for responding to the query. Firstly $\mathcal{B}$ verifies the one-time signature σ, the zero-knowledge proof π^{zk}. In addition $\mathcal{B}$ also verifies validity of all n ciphertext $(c_{1,1}, \ldots, c_{1,5})$, …, $(c_{n,1}, \ldots, c_{n,5})$ by checking the equations $e(c_{i,1}, g^{vk}U_i) = e(u_i, c_{i,3})$ and $e(c_{i,2}, g^{vk}V_i) = e(v_i, c_{i,4})$ hold for all $i \in [n]$. If any of the above verification fails, $\mathcal{B}$ immediately returns $(upk_\ell, \perp)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sends the ciphertext $(c_{i,1}, \ldots, c_{i,5})$ where i satisfies $upk'_i = upk_\ell$ together with the tag vk to the PKENO challenger and obtains the decryption result $\hat{m}$ and a proof $(\pi^{(u)}, \pi^{(v)})$. Notice that in this query $\mathcal{B}$ certainly obtains the decryption result and the proof rather than $\perp$, because (i) $vk \neq vk^*$, thus the query is not forbidden, and (ii) the ciphertext $(c_{i,1}, \ldots, c_{i,5})$ is valid in the sense that it satisfies $e(c_{i,1}, g^{vk}U_i) = e(u_i, c_{i,3})$ and $e(c_{i,2}, g^{vk}V_i) = e(v_i, c_{i,4})$, and thus it will not rejected due to the invalidity of the ciphertext. After receiving $\hat{m}$, $\pi^{(u)}$, and $\pi^{(v)}$, $\mathcal{B}$ sends $(upk_\ell, (\hat{m}, \pi^{(u)}, \pi^{(v)})$ to $\mathcal{A}$.

Challenge. When $\mathcal{A}$ requests the challenge ciphertext by issuing $(\mathcal{S}, k, m_0, m_1)$ where $S = \{ upk_1^*, \ldots, upk_n^* \}$, $\mathcal{B}$ proceeds as follows. If for some i^*, $upk_{i^*}^* = upk_\ell$, then $\mathcal{B}$ chooses a random bit b and random integers $a_1, \ldots, a_{k-1} \in \mathbb{Z}_p$, and sends $M_0 = m_b g^{i^* a_1 + (i^*)^2 a_2 + \cdots + (i^*)^{k-1} a_{k-1}}$ and $M_1 = R_\ell$ to the PKENO challenger to receive a PKENO ciphertext $(c_1^*, \ldots, c_5^*)$. Then $\mathcal{B}$ computes PKENO ciphertext $(c_{i,1}^*, \ldots, c_{i,5}^*)$ for all $i \in [n]$ as

$$(c_{i,1}^*, \ldots, c_{i,5}^*) \leftarrow \begin{cases} (u_i^{r_i}, v_i^{s_i}, (g^{vk}U_i)^{r_i}, (g^{vk}V_i)^{s_i}, g^{r_i+s_i} m_b g^{ia_1+\cdots+i^{k-1}a_{k-1}}) \\ \qquad\qquad\qquad\qquad \text{if } upk_i^* \in \{ \tilde{upk}_1, \ldots, \tilde{upk}_{\tilde{N}} \} \\ (u_i^{r_i}, v_i^{s_i}, (g^{vk}U_i)^{r_i}, (g^{vk}V_i)^{s_i}, g^{r_i+s_i} R_i) \\ \qquad\qquad\qquad\qquad \text{if } upk_i^* \in \{ upk_1, \ldots, upk_{\ell-1} \} \\ (c_1^*, \ldots, c_5^*) \qquad\qquad \text{if } upk_i^* = upk_\ell \\ (u_i^{r_i}, v_i^{s_i}, (g^{vk}U_i)^{r_i}, (g^{vk}V_i)^{s_i}, g^{r_i+s_i} m_b g^{ia_1+\cdots+i^{k-1}a_{k-1}}) \\ \qquad\qquad\qquad\qquad \text{if } upk_i^* \in \{ upk_{\ell+1}, \ldots, upk_N \}, \end{cases}$$

in which r_i, s_i $(i \in [n] \setminus \{ i \})$ is random elements in $\mathbb{Z}_p$ and R_i $(i \in [\ell - 1])$ is random elements in $\mathbb{G}$. Furthermore, $\mathcal{B}$ computes a simulated proof π^{zk} using the trapdoor for the Groth-Sahai proof and a one-time signature σ^* using the signing key sk^*, then sends $(vk^*, (c_{i,1}, \ldots, c_{i,5})_{i \in [n]}, \pi^{\mathrm{zk}}, \sigma^*)$ as the challenge ciphertext to the adversary $\mathcal{A}$.

Query II. After receiving the challenge, $\mathcal{A}$ again issues decryption queries. These queries are responded to as in the Query I phase.

Guess. When $\mathcal{A}$ outputs a guess b' and halts, $\mathcal{B}$ outputs 1 if $b = b'$ and 0 if $b \neq b'$.

Noticing that when $\mathcal{B}$ receives the PKENO challenge ciphertext with the plaintext M_0, the view of the adversary $\mathcal{A}$ is identical to that of $G_{3,\ell-1}$, and when $\mathcal{B}$ receives the challenge of M_1, the view is identical to that of $G_{3,\ell}$. Hence $\Pr[S_{3,\ell-1}] - \Pr[S_{3,\ell}] =$

$\Pr[\mathcal{B} \rightarrow 1 | \text{the challenge is of } M_0] - \Pr[\mathcal{B} \rightarrow 1 | \text{the challenge is of } M_1]$, whose right-hand side is negligible, due to the assumption that the PKENO scheme is selective-label weak chosen-ciphertext secure. □

The above lemmas complete the proof of Theorem 5.13. □

Theorem 5.17. *The construction satisfies the strong decryption consistency.*

Proof. Let a ciphertext $C = (vk, (upk_i^*)_{i\in[n]}, k, (c_{i,1}, \ldots, c_{i,5}), \pi^{\text{zk}}, \sigma)$ and two sets of decryption shares $S = \{ (upk_1, \hat{\mu}_1), \ldots, (upk_k, \hat{\mu}_k) \}$, and $S' = \{ (upk'_1, \hat{\mu}'_1), \ldots, (upk'_k, \hat{\mu}'_k) \}$ be the output of the decryption-consistency adversary. In the following we will argue that assuming every $2k$ decryption shares are verified as $\top_{\text{valid}}$, the two sets S and S' are combined to the same result, regardless of the form the shares.

Firstly, due to the construction of the DVerify algorithm, we can assume that we only have one of the following two cases:

- $\hat{\mu}_1 = \cdots = \hat{\mu}_k = \hat{\mu}'_1 = \cdots = \hat{\mu}'_k = \bot$, or
- $\hat{\mu}_1, \ldots, \hat{\mu}_k, \hat{\mu}'_1, \ldots, \hat{\mu}'_k$ are all parsed as $(\hat{m}_1, \pi_1^{(u)}, \pi_1^{(v)}), \ldots, (\hat{m}_k, \pi_k^{(u)}, \pi_k^{(v)})$, $(\hat{m}'_1, \tau_1^{(u)}, \tau_1^{(v)})$, $\ldots$, $(\hat{m}_k, \tau_k^{(u)}, \tau_k^{(v)})$, respectively.

For the first case, both S and S' are combined to $\bot$, hence in this case the adversary never violate the decryption consistency of the scheme. For the second case, due to the perfect soundness of the Groth-Sahai proof system, there exists $r_1, \ldots, r_n, s_1, \ldots, s_n, a_1, \ldots, a_{k-1} \in \mathbb{Z}_p$, and $m \in \mathbb{G}$ that satisfies

$$c_{i,1} = u_i^{r_i},$$

$$c_{i,2} = v_i^{s_i},$$

$$c_{i,5} = g^{r_i} g^{s_i} (g^{i^{k-1}})^{a_{k-1}} \cdots (g^{i^2})^{a_2} (g^i)^{a_1} m,$$

where u_i and v_i is the part of the user public key $upk_i = (u_i, v_i, U_i, V_i)$. Furthermore, due to the committing property of the PKENO scheme, we have that for all $j \in [k]$

$$\hat{m}_j = (g^{i^{k-1}})^{a_{k-1}} \cdots (g^{i^2})^{a_2} (g^i)^{a_1} m$$

with i satisfying $upk_j = upk_i^*$, and similarly for all $j \in [k]$

$$\hat{m}'_j = (g^{i^{k-1}})^{a_{k-1}} \cdots (g^{i^2})^{a_2} (g^i)^{a_1} m$$

with i satisfying $upk'_j = upk_i^*$. These equations ensure that the two sets S and S' contain shares which are honestly derived from the common polynomial $a_{k-1}X^{k-1} + \cdots + a_1 X + \log_g m$, which further ensures that S and S' will be combined to the same decryption result. □

5.5 Generic Construction of TPKE with the Strong Decryption Consistency

In this section, we present the third proposed scheme, which achieves the strong decrytption consistency or the decrytption consistency defined by Boneh, Boyen, and Halevi [BBH06].

Let us consider the k-out-of-n setting. The basic idea behind this construction is that if k servers (but not n servers) cooperating in decryption are able to recover all the shares, it is possible to ensure the stronger decrytption consistency. This is because in this way we can avoid the following situation: even though some k servers receive apparently valid shares, that is, they lie on a common degree-$(k-1)$ polynomial, another coalition of k servers receives obviously invalid shares, namely, shares that lie on a higher-degree polynomial. This situation was unavoidable in the construction in Sect. 5.3 and had it suffer the weaker notion of decrytption consistency. However, if a construction allows a coalition of k servers to recover *all* the shares, they are able to convince themselves that any coalition other than them also recovers the same set of shares and that they will agree on the (in)validity of the shares in particular.

Obviously, if we use Shamir's secret sharing scheme (or even its variation) it is not very straightforward to achieve the two properties that (i) cooperating k servers are able to recover *all* the shares, and (ii) any coalition of $k-1$ servers gains no information on the original secret. Fortunately, such a structure of distributing shares is studied in a different context, which is also useful in our current purpose. Ito, Saito, and Nishizeki showed the possibility of realizing an arbitrary (monotone) access structure by showing that such a scheme is constructed form an t-out-of-t secret sharing scheme [ISN93]. Their scheme first generates t-out-of-t shares of the secret and assigns different subsets of t shares to different parties, in such a way that no unauthorized coalition of the parties cannot complete the t shares whereas any authorized coalition can do that.[*2]

We adopt this technique for distributing *decryption keys* among the decryption servers, instead of distributing shares. Namely, at the setup, our scheme generates t instances of the PKENO scheme and assigns different subsets of t decryption keys to different decryption servers in the exactly same manner as the Ito-Saito-Nishizeki scheme assigns t shares to the parties. The ciphertext just consists of t ciphertexts of PKENO which respectively encrypts t-out-of-t shares of the actual plaintext. In this way, any coalition of k servers is able to recover the entire shares and is able to confirm that another coalition of k servers will recover the same set of shares.

The description of the scheme is as follows. Let n and k be integers with $n \geq k \geq 1$. Let $[n] = \{1, \dots, n\}$. We denote by $\mathcal{U}$ the family of all size-$(k-1)$ subsets of $[n]$ and by $\mathcal{U}_i$ its sub-family that contains only the sets that does not contain i, that is, $\mathcal{U} = \{ U \subseteq [n] \mid |U| = k-1 \}$, and $\mathcal{U}_i = \{ U \subseteq [n] \mid |U| = k-1, i \notin U \}$.[*3]

$\mathsf{ThKg}(1^\lambda, n, k)$. Generate $\binom{n}{k-1}$ pairs of PKENO keys by running $(pk_U, dk_U) \leftarrow \mathsf{NOKg}(1^\lambda)$ for $U \in \mathcal{U}$ and set $pk = (pk_U)_{U \in \mathcal{U}}$, $vk = \emptyset$, and $sk_i = (dk_U)_{U \in \mathcal{U}_i}$.

$\mathsf{ThEnc}(pk, m)$. Run $\mathsf{SgKg}(1^\lambda)$ to obtain a pair $(vk_{\text{sots}}, sk_{\text{sots}})$ of the one-time signature scheme. Let pk be parsed as $(pk_U)_{U \in \mathcal{U}}$. Generate $\binom{n}{k-1}$-out-of-$\binom{n}{k-1}$ shares of m such that $m = \bigoplus_{U \in \mathcal{U}} m_U$ and encrypt each shares as $C_U \leftarrow \mathsf{NOEnc}(pk_U, vk_{\text{sots}}, m_U)$ for all $U \in \mathcal{U}$. Then run $\mathsf{SgSign}(sk_{\text{sots}}, \langle C_U \rangle_{U \in \mathcal{U}})$ to obtain σ_{sots}. Finally output $C = (vk_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma_{\text{sots}})$ as the ciphertext.

[*2] More formally this assignment will be done as follows. Let $\mathcal{A}$ be the family of all maximum unauthorized sets. The dealer divides the original message into $|\mathcal{A}|$ shares by $|\mathcal{A}|$-out-of-$|\mathcal{A}|$ and associates each share respectively with different unauthorized set $A \in \mathcal{A}$. Each party i receives all the shares associated to the unauthorized set that does not contain i.

[*3] For example, if $(k, n) = (2, 3)$, these sets is represented as $\mathcal{U} = \{1, 2, 3\}$, $\mathcal{U}_1 = \{2, 3\}$, $\mathcal{U}_2 = \{1, 3\}$, and $\mathcal{U}_3 = \{1, 2\}$.

$\mathsf{ThDec}(pk, i, sk_i, C)$. Let dk_i and C be parsed as $(sk_U)_{U \in \mathcal{U}_i}$ and $(vk_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma_{\text{sots}})$, respectively. If $\mathsf{SgVerify}(vk_{\text{sots}}, \langle C_U \rangle_{U \in \mathcal{U}}, \sigma_{\text{sots}}) = 0$, output $(i, \perp)$. Otherwise compute $m_U \leftarrow \mathsf{NODec}(dk_U, vk_{\text{sots}}, C_U)$ and $\pi_U \leftarrow \mathsf{NOProve}(dk_U, vk_{\text{sots}}, C_U)$ for all $U \in \mathcal{U}_i$, and then output $(i, (m_U, \pi_U)_{U \in \mathcal{U}_i})$.

$\mathsf{ThVerify}(pk, vk, C, \mu)$. Parse C as $(vk_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma_{\text{sots}})$ and μ as $(i, \hat{\mu})$. Output $\top_{\text{valid}}$ if one of the following conditions holds.

1. It holds that $\hat{\mu} = \perp$ and $\mathsf{SgVerify}(vk_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma_{\text{sots}}) = 0$, or
2. $\hat{\mu}$ is parsed as $(m_U, \pi_U)_{U \in \mathcal{U}_i}$ and $\mathsf{NOProve}(pk_U, vk_{\text{sots}}, C_U, \pi_U) = 1$ for all $U \in \mathcal{U}_i$.

Otherwise output $\perp$.

$\mathsf{ThCombine}(pk, vk, C, \mu_1, \ldots, \mu_k)$. Parse μ_j as $(i_j, \hat{\mu}_j)$ for all $j \in [n]$. Output $\perp$ if $\hat{\mu}_j = \perp$ for some j. Otherwise parse $\hat{\mu}_j$ as $\hat{\mu}_j = (m_{j,U}, \pi_{j,U})_{U \in \mathcal{U}_j}$ for all $j \in [k]$. Output $\perp$ if $m_{j,U} = \perp$ for some j and U. Output $\perp$ if $m_{j,U} \neq m_{j',U}$ for some j, j', and U. Otherwise, output $m = \bigoplus_{U \in \mathcal{U}} m_U$ in which $m_U = m_{j,U}$ with arbitrary $j \in [k]$ (the choice of j's does not affect the output).

Extension to Arbitrary Access Structures. The scheme is easily extended to support any (monotone) access structure. The extension is done by simply replacing the assignment of decryption keys to each decryption server with that described by the Ito-Saito-Nishizeki construction. The security proof is done in a quite similar way to the threshold version. Thus we omit the proof for the general access structure case but only present the proof for the threshold case.

Security. Security of the proposed scheme is described as the following theorems. We emphasize that contrary to the scheme in the previous section, the construction in this section provides the strong decrytption consistency.

Theorem 5.18. *The construction is chosen-ciphertext secure if the PKENO scheme is selective-label weakly chosen-ciphertext secure and the one-time signature scheme is strongly unforgeable.*

Proof. The proof proceeds with the following sequence of games.

Game 0. Game 0 is identical to the game for the definition of the chosen-ciphertext security of TPKE.

Game 1. In Game 1, the decryption oracle is changed to reject any decryption queries that reuse the verification key from the challenge ciphertext.

Game 2. Let U^* be the set of corrupted servers. In this game, the ciphertext C_{U^*} in the challenge ciphertext is replaced with an encryption of the all-zero string.

Let S_i be the event that in Game i the adversary successfully guess the bit flipped by the challenger. In the following lemmas we prove that the changes described above changes the probability of the successful guess by the adversary.

Lemma 5.19. *Provided that the one-time signature scheme is strongly unforgeable, we have that* $\Pr[S_0] - \Pr[S_1]$ *is negligible.*

Proof (of Lemma 5.19). The proof will be done in a similar way to that of Lemma 5.8. In this case we also employ the difference lemma with the event F defined to the case in which the adversary issues a legitimate (i.e., different from the challenge) decryption query which reuses the verification key of the one-time signature scheme from the challenge and includes a valid one-time signature. By the difference lemma we obtain that $|\Pr[S_0] - \Pr[S_1]| < \Pr[F]$. Finally due to the strong unforgeability of the one-time signature scheme we conclude $\Pr[F]$ is negligible. □

Lemma 5.20. *Provided that the labeled PKENO scheme is selective-label weakly chosen-ciphertext secure, we have that* $\Pr[S_1] - \Pr[S_2]$ *is negligible.*

Proof (of Lemma 5.20). Let $\mathcal{A}$ be an arbitrary adversary of the proposed scheme. To prove this lemma we construct a simulator $\mathcal{B}$ that interacts with $\mathcal{A}$ and tries to break the chosen-ciphertext security of the PKENO scheme. The construction of $\mathcal{B}$ is as follows.

Initialize. The simulator $\mathcal{B}$ receives the set U^* of servers to be corrupted from the adversary $\mathcal{A}$.

Setup. The simulator $\mathcal{B}$ generates keys $(vk^*_{\text{sots}}, sk^*_{\text{sots}})$ of the one-time signature scheme by running $\mathsf{SgKg}(1^\lambda)$, sends vk^*_{sots} for its own challenger as the target label, and receives a public key ek^* from the challenger. Then the simulator $\mathcal{B}$ generates $\binom{n}{k-1} - 1$ key pairs by running $(ek_U, dk_U) \leftarrow \mathsf{NOKg}(1^\lambda)$ for all $U \in \mathcal{U} \setminus \{U^*\}$. Finally, the simulator sets $ek_{U^*} \leftarrow ek^*, pk_i \leftarrow (ek_U)_{U \in \mathcal{U}_i}$ for all $i \in [n]$, $vk = \emptyset$, $sk_i = (dk_U)_{U \in \mathcal{U}_i}$ for all $i \in U^*$ and sends $pk = (ek_U)_{U \in \mathcal{U}}$, vk, and $(sk_i)_{i \in U^*}$ to the adversary $\mathcal{A}$.

Query I. When the adversary $\mathcal{A}$ issues a query (i, C) where $C = (vk_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma_{\text{sots}})$, the simulator responds as follows. If $vk_{\text{sots}} = vk^*_{\text{sots}}$, the simulator responds with $(i, \perp)$. Otherwise, for all $U \in \mathcal{U}_i$ the simulator $\mathcal{B}$ computes m_U and π_U by either running $\mathsf{NODec}(dk_U, vk_{\text{sots}}, C_U)$ and $\mathsf{NOProve}(dk_U, vk_{\text{sots}}, C_U)$ for $U \neq U^*$ or querying C_U with label vk_{sots} to the decryption-and-proof oracle for $U = U^*$. The simulator responds with $(i, (m_U, \pi_U)_{U \in \mathcal{U}_i})$.

Challenge. When the adversary $\mathcal{A}$ submits two messages m_0 and m_1, the simulator proceeds as follows. The simulator first chooses a random bit $b \in \{0,1\}$ and $\binom{n}{k-1} - 1$, also chooses random messages m_U for all $U \in \mathcal{U} \setminus \{U^*\}$, and encrypts these m_U by running $\mathsf{NOEnc}(ek_U, vk^*_{\text{sots}}, m_U)$ to obtain C_U. The simulator submits $M_0 = m_b \oplus \bigoplus_{U \in \mathcal{U} \setminus \{U^*\}} m_U$ and $M_1 = 0^{|m_b|}$ to receive the challenge ciphertext. When the simulator then receives the challenge ciphertext C^*, which is an encryption of either the correct share of m_b or garbage under the label vk^*_{sots}, the simulator sets C_{U^*} to be C^*, computes a one-time signature σ^*_{sots} of $\langle C_U \rangle_{U \in \mathcal{U}}$ by running $\mathsf{SgSign}(sk^*_{\text{sots}}, \langle C_U \rangle_{U \in \mathcal{U}})$, and sends $(vk^*_{\text{sots}}, (C_U)_{U \in \mathcal{U}}, \sigma^*_{\text{sots}})$ back to the adversary $\mathcal{A}$.

Query II. Again the adversary $\mathcal{A}$ issues decryption queries, which are responded as in the previous phase by the simulator $\mathcal{B}$.

Guess. Finally the adversary $\mathcal{A}$ outputs a bit b' and halts. The simulator outputs 1 if $b = b'$ and 0 otherwise.

In this simulation, if $\mathcal{B}$ receives the challenge ciphertext C^* of M_0 the adversary's view is identical to that of Game 1, while of M_1 that is identical to Game 2. This fact shows that $\Pr[S_1] - \Pr[S_2] = \Pr[\mathcal{B} \to 1 | b = 0] - \Pr[\mathcal{B} \to 1 | b = 1]$. It is negligible because of the assumption that the PKENO scheme is selective-label weakly chosen-ciphertext secure. □

Finally we show that in the last game the adversary has no advantage on guessing the challenger's bit.

Lemma 5.21. $\Pr[S_2] = 1/2$.

Proof (of Lemma 5.21). In Game 2, regardless of the bit generated by the challenger of the TPKE scheme, all the PKENO ciphertexts are ciphertexts of either a randomly-chosen string or the all-zero string. Thus the behavior of the challenger is independent from the bit b, and it is information-theoretically hidden from the adversary. □

These three lemmas completes the proof of the actual theorem. □

Theorem 5.22. *The construction has the strong decryption consistency if the PKENO scheme is strongly committing.*

Proof. Let $\mathcal{A}$ be an adversary that attacks the decrytption consistency property of the above generic construction. We construct a simulator $\mathcal{B}$ that breaks the committing property of the PKENO scheme by interacting with $\mathcal{A}$. The description of is as follows:

Setup. The simulator $\mathcal{B}$ receives a encryption/decryption key pair (ek, dk) of the PKENO scheme from its own challenger. Then $\mathcal{B}$ sets up the TPKE scheme as follows: $\mathcal{B}$ chooses random U^* from the family $\mathcal{U}$ and sets $ek_{U^*} \leftarrow ek$ and $dk_{U^*} \leftarrow dk$. For all $U \in \mathcal{U} \setminus \{U^*\}$, $\mathcal{B}$ runs $\mathsf{NOKg}(1^\lambda)$ to obtain (ek_U, dk_U). Then $\mathcal{B}$ sets $pk \leftarrow (ek_U)_{U\in\mathcal{U}}$, $vk \leftarrow \emptyset$, and $sk_i \leftarrow (dk_U)_{U\in\mathcal{U}_i}$ for all $i \in [n]$, and sends pk, vk, sk_1, …, sk_n to the adversary $\mathcal{A}$.

Forge. When the adversary outputs a triple (C, S, S'), the simulator $\mathcal{B}$ proceeds as follows. The simulator $\mathcal{B}$ first verifies that (C, S, S') satisfies the winning condition of the decrytption consistency game. If the condition does not satisfied, the simulator aborts. Notice that if $\mathcal{B}$ does not abort, we have that $\mathsf{ThCombine}(pk, vk, C, S) \neq \mathsf{ThCombine}(pk, vk, C, S')$, since the $\mathsf{ThVerify}$ algorithm of our construction concerned never outputs $\top_{\text{invalid}}$. Then the simulator $\mathcal{B}$ finds tuple (L, c, m, π, m', π') that breaks the committing property of the PKENO scheme as follows: (i) when both S and S' claim non-$\perp$ result, we have that $\bigoplus_{U\in\mathcal{U}} m_U \neq \bigoplus_{U\in\mathcal{U}} m'_U$. It implies that for some U we have that $m_U \neq m'_U$ with valid proofs π_U and π'_U to a single ciphertext C_U.

Let U' be the set satisfying this condition. In this case $\mathcal{B}$ outputs $(L, c, m, \pi, m', \pi') = (vk_{\mathrm{sots}}, C_{U'}, m_{U'}, \pi_{U'}, m'_{U'}, \pi'_{U'})$. (i) The other case is that only one of S and S' claims $\perp$ and the other claims a non-$\perp$ plaintext. Let S claims m as the decryption result and S' claims $\perp$. In this case, for all $U \in \mathcal{U}$ it holds that $m_U \neq \perp$ and for some $U \in \mathcal{U}$ it holds that $m'_U = \perp$. Let U' be the set such that $m'_U = \perp$. The simulator $\mathcal{B}$ outputs $(L, c, m, \pi, m', \pi') = (vk_{\mathrm{sots}}, C_{U'}, m_{U'}, \pi_{U'}, \perp, \pi'_{U'})$.

The simulator $\mathcal{B}$ successfully breaks the committing property of the PKENO scheme when the adversary $\mathcal{A}$ breaks the decrytption consistency property and $U^* = U'$. Since the distribution of U^* is independent from the view of $\mathcal{A}$, the success probability of $\mathcal{B}$ is the advantage of $\mathcal{A}$ divided by $\binom{n}{k-1}$, which is even negligible by the assumption of the committing property of the PKENO scheme. □

5.6 Instantiating PKENO Supporting Labels

In this section we present two specific instantiations of labeled PKENO for completeness. The schemes can be easily obtained by tweaking the TPKE schemes by Arita and Tsurudome [AT09]. We also note that Galindo et al. have already showed a generic construction of PKENO from TPKE with appropriate security [GLF+10] and that our instantiations are almost obtained by instantiating their generic construction with the Arita-Tsurudome schemes. Despite this we present concrete descriptions of our instantiations because our schemes explicitly treat the "label" functionality, which is not carried out by Galindo et al.

5.6.1 Instantiation from the Decision Bilinear Diffie-Hellman Assumption

The first instantiation is based on the decision bilinear Diffie-Hellman assumption. The concrete construction is as follows:

$\mathsf{NOKg}(1^\lambda)$**.** Run $\mathcal{G}(1^\lambda)$ to obtain the bilinear-groups parameter $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Choose random $x \leftarrow \mathbb{Z}_p$ and random elements $v, z \leftarrow \mathbb{G}$, sets $u = g^x$, $ek = (p, \mathbb{G}, \mathbb{G}_T, e, g, u, v, z)$ and $dk = x$. Output (ek, dk).

$\mathsf{NOEnc}(ek, L, m)$**.** Choose a random integer $r \leftarrow \mathbb{Z}_p$. Computes $c_1 \leftarrow g^r$, $c_2 \leftarrow (u^L v)^r$, and $c_3 \leftarrow e(u, z)^r m$. Output (c_1, c_2, c_3).

$\mathsf{NODec}(dk, L, c)$**.** Parse c as (c_1, c_2, c_3). Verify that the equation $e(g, c_2) = e(c_1, u^L v)$ holds, otherwise output $\perp$. If the equation holds, output $c_3 / e({c_1}^x, z)$.

$\mathsf{NOProve}(dk, L, c)$**.** Parse c as (c_1, c_2, c_3). Verify that the equation $e(g, c_2) = e(c_1, u^L v)$ holds, otherwise output $\perp$. If the equation holds, output ${c_1}^x$.

$\mathsf{NOVerify}(pk, L, m, c, \pi)$**.** Parse c as (c_1, c_2, c_3). If $m = \perp$, $\pi = \perp$, and $e(g, c_2) \neq e(c_1, u^L v)$, output 1. If $m \neq \perp$ and $\pi \neq \perp$, then output 1 when $e(g, \pi) = e(c_1, u)$, $e(g, c_2) = e(c_1, u^L v)$, and $c_3 / m = e(\pi, z)$. Otherwise, output 0.

The scheme is proven to be secure under the decision bilinear Diffie-Hellman assumption.

Theorem 5.23. *The above construction is selective-label weakly chosen-ciphertext secure provided the decision bilinear Diffie-Hellman assumption holds.*

Proof. Let $\mathcal{A}$ be the adversary which attacks the DBDH-based instantiation with advantage ϵ. We construct a simulator $\mathcal{B}$ that solves the decision bilinear Diffie-Hellman problem with the same advantage. The description of $\mathcal{B}$ is as follows.

Initialize. Given a problem instance $(g, g^\alpha, g^\beta, g^\gamma, T)$, in which T is either $e(g, g)^{\alpha\beta\gamma}$ or a random element from $\mathbb{G}_T$, $\mathcal{B}$ runs $\mathcal{A}$ with input the security parameter 1^λ and receives a label L^*.

Setup. The simulator $\mathcal{B}$ then sets up the public key $ek = (g, u, v, z)$ for the scheme, where $u \leftarrow g^\beta$, $v \leftarrow u^{-L^*} g^t$ with random $t \in \mathbb{Z}_p$, and $z \leftarrow g^\gamma$. The simulator $\mathcal{B}$ sends ek to the adversary $\mathcal{A}$.

Query I. When $\mathcal{A}$ issues a decryption-and-proof query (L, c) where $c = (c_1, c_2, c_3)$, $\mathcal{B}$ responds as follows. If the query does not satisfy the equation $e(g, c_2) = e(c_1, u^L v)$, $\mathcal{B}$ responds with $(\perp, \perp)$. Otherwise $\mathcal{B}$ computes $\pi \leftarrow (c_2 / {c_1}^t)^{1/(L-L^*)}$ and $m \leftarrow c_3 / e(\pi, z)$, and returns (m, π) to $\mathcal{A}$. The simulation is perfect, that is, $\mathcal{A}$'s view in interaction with $\mathcal{B}$ is identical to the view in interaction with the challenger in the security definition.

Actually, given that the equation $(g, c_2) = (c_1, u^L v)$ holds, we have that $c_1 = g^r$ and $c_2 = (u^L v)^r$ for some $r \in \mathbb{Z}_p$. Furthermore, since $\mathcal{A}$ is restricted to issue queries that satisfy $L \neq L^*$, from a simple calculation we also have that $\pi = u^r$. From this it follows that the simulation is perfect.

Challenge. When $\mathcal{A}$ issues a pair (m_0, m_1), $\mathcal{B}$ generates the challenge ciphertext (c_1^*, c_2^*, c_3^*) by letting $c_1^* \leftarrow g^\alpha$, $c_2^* \leftarrow (g^\alpha)^t$, and $c_3^* \leftarrow T \cdot m_b$ with random $b \leftarrow \{0, 1\}$. Then $\mathcal{B}$ returns $c = (c_1, c_2, c_3)$ to $\mathcal{A}$. The simulation is again perfect except the distribution of c_3^*, which is perfect under the condition that $T = e(g, g)^{\alpha\beta\gamma}$. This follows from corresponding α to the randomness r in the (challenge) ciphertext and noticing that the ciphertext is generated under the label L^*. We also note that if T is a random element no information on b is given to the adversary in a information-theoretical sense.

Query II. In this phase $\mathcal{A}$ again issues decryption-and-proof queries, to which $\mathcal{B}$ responds as in the previous phase.

Guess. Finally the adversary $\mathcal{A}$ outputs a guess b'. The simulator $\mathcal{B}$ outputs 1 if $b = b'$ and outputs 0 otherwise.

The above description shows that if $T = e(g, g)^{\alpha\beta\gamma}$, $\mathcal{B}$ outputs 1 with probability $1/2 + \epsilon$, whereas if T is a random element outputs 1 with probability $1/2$. This shows that the advantage of $\mathcal{B}$ in solving the DBDH problem is $(1/2 + \epsilon) - 1/2 = \epsilon$, which completes the proof. □

Theorem 5.24. *The DBDH-based construction is strongly committing.*

Proof. Given two message-proof pairs (m, π) and (m', π') for a ciphertext $c = (c_1, c_2, c_3)$ and a common label L, we argue that if two proofs π and π' are both valid, two messages are equal (with probability 1). Firstly, the case that only one of m and m' is $\perp$ and the other is not, will never occur. This is simply because that the proof for $\perp$ implies that the equation $e(g, c_2) = e(c_1, u^L v)$ dose *not* hold and the proof for non-$\perp$ implies that the equation *does* hold. Hence we assume that neither m nor m' are $\perp$. Since the equations $e(g, \pi) = e(c_1, u) = e(g, \pi')$, which are verified to hold in the NOVerify algorithm of the construction, we have that $\pi = \pi'$

due to the non-degenerate (and bijective when one argument is fixed) property of the bilinear map e. It finally implies that $m = c_3/e(\pi, z) = c_3/e(\pi', z) = m'$, in which the first and third equations come from the equations verified in the NOVerify algorithm, and the second from the fact $\pi = \pi'$. □

5.6.2 Instantiation from the Decision Linear Assumption

The other instantiation is based on the decision linear assumption. The construction is as follows:

NOKg(1^λ)**.** Run $\mathcal{G}(1^\lambda)$ to obtain the bilinear-groups parameter $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Choose random x, $y \leftarrow \mathbb{Z}_p$ and random elements U, $V \leftarrow \mathbb{G}$, sets $u = g^x$, $v = g^y$, $ek = (p, \mathbb{G}, \mathbb{G}_T, e, u, v, U, V)$ and $dk = (x, y)$. Output (ek, dk).

NOEnc(ek, L, m)**.** Choose a random integer r, $s \leftarrow \mathbb{Z}_p$. Computes $c_1 \leftarrow u^r$, $c_2 \leftarrow v^s$, $c_3 \leftarrow (g^L U)^r$, $c_4 \leftarrow (g^L V)^s$, and $c_5 \leftarrow g^{r+s}m$. Output $(c_1, c_2, c_3, c_4, c_5)$.

NODec(dk, L, c)**.** Parse c as $(c_1, c_2, c_3, c_4, c_5)$. Verify the equation $e(c_3, u) = e(g^L U, c_1)$ and $e(c_4, v) = e(g^L V, c_2)$ hold, otherwise output $\perp$. If the equation holds, output $c_5/{c_1}^{1/x}{c_2}^{1/y}$.

NOProve(dk, L, c)**.** Parse c as $(c_1, c_2, c_3, c_4, c_5)$. Verify that the equation $e(c_3, u) = e(g^L U, c_1)$ and $e(c_4, v) = e(g^L V, c_2)$ hold, otherwise output $\perp$. If the equation holds, output $\pi = (\pi^{(u)}, \pi^{(v)}) = ({c_1}^{1/x}, {c_2}^{1/y})$.

NOVerify(pk, L, m, c, π)**.** Parse c as $(c_1, c_2, c_3, c_4, c_5)$. If $m = \perp$, $\pi = \perp$, and either $e(c_3, u) \neq e(g^L U, c_1)$ or $e(c_4, v) \neq e(g^L V, c_2)$ hold, output 1. If $m \neq \perp$ and π is parsed as $(\pi^{(u)}, \pi^{(v)})$, then output 1 when $e(c_1, g) = e(u, \pi^{(u)})$, $e(c_2, g) = e(v, \pi^{(v)})$, $e(c_3, u) = e(g^L U, c_1)$, $e(c_4, v) = e(g^L V, c_2)$, and $c_5/m = \pi^{(u)}\pi^{(v)}$. Otherwise output 0.

Security proofs can be done in a similar way to the DBDH-based construction. Since the security proof for this scheme will be done almost similarly to the DBDH-based scheme, we omit the formal proof for the DLIN-based instantiation.

Theorem 5.25. *The DLIN-based construction is selective-label weakly chosen-ciphertext* *secure provided the decision linear assumption holds.*

Theorem 5.26. *The DLIN-based construction is strongly committing.*

5.7 Conclusion

We presented three constructions of TPKE with decryption consistency. The first and second scheme is in fact dynamic TPKE. The first scheme is actually a generic construction from PKENO with the weak decryption consistency. The second scheme deviates from a generic construction, while providing a shorter ciphertext length than the first scheme and the strong decryption consistency. These two schemes are the first dynamic TPKE with the (weak or strong) decryption consistency which do not rely on neither q-type assumptions or random oracles. The third scheme puts forward the possibility of a generic construction of TPKE with the strong decryption consistency, and show that it is possibly at cost of being a non-dynamic scheme and allowing only a smaller (logarithmic) number of decryption servers.

Part III

Applying Group Signature to a New Extension of Public-key Encryption

Chapter 6

Restrictive Public-key Encryption

In this chapter, we present a new extension of public-key encryption, which we name restrictive public-key encryption scheme. This extension has several useful application, to name a few, multiparty computation secure against malicious adversary.

This contribution is categorized as the type (II) in the Sect. 1.1.3. The notion of restrictive PKE is obtained by translating the *revocation* extension[*1] (and its related security notion) of group signature to those of public-key encryption, via the generic construction of (ordinary) public-key encryption from group signature.

6.1 Introduction

6.1.1 Background and Motivation

Public key encryption schemes are required to hide even partial information of plaintexts. This strong requirement is formalized as the notion of semantic security [GM84], and it is currently considered as even one of the lowest requirement for encryption scheme.

As a consequence of the strong secrecy requirement of semantic security, no one can detect the ciphertext which encrypts some particular plaintexts. This paper considers how to add

[*1] The revocation extension of group signature allows the group manager to revoke the membership of group member securely. Once the membership of a group member is revoked, the member no longer able to generate a signature that verified as valid by the verification algorithm.

such a functionality to public key encryption without losing reasonable secrecy of encrypted plaintexts. To formally treat this functionality, we define the notion of *restrictive public key encryption (RPKE)*. RPKE allows a trusted third party to specify a set of *prohibited messages*, and anyone can detect a ciphertext which encrypts one of the prohibited messages. Moreover, this verification process is required not to leak information about the encrypted plaintext except whether it is a prohibited messages or not.

Such a functionality can be realized using well-known NIZK technique, like a general NIZK through the Hamilton path problem. In this paper, we explore further efficient construction, and proposes a scheme that achieves shorter ciphertext whose length does not increase as the number of allowed messages increase.

One of the application of RPKE is a countermeasure against abuse of a public key infrastructure by terrorists. This is achieved by disallowing encryption of crime-related messages, and forbid terrorists from using a public key infrastructure to planning terrorism or sending instruction for terrorist activities. Another application may be a format-checking in electronic voting, by disallowing encryption of irregular format ballots. In this application, only encryptions of correctly-formatted voting is allowed, and gateways can dispose any encrypted ballot of irregular format without violating privacy of voters. Parental control and SPAM-mail filtering are further potential application of RPKE. Restrictive PKE would be also useful in multiparty computation in which parties sends ciphertexts whose plaintexts should have some specific formats. In such a case a malicious participants sends encryption of irregular-format plaintexts which will cause several security issues. A RPKE scheme allows honest parties to detect such irregular ciphertexts.

6.1.2 Contributions

In this paper, we give a formal definition of RPKE. We also give efficient constructions of RPKE. The definition even captures very strong security of chosen-ciphertext security. The construction utilizes the techniques of Teranishi et al. [TFS09], Boudot [Bou00], and Nakanishi et al. [NFHF10], which techniques are all developed in the context of group signature with revocation, in order to obtain an efficient construction. The construction also has a capability of updating the message space specified by the authority. We again emphasize that the construction given in this paper is quite more efficient than the trivial construction employing the general NIZK technique through the Hamilton path problem and even more efficient than a simple OR-proof based construction. More concretely, the encryption cost, the verification cost, and the ciphertext length is constant (independent from the number of allowed messages and the number of prohibited messages), whereas in the OR-proof construction they all linearly increase as allowed messages increase. This efficiency is achieved by the use of techniques of Teranishi et al. [TFS09], Boudot [Bou00], and Nakanishi et al. [NFHF10]. The proposed construction uses the BB signature [BB08] and the BBS+ signature [ASM06, BBS04, FI06] and further uses non-interactive proofs proving possession of the signature. This non-interactive proofs are constructed from novel algebraic properties of the BB signature and the BBS+ signature. Furthermore, we also briefly discuss the OR-proof construction as an alternative construction suitable for the case of small number of the permitted messages.

6.1.3 Related Work

Verifiable encryption [CD00, CS03a, TV09] is one of the most widely known ways to restrict contents under the secret channel and enables anyone to verify whether encrypted messages satisfy certain restrictions or not without leaking other information about plaintexts. However, verifiable encryption does not have the capability of disallowing to encrypt some specified messages. Fuchsbauer and Pointcheval [FP09] proposed a techniques to verify whether an encrypted plaintext satisfies some pairing-product equation, but it also lacks a capability of restricting the message space. Searchable encryption [BDCOP04] seems to be a promising technique to construct RPKE, that is, once the authority publicizes trapdoor information corresponding to some prohibited keyword, anyone can detect ciphertexts that encrypt one of the prohibited keyword. However, in order to update the message space publicized by the authority, this approach requires to revoke the trapdoor previously publicized. Since all known searchable encryption schemes does not have such a capability, searchable encryption cannot be directly adopted to the RPKE context. Another approach is publicizing all trapdoors for allowed messages, in order to recover encrypted message by using this trapdoor information and detecting prohibited messages. However, this approach is also inappropriate, because the information of encrypted message is completely leaked due to the trapdoor information publicized by the authority. González Nieto et al. [GNMP+12] proposed a primitive called publicly verifiable ciphertexts [GNMP+12], in which the consistency check of chosen-ciphertext secure encryption can be outsourced from the receiver, and a ciphertext can be converted into another form which still guarantees chosen-plaintext security.

There are several recent development on the techniques for efficient NIZK proof applicable to broad classes of languages (such as the Groth-Sahai proofs [GS08]). These techniques can achieve quite high security level (as its security can be proved without relying on random oracles from standard assumption), without loosing fairly practical efficiency. However, these techniques do not achieve really practical performance comparing to the techniques developed in the random oracle model such as the Fiat-Shamir transformation.

6.2 Restrictive Public Key Encryption

6.2.1 Motivating Discussion

First, to control the contents under secure communications, we consider a scenario as follows. Let us consider four entities: a message restriction authority (MRA), a verifier, a sender, and a receiver. The MRA indicates a restricted message space MS (a set of allowed messages), and publicizes the corresponding public verification key to verifiers and senders. A verifier (which is assumed to be a gateway) inspects whether a ciphertext sent by the sender is an encryption of a value belonging to the message space MS. We require that any information about a plaintext is not revealed from a ciphertext, except the above information. In addition, a verifier inspects all ciphertext, and disposes it if it does not pass the verification process. In this scenario, the MRA can control the encrypted contents without compromising user privacy.

6.2.2 Formal Definitions

Definition 6.1. A restrictive public key encryption (RPKE) consists of six algorithms (MRASetup, RKeyGen, MSSetup, REnc, VerifyMS, RDec) such that:

MRASetup: The key generation algorithm for the MRA takes as input a security parameter $\kappa \in \mathbb{N}$, and returns a public key pk_{MRA} and a private key sk_{MRA}.

RKeyGen: The receiver key generation algorithm takes as input pk_{MRA}, and returns a public key pk_{dec} and a private key sk_{dec}.

MSSetup: The public verification key generation algorithm takes as inputs pk_{MRA}, sk_{MRA}, and MS, and returns the public verification key pk_{MS}.

REnc: The encryption algorithm takes as inputs pk_{MRA}, pk_{dec}, MS, pk_{MS}, and a message M, and returns a ciphertext C. If $M \notin MS$, then the algorithm returns $\perp$.

VerifyMS: The public verification algorithm takes as inputs pk_{MRA}, pk_d, MS, pk_{MS}, and C, and returns a bit 1 or 0.

RDec: The decryption algorithm takes as inputs pk_{MRA}, pk_d, sk_d, MS, pk_{MS}, and C, and returns M or $\perp$.

As a correctness, a RPKE scheme has to satisfy that for any $\kappa \in \mathbb{N}$, any restrictive message space MS, any message $M \in MS$, $(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}) \leftarrow \mathsf{MRASetup}(1^\kappa)$, $(pk_d, sk_d) \leftarrow \mathsf{RKeyGen}(pk_{\mathrm{MRA}})$, $pk_{MS} \leftarrow \mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS)$, and $C \leftarrow \mathsf{REnc}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, M)$, it holds that $\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, MS, pk_{MS}, C) = M$ and $\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, C) = 1$.

Here, we describe an implementation of a scenario to control the contents under secure communications (presented in Sect. 6.2.1) using RPKE notations. The MRA runs $\mathsf{MRASetup}(1^\kappa)$, publicizes its public key pk_{MRA}, and keeps its secret key sk_{MRA}. The MRA indicates an allowed message space MS, runs $\mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS)$, and publicizes pk_{MS}. A receiver runs $\mathsf{RKeyGen}(pk_{\mathrm{MRA}})$ and publicizes its public key pk_{dec}, and keeps its corresponding secret key sk_{dec}. For a plaintext M, a sender computes a ciphertext C

by running $\mathsf{REnc}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, M)$, and sends C to a verifier (which is assumed to be gateway). By using only public values pk_{MRA}, pk_{dec}, and pk_{MS}, a verifier checks whether $M \in MS$ or not *without decrypting* C. In addition, this procedure should be done without any interaction with other entities. If $\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS^*}, C) = 1$ (i.e., $M \in MS$), then the verifier forwards C to the corresponding receiver. Otherwise, if $\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS^*}, C) = 0$ (i.e., $M \notin MS$ or C is an ill-formed value), the verifier disposes C. The receiver runs $\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, MS, pk_{MS}, C)$, and obtains M. The receiver does not have to consider MS for decrypting C. In the above scenario, The MRA and the verifier cannot obtain any information about a plaintext M from a ciphertext, except whether $M \in MS$ or not. If MS is changed updated to MS', then the MRA runs $\mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS')$, and publicizes $pk_{MS'}$ again. And then $pk_{MS'}$ is broadcasted to all users.

Here we define verification soundness, which requires that all dishonestly-generated ciphertext never passes the verification process of $\mathsf{VerifyMS}$. Furthermore, this notion requires even a ciphertext which is honestly-generated with MS not to pass the verification process with a different message space MS'. The latter prevents a sender from reusing a previous public verification key pk_{MS}. To guarantee that even a receiver cannot produce such a invalid (dishonestly-generated but passing the verification) ciphertext, we allow $\mathcal{A}$ to obtain (pk_d, sk_d).

Definition 6.2. A RPKE is said to satisfy verification soundness if the advantage

$$\begin{aligned}
&\Pr[(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}) \leftarrow \mathsf{MRASetup}(1^\kappa);\\
&(pk_{\mathrm{dec}}, sk_{\mathrm{dec}}) \leftarrow \mathsf{RKeyGen}(pk_{\mathrm{MRA}});\\
&(MS^*, pk_{MS^*}, C^*) \leftarrow \mathcal{A}^{\mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, \cdot)}(pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, sk_{\mathrm{dec}});\\
&: \mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, MS^*, pk_{MS^*}, C^*) = 1\\
&\wedge \mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, MS^*, pk_{MS^*}, C^*) \notin MS^*\\
&\qquad\qquad \wedge\ pk_{MS^*} \text{ is received from } \mathsf{MSSetup} \text{ oracle by query } MS^*]
\end{aligned}$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we define indistinguishability with restrictive message space under chosen ciphertext attack (IND-MSR-CCA). To guarantee that even the MRA cannot decrypt a ciphertext, we assume that $\mathcal{A}$ can obtain $(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}})$.

Definition 6.3. A RPKE is said to satisfy IND-MSR-CCA if the advantage

$$\begin{aligned}
&\Pr[(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}) \leftarrow \mathsf{MRASetup}(1^\kappa);\\
&(pk_{\mathrm{dec}}, sk_{\mathrm{dec}}) \leftarrow \mathsf{RKeyGen}(pk_{\mathrm{MRA}});\\
&(M_0^*, M_1^*, MS^*) \leftarrow \mathcal{A}^{\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, sk_{\mathrm{dec}}, \cdot,\cdot,\cdot)}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, pk_d);\\
&b \leftarrow \{0,1\}; pk_{MS^*} \leftarrow \mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS^*);\\
&C^* \leftarrow \mathsf{REnc}(pk_{\mathrm{MRA}}, pk_d, MS^*, pk_{MS^*}, M_b);\\
&\qquad\qquad b' \leftarrow \mathcal{A}^{\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, sk_{\mathrm{dec}}, \cdot,\cdot,\cdot)}(s, C^*) : b = b'] - 1/2
\end{aligned}$$

is negligible for any PPT adversary $\mathcal{A}$ which satisfies the following conditions: (1) The adversary $\mathcal{A}$ does not query the decryption oracle $\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, \cdot, \cdot, \cdot)$ with the query (MS^*, pk_{MS^*}, C^*) after receiving the challenge ciphertext C^* and (2) M_0^*, M_1^*, and MS^* output by the adversary $\mathcal{A}$ always satisfy that $M_0^*, M_1^* \in MS^*$.

6.3 Useful Techniques

In this section, we explain the techniques used in the main scheme presented in Sect. 6.4, which are Teranishi et al., Boudot, and Nakanishi et al. [Bou00, NFHF10, TFS09]. For ease of understanding at first we explain the proposed scheme in a somewhat abstract manner, and then we describe the scheme in detail.

Let $[1, N] = \{1, \ldots, N\}$ be a set of all possible messages (may or may not be prohibited) and r be the number of all prohibited messages. We say that the sequence $(m_1, \ldots, m_r)$ is the consecutive prohibited messages of MS when $\{m_1, \ldots, m_r\}$ is the all prohibited messages of MS and it holds that $m_1 < \cdots < m_r$. Later $(m_1, \ldots, m_r)$ denotes the consecutive prohibited messages of MS, where MS is the allowed message space implicit in the context.

From the highest perspective, the proposed construction is to encrypt a plaintext M by computing $c = \mathsf{PEnc}(pk_d, M; u)$ and adding a non-interactive proof π that proves that $M \in MS$, and constitute a whole ciphertext (c, π) as $c = \mathsf{PEnc}(pk_d, M; u)$ and

$$\pi = \mathit{NIZK}\{\,(M, u) : c = \mathsf{PEnc}(pk_d, M; u) \wedge M \in MS\,\}.$$

For example, π is a OR-proof (through Fiat-Shamir heuristics) as $\mathit{NIZK}\{M : (M = M_1) \vee \cdots \vee (M = M_{N-r})\}$, however, the efficiency might linearly depend on the number of allowed messages $N - r$. Or, when using inequality proof with OR-proof to construct a proof π, the efficiency increases linearly depends on the number of prohibited messages r. These construction does not provide satisfiable efficiency. To improve the efficiency of the construction, we employ a technique developed by Nakanishi et al. [NFHF10]. Before explaining the Nakanishi et al. technique, we explain two other technique developed by Teranishi et al. [TFS09] and Boudot [Bou00], which are used as building blocks in the Nakanishi et al. technique.

6.3.1 Teranishi et al. Technique [TFS09]

Using the technique of Teranishi, Furukawa, and Sako [TFS09], we can reduce the computational complexity just mentioned above. Briefly speaking, Teranishi et al. technique is an NIZK proof of knowledge that proves a secret knowledge ω is in the interval $[1, N]$. This technique involves a signature scheme $(\mathsf{SgKg}, \mathsf{SgSign}, \mathsf{SgVerify})$, and its NIZK proof has the form of $\mathit{NIZK}\{(S, \omega) : \mathsf{SgVerify}(\omega, S) = 1\}$ (where the witness is (S, ω)). This proof system can be efficiently constructed by using an appropriate signature scheme (the BB signature [BB08] is used indeed) and its algebraic property.

When applying this technique to the RPKE construction, we get the following improvement: In the setup, the MRA generates a verification/signing key pair and secretly possesses the signing key. The MRA then publicizes signatures for all allowed messages. To encrypt a message M, a sender uses the signature S publicized by the MRA and compute a ciphertext $c = \mathsf{PEnc}(pk_d, M; u)$ and a proof of knowledge

$$\pi = \mathit{NIZK}\{\,(M, u, S) : c = \mathsf{PEnc}(pk_d, M; u) \wedge \mathsf{SgVerify}(M, S) = 1\,\}, \quad (6.1)$$

which is attached to the ciphertext c. In this way, if a sender wants to encrypt an allowed message $M \in MS$, he can generate an acceptable π using $\mathsf{SgSign}(M)$ publicized by the MRA. In contrast, if a sender wants to encrypt a prohibited message $M' \notin MS$, he cannot generate an acceptable proof π of the form above. This is because the MRA does not publicize $\mathsf{SgSign}(M')$, nor the sender cannot generate $\mathsf{SgSign}(M')$ by himself (due to the unforgeability of the signature), and thus the sender cannot generate the proof of knowledge $NIZK\{(M', S) : \mathsf{SgVerify}(M', S) = 1\}$ due to the lack of the knowledge needed.

Furthermore, the computational cost of verification of the proof does not depend on the number of allowed messages, because the proof of knowledge $NIZK\{(M, u, S) : c = \mathsf{PEnc}(pk_d, M; u) \wedge \mathsf{SgVerify}(M, S) = 1\}$ used here does not depend on the number of allowed messages.

6.3.2 Boudot Technique [Bou00]

The construction discussed above requires the MRA to publicize $|MS|$ signatures. Motivated to reduce this large size of the public parameter, we then introduce Boudot's technique [Bou00]. Although the technique itself is not directly applied to our context of the set-membership proof, it was shown to be applicable to this context by Nakanishi et al. [NFHF10].

The technique is proving a relationship between hidden integers. The relationship proved by this technique is the form $\omega = {\omega_1}^2 + \omega_2$ in which ω, ω_1, and ω_2 are integers hidden as witness. This technique can potentially reduce the size of the public parameter of our construction.

The point is that when an arbitrary integer $\omega \in [1, N]$ is expressed as $\omega = \omega_1 + \omega_2$, ω_1 and ω_2 run the smaller ranges of $[1, N_1] = \lfloor \sqrt{N} \rfloor$ and $[1, N_2] = \lfloor 2\sqrt{N} \rfloor$ respectively. Utilizing this fact, if we want to prove that an encrypted plaintext is in the range $[1, N]$ of this specific form, we can directly apply the Boudot technique. Instead of publicizing N signatures of messages $1, \ldots, N$ (as in the Teranishi et al. technique), the authority publicizes signatures on messages $1, \ldots, N_1 = \lfloor \sqrt{N} \rfloor$ and signature on $1, \ldots, N_2 = \lfloor 2\sqrt{N} \rfloor$ (by a different signing key). A sender who wants to prove his ciphertext encrypts a message ω is in the range $[1, N]$ can construct an non-interactive proof as follows: First he decomposes the message ω as $\omega = {\omega_1}^2 + \omega_2$ in which $\omega_1 \in [1, N_1]$ and $\omega_2 \in [1, N_2]$. Then prove the knowledge of the signatures on ω_1 and ω_2 and simultaneously proves the relationship $\omega = {\omega_1}^2 + \omega_2$ between the two signed message and the encrypted message.

This technique can only prove membership of a single interval $[a, b]$, but it is not able to prove membership of union of several intervals $[a_1, b_1] \cup [a_2, b_2] \cup \cdots \cup [a_n, b_n]$. The latter type of proof system can be used for our purpose of constructing RPKE. Teranishi et al. provided a technique for the set-membership proof of this type, which is explained in the next section.

6.3.3 Nakanishi et al. Technique [NFHF10]

The proposed scheme is obtained by combining Nakanishi et al.'s technique with the Boudot techniques. Nakanishi et al.'s technique utilizes the fact that if all the prohibited messages are denoted as m_1, ..., m_r, and $m_1 < \cdots < m_r$ holds, then any allowed message $M \in MS$ has a unique "position" j such that $m_j < M < m_{j+1}$ holds, and any prohibited message $M \notin MS$ has no such position. Another fact that the technique relies on is that when $N < p/2$ holds, $y > x$ is logically equivalent to $y - x \mod p \in [1, N]$ for any x, $y \in [1, N]$. Using these properties, one can prove the fact $M \in MS$ by proving the existence of j such that $m_j < M < m_{j+1}$ instead, and prove $m_j < M < m_{j+1}$ itself by proving $M - m_j \in [1, N] \wedge m_{j+1} - M \in [1, N]$. To prove $M - m_j \in [1, N]$, one can further apply the Boudot technique as proving knowledge of signatures $\mathsf{SgSign}(\delta_1)$ and $\mathsf{SgSign}(\delta_2)$ such that $M - m_j = {\delta_1}^2 + \delta_2$ to reduce the size of the public parameter that the MRA has to prepare. More precisely, in order to ensure that m_j and m_{j+1} used to prove $M - m_j \in [1, N]$ are the prohibited messages, a sender also proves knowledge of signatures $\mathsf{SgSign}(m_j, m_{j+1})$. This technique is originally developed by Nakanishi et al. [NFHF10] in the context of revocable group signature. We further apply the technique of Nakanishi et al., in order to construct an efficient RPKE scheme.

Putting all together, the ciphertext of the proposed construction has the form of (c, π), and each of components are computed as

$$c = \mathsf{PEnc}(pk_d, M; u),$$

$$\pi = NIZK \left\{ \begin{array}{l} (M, u, S'', S_1, S'_1, S_2, S'_2, \\ \delta_1, \delta_2, \epsilon_1, \epsilon_2, m_j, m_{j+1}) \\ : \mathsf{SgVerify}((m_j, m_{j+1}), S'') = 1 \\ \wedge\ \mathsf{SgVerify}(\delta_1, S_1) = 1 \\ \wedge\ \mathsf{SgVerify}(\epsilon_1, S'_1) = 1 \\ \wedge\ \mathsf{SgVerify}'(\delta_2, S_2) = 1 \\ \wedge\ \mathsf{SgVerify}'(\epsilon_2, S'_2) = 1 \\ \wedge\ M - m_j = {\delta_1}^2 + \delta_2 \bmod p \\ \wedge\ m_{j+1} - M = {\epsilon_1}^2 + \epsilon_2 \bmod p \\ \wedge\ c = \mathsf{PEnc}(pk_d, M; u) \end{array} \right\}. \tag{6.2}$$

We again emphasize that π, a non-interactive proof of knowledge, can be instantiated efficiently, which is obtained from algebraic properties of the involved public key encryption scheme and signature scheme (More concretely, algebraic property of the BB signature [BB08] and the BBS+ signature [ASM06, BBS04, FI06] is used, and for the detailed description of these two signature scheme see Sect. 6.4.3).

6.3.4 Updating pk_{MS}

The above idea does not provide the functionality of updating the message space, but a simple modification (which will be explained below) enables us to obtain such a functionality. To update the message space specified by the MRA from MS to MS' where the prohibited messages of MS and MS' are $\{m_1, \dots, m_r\}$ and $\{m'_1, \dots, m'_{r'}\}$ respectively, one may think that just re-publicizing signatures $\mathsf{SgSign}''(m'_i, m'_{i+1})$ for all $i \in \{0, \dots, r'\}$ is suffice to do that (where $m'_0 = 0$ and $m'_{r'+1} = N + 1$ as in the construction). However, in this way, a malicious sender will re-use some old signature $\mathsf{SgSign}''(m_i, m_{i+1})$ and try to fool the verification process, which inspects whether a ciphertext encrypts a value belonging to a new message space MS'. A simple way to avoid the above attack is to publicize signatures $\mathsf{SgSign}(t, m_i, m_{i+1})$ where t is a serial number, instead of $\mathsf{SgSign}(m_i, m_{i+1})$. In this case a malicious sender is no longer able to re-use old signatures to fool the verification process.

6.4 Constructions

In this section we describes the main proposed scheme. The proposed scheme borrow the ideas described in the previous section, which have been developed in the context of revocable group signature. Noticing the similarity between requirement of revocation and that of RPKE, in which the former demands that a group signature hides all partial information of the member's identity and makes the fact that the signer is not one of the revoked members publicly verifiable, whereas the latter demands a ciphertext to hide all partial information of the plaintext whereas it also claims the encrypted plaintext is not one of the prohibited messages in a publicly-verifiable form, we decide to borrow the idea from revocable group signature schemes to construct efficient RPKE schemes.

6.4.1 High Level Description of the Proposed Scheme

Using the ideas we mentioned above, we show a construction of RPKE. In order to give a high-level overview of the proposed scheme, we first show the proposed scheme in the form of generic construction. Then we present the specific construction of RPKE in a later section. The generic construction is based on an IND-CCA secure PKE, an EUF-CMA secure signature, and a Σ-protocol for Eq. (6.2). In the following, let $[1, N] = \{1, \ldots, N\}$ be a set of whole possible messages (they may or may not be prohibited), r be the number of prohibited messages. Let (PKg, PEnc, PDec) be an IND-CCA secure public key encryption scheme, (SgKg, SgSign, $\mathsf{SgVerify}$), (SgKg', SgSign', $\mathsf{SgVerify}'$), and (SgKg'', SgSign'', $\mathsf{SgVerify}''$) be EUF-CMA secure signature schemes. The construction is as follows:

$\mathsf{MRASetup}(1^\kappa)$**:** Run $(K_s, K_v) \leftarrow \mathsf{SgKg}(1^\kappa)$, $(K'_s, K'_v) \leftarrow \mathsf{SgKg}'(1^\kappa)$, and $(K''_s, K''_v) \leftarrow \mathsf{SgKg}''(1^\kappa)$. For $k \in [1, \lfloor \sqrt{N} \rfloor]$, compute $\sigma_{1,k} \leftarrow \mathsf{SgSign}'(K'_s, k)$. For $k \in [0, \lfloor 2\sqrt{N} \rfloor]$, compute $\sigma_{2,k} \leftarrow \mathsf{SgSign}''(K''_s, k)$. Output $pk_{\mathrm{MRA}} = (K_v, K'_v, K''_v, \{\sigma_{1,k}\}_{k=1}^{\lfloor \sqrt{N} \rfloor}, \{\sigma_{2,k}\}_{k=0}^{\lfloor 2\sqrt{N} \rfloor})$ and $sk_{\mathrm{MRA}} = (K_s, K'_s, K''_s)$.

$\mathsf{RKeyGen}(pk_{\mathrm{MRA}})$**:** Run $(pk, sk) \leftarrow \mathsf{PKg}(1^\kappa)$, and output $pk_d = pk$ and $sk_d = sk$.

$\mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS)$**:** Let $MS = [1, N] \setminus \{m_1, \ldots, m_r\}$ where $m_1 < \cdots < m_r$, $m_0 =$

0, and $m_{r+1} = N + 1$. Choose a current serial number $t \in \mathbb{Z}_p$. For $\ell \in [0, r]$, compute $\sigma_\ell \leftarrow \mathsf{SgSign}(K_s, t, m_\ell, m_{\ell+1})$. Output $pk_{MS} = (t, \{\sigma_\ell\}_{\ell=0}^{r})$.

$\mathsf{REnc}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, M)$**:** For $M \in MS$, find the position j such that $m_j < M < m_{j+1}$. If there is no such m_j (which means $M \notin MS$), output $\perp$. Otherwise, find σ_j from pk_{MS}, compute $\delta_1, \epsilon_1 \in [1, \lfloor \sqrt{N} \rfloor]$, and $\delta_2, \epsilon_2 \in [0, \lfloor 2\sqrt{N} \rfloor]$, where $M - m_j = {\delta_1}^2 + \delta_2$ and $m_{j+1} - M = {\epsilon_1}^2 + \epsilon_2$ and find σ_{1,δ_1}, σ_{2,δ_2}, σ_{1,ϵ_1}, and σ_{2,ϵ_2} from pk_{MRA}. Compute $c = \mathsf{PEnc}(pk_d, M; u)$, and π of the following relations:

$$NIZK\left\{\begin{array}{l}(M, u, \sigma_j, \sigma_{1,\delta_1}, \sigma_{2,\delta_2}, \sigma_{1,\epsilon_1}, \sigma_{2,\epsilon_2},\\ \delta_1, \delta_2, \epsilon_1, \epsilon_2, m_j, m_{j+1})\\ : \sigma_j = \mathsf{SgSign}(t, m_j, m_{j+1})\\ \ \wedge \mathsf{SgVerify}'(\delta_1, \sigma_{1,\delta_1}) = 1\\ \ \wedge \mathsf{SgVerify}''(\delta_2, \sigma_{2,\delta_2}) = 1\\ \ \wedge \mathsf{SgVerify}'(\epsilon_1, \sigma_{1,\epsilon_1}) = 1\\ \ \wedge \mathsf{SgVerify}''(\epsilon_2, \sigma_{2,\epsilon_2}) = 1\\ \ \wedge M - m_j = {\delta_1}^2 + \delta_2 \bmod p\\ \ \wedge m_{j+1} - M = {\epsilon_1}^2 + \epsilon_2 \bmod p\\ \ \wedge c = \mathsf{PEnc}(pk_d, M; u)\end{array}\right\}$$

Finally, output $C = (c, \pi)$.

$\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, C)$**:** Output 1 if π is a valid proof, and 0, otherwise.

$\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, MS, pk_{MS}, C)$**:** Verify the ciphertext as above and output $\mathsf{PDec}(sk_d, C)$ if the verification succeeds, otherwise output $\perp$.

The above construction, especially the zero-knowledge proof of Eq. (6.1), can be quite efficiently instantiated when one adopts appropriate digital signatures and Teranishi et al., Boudot, and Nakanishi et al. techniques. More concretely, the BBS+ signature [ASM06, BBS04, FI06] is applied for SgSign, and two instance of the BB signature [BB08] are applied for SgSign′ and SgSign″. When applying the BBS+ signature and the BB signature, adopting the techniques of [BBS04, NFHF10], the zero-knowledge proof of Eq. (6.1) is efficiently constructed, and the entire RPKE construction becomes drastically more efficient than the construction employing the general NIZK technique. We choose these two schemes because of the following reasons: the BB signature is chosen due to suitability for constructing higher-level protocols and additionally its efficiency. The BB scheme, which is not only one of the most efficient signature schemes in the discrete-log type schemes, but also allows us to construct efficient zero-knowledge protocols [BBS04] of proving the knowledge of the signature. We also employ the BBS+ scheme, in addition to the BB scheme, in order to sign sequences of integers without destroying algebraic property of the integers to be signed (like hashing integers into a single one), which property is utilized in constructing the zero-knowledge proof involving the BBS+ signatures. We choose the BBS+ scheme for this domain-extension purpose because this scheme is a natural extension of the BB scheme, and thus we do not need any additional assumption to that of the BB scheme. Other type of signatures is also known to allows this domain extension and to be able to provide appropriate zero-knowledge protocols (to name a few, the Camenisch-Lysyanskaya signature also allows this kind of domain extension [CL03, CL04]). However, these signature schemes require extra assumptions and may reduce simplicity of the resulted RPKE scheme.

A drawback of this construction is that the plaintext space has to be small. More concretely, $[1, N]$, the set of all possible (prohibited or allowed) messages, has to be just a polynomially (not exponentially) large. Due to the construction of NIZK proof, a message $M \in [1, N]$ have to be encoded into the underlying group as g^M, and hence a receiver must compute a discrete logarithm of g^M in order to recover the message M. This constraint causes an inefficient decryption. However, it can be bypassed by restricting N to be sufficiently small to compute a discrete logarithm efficiently. When one can use Pollard's lambda method, M can be recovered from g^M in $O(\sqrt{N})$ computation time.

6.4.2 Security Analysis

The above construction satisfies the security requirement of verification soundness and IND-MSR-CCA security.

Theorem 6.4. *The construction given above satisfies verification soundness if the underlying signature scheme* $(\mathsf{SgKg}, \mathsf{SgSign}, \mathsf{SgVerify})$ *is EUF-CMA secure, signatures* $(\mathsf{SgKg}'$, SgSign', $\mathsf{SgVerify}')$ *and* $(\mathsf{SgKg}''$, SgSign'', $\mathsf{SgVerify}'')$ *are EUF-wCMA secure, and the*

NIZK proof is constructed from Σ-protocol by using the Fiat-Shamir heuristics.

Proof. The NIZK proof has an extractor of the proved secret knowledge: given two accepting protocol views, where commitments are the same but challenges are different. By $\sigma^* = \mathsf{SgSign}(\cdot,\cdot,\cdot)$ extracted from the output of $\mathcal{A}$, we consider two cases (1) $\sigma^* \notin pk_{MS^*}$, and (2) $\sigma^* \in pk_{MS^*}$. Let $M^* := \mathsf{RDec}(pk_{\mathrm{MRA}}, pk_d, sk_d, C^*)$. From the definition of verification soundness, $M^* \in \{m_1, m_2, \ldots, m_r\}$.

Case 1: We construct an algorithm $\mathcal{B}$ that forges one of the underlying signature scheme (SgKg, SgSign, $\mathsf{SgVerify}$). Let C be the challenger of unforgeability game of this signature. C sends a public value for verification K_v to $\mathcal{B}$. $\mathcal{B}$ computes other public values, and sends pk_{MRA} to $\mathcal{A}$. By using the signing oracle of the unforgeability game, $\mathcal{B}$ can answer message space queries sent from $\mathcal{A}$. $\mathcal{A}$ outputs C^*. Since $\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, pk_{MS^*}, C^*) = 1$, using the extractor of the NIZK, $\mathcal{B}$ obtains σ^* and the corresponding signed messages. Since $\sigma^* \notin pk_{MS^*}$, σ^* is not an answer of the signing oracle. Therefore, $\mathcal{B}$ outputs a forged signature σ^* and wins.

Case 2: We construct an algorithm $\mathcal{B}'$ that forges one of the underlying signature scheme (SgKg', SgSign', $\mathsf{SgVerify}'$). Let C' be the challenger of unforgeability game of this signature under the weakly chosen message attack [BB08]. First, $\mathcal{B}'$ sends messages $1, \ldots, \lfloor \sqrt{N} \rfloor$ to C'. Although, we describe the attack of signatures $\sigma_{1,*}$, the attack of signatures $\sigma_{2,*}$ is similarly described, and therefore we omit this part (in this case $\mathcal{B}'$ sends messages $0, \ldots, \lfloor 2\sqrt{N} \rfloor$ to C'). C' sends a public value for verification K_v' to $\mathcal{B}'$. $\mathcal{B}'$ obtains $\{\sigma_{1,k}\}_{k=1}^{\lfloor \sqrt{N} \rfloor}$ from $\mathcal{A}$, computes other public values, and sends pk_{MRA} to $\mathcal{A}$. Since $\mathcal{B}'$ has a signing key K_s of the underlying signature scheme ($\mathsf{SgKg}, \mathsf{SgSign}, \mathsf{SgVerify}$), $\mathcal{B}'$ can answer message space queries. $\mathcal{A}$ outputs C^*. Since $\mathsf{VerifyMS}(pk_{\mathrm{MRA}}, pk_d, pk_{MS^*}, C^*) = 1$, using the extractor of the NIZK, $\mathcal{B}'$ obtains σ^* and the corresponding signed messages. Since $\sigma^* \in pk_{MS^*}$, let $\sigma^* := \mathsf{SgSign}(t, m_{j^*}, m_{j^*+1})$. Using the extractor of the NIZK, $\mathcal{B}'$ obtains δ_1, δ_2, σ_{1,δ_1}, σ_{2,δ_2}, ϵ_1, ϵ_2, σ_{1,ϵ_1}, and σ_{2,ϵ_2}, with the conditions $M^* - m_{j^*} \bmod p = {\delta_1}^2 + \delta_2$ and $m_{j^*+1} - M^* \bmod p = {\epsilon_1}^2 + \epsilon_2$. Next, we show that $m_{j^*+1} \leq M^*$ or $m_{j^*} \geq M^*$ holds as follows: Since $M^* \notin MS^*$ from the definition of verification soundness, there exists $m_{j'}$ such that $j' \in [1, r]$ and $M^* = m_{j'}$.

For all $m_\ell < m_{j'}$, $m_\ell < M^*$ and $m_{\ell+1} \leq M^*$ hold. In addition, for all $m_\ell \geq m_{j'}$, $m_\ell \geq M^*$ and $m_{\ell+1} > M^*$ hold. Therefore, for the consecutive m_{j^*} and m_{j^*+1}, $m_{j^*+1} \leq M^*$ or $m_{j^*} \geq M^*$ holds[*2]. This means $m_{j^*+1} - M^* \leq 0$ or $0 \geq M^* - m_{j^*}$ in $\mathbb{Z}$. If $0 \geq M^* - m_{j^*}$, then set $\delta := M^* - m_{j^*} \bmod p$, $\delta(1) := \delta_1$, $\delta(2) := \delta_2$, $\sigma_{1,\delta} := \sigma_{1,\delta_1}$, and $\sigma_{2,\delta} := \sigma_{2,\delta_2}$. Otherwise, If $m_{j^*+1} - M^* \leq 0$, then set $\delta := m_{j^*+1} - M^* \bmod p$, $\delta(1) := \epsilon_1$, $\delta(2) := \epsilon_2$, $\sigma_{1,\delta} := \sigma_{1,\epsilon_1}$, and $\sigma_{2,\delta} := \sigma_{1,\epsilon_2}$. Under the assumption $\lfloor \sqrt{N} \rfloor^2 + \lfloor 2\sqrt{N} \rfloor < p/2$, $\delta \in [p/2, p-1]$ holds, and therefore $\delta(1) \notin [1, \lfloor \sqrt{N} \rfloor]$ or $\delta(2) \notin [0, \lfloor 2\sqrt{N} \rfloor]$ hold. If $\delta(2) \notin [0, \lfloor 2\sqrt{N} \rfloor]$, then $\mathcal{B}'$ aborts. Note that the case $\delta(2) \notin [0, \lfloor 2\sqrt{N} \rfloor]$ can be captured in the attack of the signature scheme $(\mathsf{SgKg}'', \mathsf{SgSign}'', \mathsf{SgVerify}'')$. Now we assume that $\delta(1) \notin [1, \lfloor \sqrt{N} \rfloor]$. $\mathcal{B}'$ outputs a forged signature and message pair $(\delta(1), \sigma_{1,\delta(1)})$, and wins, since $\delta(1)$ is not an input of the signing oracle. □

Theorem 6.5. *The construction given above is IND-MSR-CCA secure if the underlying PKE scheme is IND-CCA secure and the NIZK proof is constructed from* Σ*-protocol by using Fiat-Shamir heuristics.*

Proof. Due to the zero-knowledge-ness of NIZK proof, any information is not revealed from π. Therefore, we can reduce the IND-MSR-CCA game to the IND-CCA game of the underlying PKE scheme. Let $\mathcal{A}$ be an adversary who breaks the IND-MSR-CCA security of our RPKE scheme, and C the challenger of the IND-CCA game of the corresponding PKE scheme. Then, we can construct an algorithm $\mathcal{B}$ that breaks the IND-CCA security of the underlying PKE scheme. First, C gives a public key of the PKE scheme pk to $\mathcal{B}$. $\mathcal{B}$ sets pk to pk_d, and sends pk_d to $\mathcal{A}$. When $\mathcal{A}$ issues a decryption query $C = (c, \pi)$, $\mathcal{B}$ checks whether C is a valid ciphertext or not. If C is valid, then $\mathcal{B}$ simply forwards the corresponding part of this query c to C as a decryption query of the IND-CCA game. When $\mathcal{A}$ sends the challenge messages M_0^* and M_1^*, $\mathcal{B}$ forwards M_0 and M_1 to C as the challenge messages. C returns the challenge ciphertext c^*. $\mathcal{B}$ computes the challenge ciphertext of RPKE by applying the simulated NIZK proofs, say π^*. $\mathcal{B}$ sends the challenge ciphertext of RPKE (c^*, π^*) to $\mathcal{A}$. If $\mathcal{A}$ issues a valid (which means the $\mathsf{VerifyMS}$ algorithm returns 1) decryption query $C = (c^*, \pi)$,

[*2] Note that both cases $M^* < m_1$ and $m_r < M^*$ are included in these two cases.

then we can construct an algorithm $\mathcal{B}'$ who extracts signed messages m_j, δ_1, and δ_2 from C, computes $M_b = {\delta_1}^2 + \delta_2 + m_j$, outputs b, and wins. For other decryption queries, $\mathcal{B}$ can apply the decryption oracle of the underlying PKE scheme. Finally, $\mathcal{A}$ outputs the guessing bit, and $\mathcal{B}$ also outputs the same bit as the guessing bit of the IND-CCA game. $\square$

6.4.3 Concrete Construction

6.4.3.1 Construction

In this section, we give a concrete instantiation of RPKE. From the viewpoint of efficiency, we apply BB [BB08] and BBS+ [ASM06, BBS04, FI06] signatures to implement Teranishi/Nakanishi proof system. In addition, we apply an ElGamal type double encryption $\mathsf{DoubleEnc}$ to implement the building PKE scheme.

In the following scheme, $(g, g_1, g_2, g_3, g_4, Y_1)$ is a verification key of BBS+ signatures $\{(B_j, y_j, z_j)\}_{j=1}^{r-1}$, $(\tilde{g}, Y_2)$ is a verification key of BB signatures $\{F_{1,k}\}_{k=1}^{\lfloor\sqrt{N}\rfloor}$, $(\dot{g}, Y_3)$ is a verification key of BB signatures $\{F_{2,k}\}_{k=0}^{\lfloor 2\sqrt{N}\rfloor}$, and $pk_d = (\hat{f}, \hat{g}_1, \hat{g}_2, \hat{h})$ is a public key of the double encryption scheme $\mathsf{DoubleEnc}$. For a plaintext $M' \in \mathbb{G}'$ and a random number $u \in \mathbb{Z}_p$, $\mathsf{DoubleEnc}_{pk_d}(M'; u) = (\hat{g}_1^u, \hat{g}_2^u, M' \cdot \hat{h}^u)$. Other parameters are for computing NIZK proofs. These NIZK proofs work for exponent in $\mathbb{Z}_p$ so we need to encrypt M by $\hat{f}^M$ for some generator $\hat{f}$. Therefore, to apply this proving system to PKE, we require that a plaintext of the building PKE scheme $\mathsf{PEnc}(\cdot)$ is $\hat{f}^M$, and the knowledge of M need to be proved from a ciphertext $\mathsf{PEnc}(\hat{f}^M)$ by using NIZK system. When receiver obtains $\hat{f}^M$ by using own sk_d, receiver needs to solve the DL problem to obtain M from $\hat{f}^M$. Therefore, as in Boneh et al. [BGN05] and Okamoto et al. [OT08], we assume N is small with the condition that the DL problem $(\hat{f}, \hat{f}^M)$ can be solved efficiently (e.g., by using baby-step-giant-step algorithm or Pollard's lambda method with expected time $O(\sqrt{N})$). For our purpose the lambda method is particularly more suitable than the (for example) rho method. This is because the running time of the former method is proportional to the square root of the size of a previously known range in which the discrete logarithm lies, whereas that of the latter is proportional to the square root of the order of the group. That is, if one previously knows that the discrete logarithm x to be computed is in a certain interval $[a, b]$, which holds in the current context, the lambda method requires roughly $2\sqrt{b-a}$ multiplications.

The concrete construction we propose is as follows:

$\mathsf{MRASetup}(1^\kappa)$**:** Let $(\mathbb{G}, \mathbb{G}_T)$ be a bilinear group with a κ-bit prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. In addition, let $\mathbb{G}'$ be a DDH-hard group with the same order p. Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function for NIZK proofs. Choose generators $g, \tilde{g}, \dot{g}, g_1, \tilde{g}_1, g_2, g_3, g_4, g_5 \in \mathbb{G}$, $\hat{f} \in \mathbb{G}'$, a signing key of BBS+ signatures $X_1 \in \mathbb{Z}_p$, and signing keys of BB signatures X_2, $X_3 \in \mathbb{Z}_p$, and compute the a verification key of BBS+ signatures $Y_1 = g^{X_1}$, and verification keys of BB signatures $Y_2 = g^{X_2}$ and $Y_3 = g^{X_3}$. For $k \in [1, \lfloor\sqrt{N}\rfloor]$, compute $\mathsf{SgSign}'_{\mathrm{BB}}(k) := F_{1,k} = \tilde{g}^{\frac{1}{X_2+k}}$. For $k \in [0, \lfloor 2\sqrt{N}\rfloor]$, compute $\mathsf{SgSign}''_{\mathrm{BB}}(k) := F_{2,k} = \dot{g}^{\frac{1}{X_3+k}}$. Output $pk_{\mathrm{MRA}} = (p, e, \mathbb{G}, \mathbb{G}_T, \mathbb{G}', H, Y_1, Y_2, Y_3, \{F_{1,k}\}_{k=1}, \{F_{2,k}\}_{k=0}^{\lfloor 2\sqrt{N}\rfloor}, \hat{f})$, and $sk_{\mathrm{MRA}} = (X_1, X_2, X_3)$.

$\mathsf{RKeyGen}(pk_{\mathrm{MRA}})$**:** Choose $\hat{g}_1, \hat{g}_2 \in \mathbb{G}'$ and $z \in \mathbb{Z}_p$, and compute $\hat{h} = \hat{g}_1^z$. Output a public key of an ElGamal type double encryption scheme $pk_d = (\hat{g}_1, \hat{g}_2, \hat{h})$ and the corresponding secret key $sk_d = z$.

$\mathsf{MSSetup}(pk_{\mathrm{MRA}}, sk_{\mathrm{MRA}}, MS)$**:** Let $(m_1, m_2, \ldots, m_r)$ be consecutive prohibited messages, $m_0 = 0$, and $m_{r+1} = N + 1$. Choose a current serial number $t \in \mathbb{Z}_p$. For $\ell \in [0, r]$, compute BBS+ signatures of three signed messages $(t, m_\ell, m_{\ell+1})$ $\mathsf{SgSign}_{\mathrm{BBS+}}(t, m_\ell, m_{\ell+1}) := (B_\ell, y_\ell, z_\ell)$, where $B_\ell = (g_1^t g_2^{m_\ell} g_3^{m_{\ell+1}} g_4^{y_\ell} g)^{\frac{1}{X_1+z_\ell}}$, and y_ℓ, $z_\ell \in \mathbb{Z}_p$. Output $pk_{MS} = (t, \{(m_\ell, m_{\ell+1}, B_\ell, y_\ell, z_\ell)\}_{\ell=0}^r)$.

$\mathsf{REnc}(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, M)$**:** For $M \in MS$, find the position j such that $m_j < M < m_{j+1}$. If there is no such m_j (which means $M \notin MS$), output $\perp$. compute $\delta_1, \epsilon_1 \in [1, \lfloor\sqrt{N}\rfloor]$, and $\delta_2, \epsilon_2 \in [0, \lfloor 2\sqrt{N}\rfloor]$, where $M - m_j = {\delta_1}^2 + \delta_2$ and $m_{j+1} - M = {\epsilon_1}^2 + \epsilon_2$ and find $\mathsf{SgSign}'_{\mathrm{BB}}(\delta_1) = F_{1,\delta_1}$, $\mathsf{SgSign}''_{\mathrm{BB}}(\delta_2) = F_{2,\delta_2}$, $\mathsf{SgSign}'_{\mathrm{BB}}(\epsilon_1) = F_{1,\epsilon_1}$, and $\mathsf{SgSign}''_{\mathrm{BB}}(\epsilon_2) = F_{2,\epsilon_2}$ from pk_{MRA}. Compute $c = \mathsf{DoubleEnc}_{pk_d}(\hat{f}^M; u)$, and π of the following relations:

$$\pi = NIZK \left\{ \begin{array}{l} (M, S'', S_1, S'_1, S_2, S'_2, \delta_1, \delta_2, \epsilon_1, \epsilon_2) \\ : S'' = \mathsf{SgSign}_{\mathrm{BBS+}}(t, m_j, m_{j+1}) \\ \wedge\ \mathsf{SgVerify}'_{\mathrm{BB}}(\delta_1, S_1) = 1 \\ \wedge\ \mathsf{SgVerify}''_{\mathrm{BB}}(\delta_2, S_2) = 1 \\ \wedge\ \mathsf{SgVerify}'_{\mathrm{BB}}(\epsilon_1, S'_1) = 1 \\ \wedge\ \mathsf{SgVerify}''_{\mathrm{BB}}(\epsilon_2, S'_2) = 1 \\ \wedge\ M - m_j = {\delta_1}^2 + \delta_2 \bmod p \\ \wedge\ m_{j+1} - M = {\epsilon_1}^2 + \epsilon_2 \bmod p \\ \wedge\ c = \mathsf{DoubleEnc}_{pk_d}(\hat{f}^M; u) \end{array} \right\}$$

Concretely, choose $\alpha, \beta_{1,1}, \beta_{1,2}, \beta_{2,1}, \beta_{2,2}$, u, ξ_1, ξ'_1, ξ_2, $\xi'_2 \in \mathbb{Z}_p$, compute $C_1 = B_j g_5^{\alpha}$, $C_2 = F_{1,\delta_1} g_5^{\beta_{1,1}}$, $C_3 = F_{2,\delta_2} g_5^{\beta_{1,2}}$, $C_4 = F_{1,\epsilon_1} g_5^{\beta_{2,1}}$, $C_5 = F_{2,\epsilon_2} g_5^{\beta_{2,2}}$, $C_6 = \tilde{g}^{\delta_1} \tilde{g}_1^{\xi_1}$, $C_7 = \tilde{g}^{{\delta_1}^2} \tilde{g}_1^{\xi'_1}$, $C_8 = \tilde{g}^{\epsilon_1} \tilde{g}_1^{\xi_2}$, $C_9 = \tilde{g}^{{\epsilon_1}^2} \tilde{g}_1^{\xi'_2}$, $\xi''_1 := \xi'_1 - \xi_1 \delta_1$, $\xi''_2 := \xi'_2 - \xi_2 \epsilon_1$, $C_{10} = \hat{g}_1^u$, $C_{11} = \hat{g}_2^u$, $C_{12} = \hat{f}^M \hat{h}^u$, $\zeta = \alpha z_j$, $\theta_{1,1} := \beta_{1,1}\delta_1$, $\theta_{1,2} := \beta_{1,2}\delta_2$, $\theta_{2,1} := \beta_{2,1}\epsilon_1$, and $\theta_{2,2} := \beta_{2,2}\epsilon_2$. In addition, compute

$$\pi = NIZK\left\{\begin{array}{l}(M,\zeta,\alpha,y_j,z_j,m_j,m_{j+1},\\ \delta_1,\delta_2,\epsilon_1,\epsilon_2,\theta_{1,1},\theta_{1,2},\theta_{2,1},\theta_{2,2},\\ \beta_{1,1},\beta_{1,2},\beta_{2,1},\beta_{2,2},\xi_1,\xi_1',\xi_1'',\xi_2,\xi_2',\xi_2'',u)\\ :e(C_1,Y_1)/e(g,g)=\\ \qquad e(g_5,Y_1)^{\alpha}e(g_5,g)^{\zeta}e(g_1^t,g)\\ \qquad e(g_2,g)^{m_j}e(g_3,g)^{m_{j+1}}\\ \qquad e(g_4,g)^{y_j}/e(C_1,g)^{z_j}\\ \wedge\, e(C_2,Y_2)/e(\tilde{g},g)=\\ \qquad e(g_5,Y_2)^{\beta_{1,1}}e(g_5,g)^{\theta_{1,1}}/e(C_2,g)^{\delta_1}\\ \wedge\, e(C_3,Y_3)/e(\dot{g},g)=\\ \qquad e(g_5,Y_3)^{\beta_{1,2}}e(g_5,g)^{\theta_{1,2}}/e(C_3,g)^{\delta_2}\\ \wedge\, e(C_4,Y_2)/e(\tilde{g},g)=\\ \qquad e(g_5,Y_2)^{\beta_{2,1}}e(g_5,g)^{\theta_{2,1}}/e(C_4,g)^{\epsilon_1}\\ \wedge\, e(C_5,Y_3)/e(\dot{g},g)=\\ \qquad e(g_5,Y_3)^{\beta_{2,2}}e(g_5,g)^{\theta_{2,2}}/e(C_5,g)^{\epsilon_2}\\ \wedge\, C_6=\tilde{g}^{\delta_1}\tilde{g}_1^{\xi_1}\wedge C_7=C_6^{\delta_1}\tilde{g}_1^{\xi_1''}\\ \wedge\, C_7=\tilde{g}^{-\delta_2+M-m_j}\tilde{g}_1^{\xi_1'}\\ \wedge\, C_8=\tilde{g}^{\epsilon_1}\tilde{g}_1^{\xi_2}\wedge C_9=C_8^{\epsilon_1}\tilde{g}_1^{\xi_2''}\\ \wedge\, C_9=\tilde{g}^{-\epsilon_2+m_{j+1}-M}\tilde{g}_1^{\xi_2'}\\ \wedge\, C_{10}=\hat{g}_1^u\wedge C_{11}=\hat{g}_2^u\wedge C_{12}=\hat{f}^M\hat{h}^u\end{array}\right\}$$

(Detailed NIZK proofs are described in the following). Output a ciphertext $C = (C_1,\dots,C_{12},\pi)$.

VerifyMS$(pk_{\mathrm{MRA}}, pk_d, MS, pk_{MS}, C)$**:** Output 1 if π is a valid proof, and 0, otherwise.

RDec$(pk_{\mathrm{MRA}}, pk_d, sk_d, MS, pk_{MS}, C)$**:** Verify the ciphertext C under pk_{MS}. If the verification succeeds, compute $\hat{f}^M = C_{12}/C_{10}^z$, solve the DL problem $(\hat{f}, \hat{f}^M)$, and output M. If the verification fails, output $\perp$.

If MS is changed (let MS' be a new message space), then MRA chooses t' ($t' \neq t$ for all previous t), and opens BBS+ signatures $\mathsf{SgSign}_{\mathrm{BBS+}}(t', m'_\ell, m'_{\ell+1})$ for all $\ell \in [0, r']$ as $pk_{MS'}$, where $(m'_1, m'_2, \dots, m'_{r'})$ is the new consecutive prohibited messages, $m'_0 = 0$, and $m'_r = N$.

6.4.3.2 Detailed NIZK Proofs

Here, we show the detailed proof π of our RPKE scheme. Concretely, π is computed as follows. Note that all pairing values are pre-computable.

1. Choose $r_M, r_\zeta, r_\alpha, r_{y_j}, r_{z_j}, r_{m_j}, r_{m_{j+1}}, r_{\delta_1}, r_{\delta_2}, r_{\epsilon_1}, r_{\epsilon_2}, r_{\theta_{1,1}}, r_{\theta_{1,2}}, r_{\theta_{2,1}}, r_{\theta_{2,2}}, r_{\beta_{1,1}}, r_{\beta_{1,2}}, r_{\beta_{2,1}}, r_{\beta_{2,2}}, r_{\xi_1}, r_{\xi_1'}, r_{\xi_1''}, r_{\xi_2}, r_{\xi_2'}, r_{\xi_2''}, r_u \in \mathbb{Z}_p$.
2. Compute

$$\begin{aligned}
R_1 &= e(g_5, Y_1)^{r_\alpha} e(g_5, g)^{r_\zeta - \alpha r_{z_j}} e(g_1, g)^t \\
&\qquad e(g_2, g)^{r_{m_j}} e(g_3, g)^{r_{m_{j+1}}} e(g_4, g)^{r_{y_j}} / e(B_j, g)^{r_{z_j}}, \\
R_2 &= e(g_5, Y_2)^{r_{\beta_{1,1}}} e(g_5, g)^{r_{\theta_{1,1}} - \beta_{1,1} r_{\delta_1}} / e(F_{1,\delta_1}, g)^{r_{\delta_1}}, \\
R_3 &= e(g_5, Y_3)^{r_{\beta_{1,2}}} e(g_5, g)^{r_{\theta_{1,2}} - \beta_{1,2} r_{\delta_2}} / e(F_{2,\delta_2}, g)^{r_{\delta_2}}, \\
R_4 &= e(g_5, Y_2)^{r_{\beta_{2,1}}} e(g_5, g)^{r_{\theta_{2,1}} - \beta_{2,1} r_{\epsilon_1}} / e(F_{1,\epsilon_1}, g)^{r_{\epsilon_1}}, \\
R_5 &= e(g_5, Y_3)^{r_{\beta_{2,2}}} e(g_5, g)^{r_{\theta_{2,2}} - \beta_{2,2} r_{\epsilon_2}} / e(F_{2,\epsilon_2}, g)^{r_{\epsilon_2}}, \\
R_6 &= \tilde{g}^{r_{\delta_1}} \tilde{g}_1^{r_{\xi_1}}, \\
R_7 &= C_6^{r_{\delta_1}} \tilde{g}_1^{r_{\xi_1''}}, \\
R_8 &= \tilde{g}^{-r_{\delta_2} + r_M - r_{m_j}} \tilde{g}_1^{r_{\xi_1'}}, \\
R_9 &= \tilde{g}^{r_{\epsilon_1}} \tilde{g}_1^{r_{\xi_2}}, \\
R_{10} &= C_8^{r_{\epsilon_1}} \tilde{g}_1^{r_{\xi_2''}}, \\
R_{11} &= \tilde{g}^{-r_{\epsilon_2} + r_{m_{j+1}} - r_M} \tilde{g}_1^{r_{\xi_2'}}, \\
R_{12} &= \hat{g}_1^{r_u}, \\
R_{13} &= \hat{g}_2^{r_u}, \\
R_{14} &= \hat{f}^{r_M} \hat{h}^{r_u}.
\end{aligned}$$

3. Compute $c = H(R_1, \ldots, R_{14}, C_1, \ldots, C_{12}, pk_{\mathrm{MRA}}, pk_{MS}, pk_d)$
4. Compute $s_M = r_M + cM$, $s_\zeta = r_\zeta + c\zeta$, $s_\alpha = r_\alpha + c\alpha$, $s_{y_j} = r_{y_j} + cy_j$, $s_{z_j} = r_{z_j} + cz_j$, $s_{m_j} = r_{m_j} + cm_j$, $s_{m_{j+1}} = r_{m_{j+1}} + cm_{j+1}$, $s_{\delta_1} = r_{\delta_1} + c\delta_1$, $s_{\delta_2} = r_{\delta_2} + c\delta_2$, $s_{\epsilon_1} = r_{\epsilon_1} + c\epsilon_1$, $s_{\epsilon_2} = r_{\epsilon_2} + c\epsilon_2$, $s_{\theta_{1,1}} = r_{\theta_{1,1}} + c\theta_{1,1}$, $s_{\theta_{1,2}} = r_{\theta_{1,2}} + c\theta_{1,2}$, $s_{\theta_{2,1}} = r_{\theta_{2,1}} + c\theta_{2,1}$, $s_{\theta_{2,2}} = r_{\theta_{2,2}} + c\theta_{2,2}$,

$s_{\beta_{1,1}} = r_{\beta_{1,1}} + c\beta_{1,1}$, $s_{\beta_{1,2}} = r_{\beta_{1,2}} + c\beta_{1,2}$, $s_{\beta_{2,1}} = r_{\beta_{2,1}} + c\beta_{2,1}$, $s_{\beta_{2,2}} = r_{\beta_{2,2}} + c\beta_{2,2}$, $s_{\xi_1} = r_{\xi_1} + c\xi_1$, $s_{\xi'_1} = r_{\xi'_1} + c\xi'_1$, $s_{\xi''_1} = r_{\xi''_1} + c\xi''_1$, $s_{\xi_2} = r_{\xi_2} + c\xi_2$, $s_{\xi'_2} = r_{\xi'_2} + c\xi'_2$, $s_{\xi''_2} = r_{\xi''_2} + c\xi''_2$, and $s_u = r_u + cu$.

5. Output $C = (C_1, \ldots, C_{12}, \pi)$, where $\pi = (c,\ s_M,\ s_\zeta,\ s_\alpha,\ s_{y_j},\ s_{z_j},\ s_{m_j},\ s_{m_{j+1}},\ s_{\delta_1},\ s_{\delta_2},\ s_{\epsilon_1},\ s_{\epsilon_2},\ s_{\theta_{1,1}},\ s_{\theta_{1,2}},\ s_{\theta_{2,1}},\ s_{\theta_{2,2}},\ s_{\beta_{1,1}},\ s_{\beta_{1,2}},\ s_{\beta_{2,1}},\ s_{\beta_{2,2}},\ s_{\xi_1},\ s_{\xi'_1},\ s_{\xi''_1},\ s_{\xi_2},\ s_{\xi'_2},\ s_{\xi''_2},\ s_u)$.

Next, we show the verification of the above π. Note that all pairing values are precomputable, except the followings $e(C_1, g^{s_{z_j}} Y_1^c)$, $e(C_2, g^{s_{\delta_1}} Y_2^c)$, $e(C_3, g^{s_{\delta_2}} Y_3^c)$, $e(C_4, g^{s_{\epsilon_1}} Y_2^c)$, and $e(C_5, g^{s_{\epsilon_2}} Y_3^c)$.

1. Compute

$$
\begin{aligned}
R'_1 &= e(g_5, Y_1)^{s_\alpha} e(g_5, g)^{s_\zeta} e(g_1, g)^t \\
&\quad e(g_2, g)^{s_{m_j}} e(g_3, g)^{s_{m_{j+1}}} e(g_4, g)^{s_{y_j}} \\
&\quad e(g, g)^c / e(C_1, g^{s_{z_j}} Y_1^c), \\
R'_2 &= e(g_5, Y_2)^{s_{\beta_{1,1}}} e(g_5, g)^{s_{\theta_{1,1}}} e(\tilde{g}, g)^c / e(C_2, g^{s_{\delta_1}} Y_2^c), \\
R'_3 &= e(g_5, Y_3)^{s_{\beta_{1,2}}} e(g_5, g)^{s_{\theta_{1,2}}} e(\dot{g}, g)^c / e(C_3, g^{s_{\delta_2}} Y_3^c), \\
R'_4 &= e(g_5, Y_2)^{s_{\beta_{2,1}}} e(g_5, g)^{s_{\theta_{2,1}}} e(\tilde{g}, g)^c / e(C_4, g^{s_{\epsilon_1}} Y_2^c), \\
R'_5 &= e(g_5, Y_3)^{s_{\beta_{2,2}}} e(g_5, g)^{s_{\theta_{2,2}}} e(\dot{g}, g)^c / e(C_5, g^{s_{\epsilon_2}} Y_3^c), \\
R'_6 &= \tilde{g}^{s_{\delta_1}} \tilde{g}_1^{s_{\xi_1}} C_6^{-c}, \\
R'_7 &= C_6^{s_{\delta_1}} \tilde{g}_1^{s_{\xi''_1}} C_7^{-c}, \\
R'_8 &= \tilde{g}^{-s_{\delta_2} + s_M - s_{m_j}} \tilde{g}_1^{s_{\xi'_1}} C_7^{-c}, \\
R'_9 &= \tilde{g}^{s_{\epsilon_1}} \tilde{g}_1^{s_{\xi_2}} C_8^{-c}, \\
R'_{10} &= C_8^{s_{\epsilon_1}} \tilde{g}_1^{s_{\xi''_2}} C_9^{-c}, \\
R'_{11} &= \tilde{g}^{-s_{\epsilon_2} + s_{m_{j+1}} - s_M} \tilde{g}_1^{s_{\xi'_2}} C_9^{-c}, \\
R'_{12} &= \hat{g}_1^{s_u} C_{10}^{-c}, \\
R'_{13} &= \hat{g}_2^{s_u} C_{11}^{-c}, \\
R'_{14} &= \hat{f}^{s_M} \hat{h}^{s_u} C_{12}^{-c}.
\end{aligned}
$$

2. Check $c \overset{?}{=} H(R'_1, \ldots, R'_{14}, C_1, \ldots, C_{12}, pk_{\mathrm{MRA}}, pk_{MS}, pk_d)$.

The security of the above construction is described as follows. A proof of the theorem can be obtained with a similar way to the proof of Theorem 6.4 and 6.5.

Theorem 6.6. *The construction has verification soundness in the random oracle model when the q-strong Diffie-Hellman assumption holds on the bilinear group* $(\mathbb{G}, \mathbb{G}_T)$. *The construction is IND-MSR-CCA secure in the random oracle model, when the DDH assumption holds on* $\mathbb{G}'$.

6.5 Alternative Constructions for Small Message Space

In this section we describe alternative constructions of RPKE, which is suitable for a small set of permitted message space. Comparing with the construction in the previous section, the constructions here have an advantage in terms of performance, when a very few message is allowed by the MRA.

Such a small message space frequently appears in several application. Recently, Nuida et al. proposed privacy-preserving database search protocols [NSA+12], whose search queries consist of bit-string which is encrypted in a bit-by-bit manner (with some additive-homomorphic encryption scheme). In order to protect the database from information leakage, it is important to prohibit a (possibly malicious) client from sending a query which encrypts neither 0 nor 1. These bit-by-bit encryption is also used in a different context. One of the example is the non-interactive proof system proposed by Gorth, Ostrovsky, and Sahai [GOS12]. The proof system, which is for circuit satisfiability, encrypts all the assignments for the wires for each wire, and demonstrates that for all (NAND) gates, the wires that are connected to the gate satisfies NAND relation.

The construction presented below is based on the OR-proof technique [CDS94], which is able to prove that one of pre-specified statements are holds without revealing which statements actually holds (in fact in a zero-knowledge manner). Utilizing this functionality, the ciphertext of the scheme includes a non-interactive proof which shows that the encrypted message is one of the permitted messages, which are constructed by the OR-proof technique. For simplicity we only present a construction for the single-bit message space of {0, 1}, but it would be straightforward to extend it a more general message space.

The underlying intuition behind the OR-proof is that two (interactive) proofs for the two pre-specified statements are executed *in parallel* in which only one of these proofs is in fact performed *with a real witness*, while the other is *simulated* by the zero-knowledge simulator. This feature is achieved by allowing the prover to control (part of) the challenge message. The OR-proof construction, which runs two protocol instances in parallel, allows the prover to control the challenge message of *one of* the two protocol instances, while the construction gives the full control of the challenge message of the *other* protocol instance to the verifier. This allows the prover to convince the verifier without knowing the witness for one protocol instance, while it also enforce the prover to know the witness for the other protocol instance. With this feature the entire protocol ensures that the prover (who convince the verifier) knows at least one of the witnesses of the pre-specified statements.

The following RPKE construction proves the following statement using the OR-proof technique: given a ciphertext (C_1, C_2, C_3) and a public key $\hat{g}_1$, $\hat{g}_2$, $\hat{h}$, and $\hat{f}$, there exists u (randomness of the ciphertext) such that either

$$(C_1, C_2, C_3) = (\hat{g}_1^u, \hat{g}_2^u, \hat{h}^u \hat{f}^0)$$

(which suggests that the plaintext is 0) or

$$(C_1, C_2, C_3) = (\hat{g}_1^u, \hat{g}_2^u, \hat{h}^u \hat{f}^1)$$

(which suggests the plaintext is 1). In this way it achieves the functionality of RPKE with message space $\{0, 1\}$. The components C_1 and C_3 constitute the ElGamal encryption, while the component C_2 establishes chosen-ciphertext security (in combination with the zero-knowledge proof). The mechanism behind the chosen-ciphertext security is quite similar to the well-known Cramer-Shoup encryption [CS03b].

MRASetup(1^κ). Choose a DDH-hard $\mathbb{G}'$ of prime order p of length κ and a random element $\hat{f} \in \mathbb{G}'$. Output $pk_{MS} = (\mathbb{G}', \hat{f}, H)$ and $sk_{\mathrm{MRA}} = \emptyset$, where H is a hash function (modeled as the random oracle).

MSSetup(pk_{MRA}, MS). As mentioned above, we here only consider the case of $MS = \{0, 1\}$. In this case the algorithm simply output $pk_{MS} = (m_1, \dots, m_n)$. For a more general case in which $MS = \{m_1, \dots, m_n\}$, it outputs $pk_{MS} = (m_1, \dots, m_n)$.

RKeyGen(pk_{MRA}). Choose a random integer $z \in \mathbb{Z}_p$ and random elements $\hat{g}_1\ \hat{g}_2 \in \mathbb{G}'$. Compute $\hat{h} = \hat{g}_1^z$ and output $pk_{\mathrm{dec}} = (\hat{g}_1, \hat{g}_2, \hat{h})$ and $sk_{\mathrm{dec}} = z$.

REnc($pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, MS, pk_{MS}, M$). To encrypt $M \in \{0, 1\}$, choose a random integer $u \in \mathbb{Z}_p$ and compute $(C_1, C_2, C_3) = (\hat{g}_1^u, \hat{g}_2^u, \hat{h}^u \hat{f}^M)$. Further compute an NIZK proof, by choosing random r_M, c_{1-M}, s_{1-M} from $\mathbb{Z}_p$, compute

$$R_{M,1} = \hat{g}_1^{r_M}, \tag{6.3a}$$

$$R_{M,2} = \hat{g}_2^{r_M}, \tag{6.3b}$$

$$R_{M,3} = \hat{h}^{r_M}, \tag{6.3c}$$

$$R_{1-M,1} = \hat{g}_1^{s_{1-M}} / C_1^{c_{1-M}}, \tag{6.3d}$$

$$R_{1-M,2} = \hat{g}_2^{s_{1-M}} / C_2^{c_{1-M}} \tag{6.3e}$$

$$R_{1-M,3} = \hat{h}^{s_{1-M}} / (C_3 / \hat{f}^{1-M})^{c_{1-M}}. \tag{6.3f}$$

Then compute $c = H(R_{0,1}, R_{0,2}, R_{0,3}, R_{1,1}, R_{1,2}, R_{1,3}, C_1, C_2, C_3, pk)$. Finally compute $c_M = c - c_{1-M}$ and $s_M = r_M + c_M u$, and output $C = (C_1, C_2, C_3, c_0, c_1, s_0, s_1)$ as the ciphertext.

VerifyMS($pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, MS, pk_{MS}, C$). Compute

$$R'_{0,1} = \hat{g}_1^{s_0} / C_1^{c_0},$$

$$R'_{0,2} = \hat{g}_2^{s_0} / C_2^{c_0},$$

$$R'_{0,3} = \hat{h}^{s_0} / C_3^{c_0},$$

$$R'_{1,1} = \hat{g}_1^{s_1} / C_1^{c_1},$$

$$R'_{1,2} = \hat{g}_2^{s_1} / C_2^{c_1},$$

$$R'_{1,3} = \hat{h}^{s_1} / (C_3 / \hat{f})^{c_0},$$

then verify whether the equation $c_0 + c_1 = H(R'_{0,1}, R'_{0,2}, R'_{0,3}, R'_{1,1}, R'_{1,2}, R'_{1,3}, C_1, C_2, C_3,$

pk) holds.

$\mathsf{RDec}(pk_{\mathrm{MRA}}, pk_{\mathrm{dec}}, sk_{\mathrm{dec}}, MS, pk_{MS}, C)$**.** Verify the ciphertext as above, and output $\perp$ if the verification fails. Otherwise compute C_3/C_1^z and output 0 if it is equal to 1, output 1 if equal to $\hat{f}$, otherwise output $\perp$.

In the above construction, Eqs. (6.3) show how the OR-proof is constructed. Eqs. (6.3a)-(6.3c) is computing the proof with the *real witness* u, whereas Eqs. (6.3d)-(6.3f) is computing the *simulated proof* without the corresponding valid witness. The relation $c_M = c - c_{1-M}$ enforces the prover to arbitrary choose only one of c_M and c_{1-M} (in this case the prover chooses c_{1-M} by itself for a successful simulation), as the value c is determined by the output of the random oracle. This fact is crucial for constructing a secure OR-proof. More concretely, the fact that the prover can control c_{1-M} allows the prover to construct the simulated proof, and the fact that the prover cannot control c_M ensures soundness (the property that no adversarial provers cannot construct a proof that passes the verification for a ciphertext that encrypts neither 0 nor 1).

Security of this construction is described as below.

Theorem 6.7. *The construction has verification soundness in the random oracle model. Provided the DDH assumption, the construction is IND-MSR-CCA secure in the random oracle model.*

Part IV

Conclusion

Chapter 7

Conclusion

The thesis consists of the following two contributions:

(I) Providing guidelines for constructing new cryptographic primitives, and

(II) providing a more comprehensive approach for detecting overlooked threats.

In particular, we presented the following four example of the above approach. As contributions from the viewpoint (II), Chap. 3 and Chap 6 proposed two new security notion and extension of cryptographic primitives. Chap. 3 showed that a new security notion for group signature is obtained from a security notion (and its corresponding security notion) for public-key encryption called PKENO, while Chap. 6 showed that a new extension of public-key encryption is obtained from an extension of group signature called revocable group signature. In addition, from the viewpoint (I), Chap. 4 showed a necessity condition for obtaining a group signature scheme with the new security notion by presenting a generic construction of cryptographic primitives. Finally, Chap. 5 showed that a slightly different extension of public-key encryption called threshold encryption, is in fact tightly related to the PKENO extension.

In particular, our approach is providing guidelines for designing cryptographic schemes by showing generic constructions of some primitive with some security notions from other primitive(s) with other security notions. The existence of such a generic construction not only shows that it is sufficient to construct the latter primitives for obtaining the former scheme, but also shows that it is unavoidable to rely on an complexity theoretic assumption strong enough to construct the latter primitives. In other words, the generic construction suggests that the latter primitives themselves can be a crucial building block for the former primitive.

In addition, when one tries to obtain such a generic construction, it might be possible that there are no known security notions for some primitive that exactly corresponds to the security notion of the other primitive. In that case one can define a new security notions to complement the correspondence. We argued that such a new security notion can capture some overlooked practical threats, as the corresponding old security notion is defined to capture some natural and practical threats.

Following these approach, this thesis present the following contributions.

In Chap. 3, we identified the relationship between an extension of public-key encryption called PKENO and an anonymous authentication primitive called group signature. There exists an relationship between group signature and public-key encryption [BMW03, BSZ05, AW04, OFHO09], in which the CCA security for public-key encryption roughly corresponds to the anonymity notion for group signature. However, there had been no known notions for group signature that corresponds to the security notions achieved by PKENO, and thus we defined this as *opening soundness*. We further discussed on the practical importance of opening soundness, and present concrete scenarios in which opening soundness serves a crucial role. We also presented group signature schemes with opening soundness, relying on techniques used in the PKENO context. These constructions of group signature with opening soundness do not fall into the category of generic construction in the exact sense. Providing such a generic construction is one of the interesting research topics.

In Chap. 4, we showed that it is essentially unavoidable to use PKENO for constructing a group signature scheme with opening soundness. It was showed by following our approach (I), that is, we showed that any group signature scheme with opening soundness can be generically transformed to a secure PKENO scheme.

In Chap. 5, we showed that any secure PKENO scheme can be transformed to a robust threshold encryption scheme. The other direction, a generic construction of PKENO from robust threshold encryption, was already shown by Galindo et al.[GLF$^+$10], but that direction was not clearly and rigorously stated ever. This result establishes an equivalence between PKENO and threshold encryption in a rigorous sense (although an informal similarity was already observed by several authors), and presents useful suggestion for constructing practical PKENO and threshold encryption schemes.

In Chap. 6, we extended the relationship between group signature and public-key encryption to *revocable* group signature. Following the viewpoint (II), we obtained a new extension of public-key encryption, which corresponds to revocable group signature, called restrictive public-key encryption. Furthermore, we also presented an efficient construction of restrictive public-key encryption scheme, using techniques of revocable group signature. Unfortunately, as in Chap. 3, this construction is not a generic construction in a rigorous sense. It is another interesting open problem for investigating possibility of a generic construction of restrictive public-key encryption from revocable group signature.

This research contributes, by the aforementioned results, to clarifying the relationship between various sophisticated cryptographic primitives, namely, PKENO, group signature, threshold encryption, revocable group signature, and restrictive public-key encryption, and to providing useful guidelines for constructing efficient schemes of these primitives. We can expect the same approach is promising for clarifying relationships of more other primitives and for providing guidelines to construct efficient schemes of these primitives.

References

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010, Proceedings*, Vol. 5978 of *Lecture Notes in Computer Science*, pp. 480–497. Springer Berlin Heidelberg, 2010.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, Vol. 1880 of *Lecture Notes in Computer Science*, pp. 255–270. Springer Berlin Heidelberg, 2000.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, Vol. 4116 of *Lecture Notes in Computer Science*, pp. 111–125. Springer Berlin Heidelberg, 2006.

[AT99] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In Matthew Franklin, editor, *Financial Cryptography, Third International Conference, FC'99, Anguilla, British West Indies, February 22-25, 1999, Proceedings*, Vol. 1648 of *Lecture Notes in Computer Science*, pp. 196–211. Springer Berlin Heidelberg, 1999.

[AT09] Seiko Arita and Koji Tsurudome. Construction of threshold public-key encryptions through tag-based encryptions. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009, Proceedings*, Vol. 5536 of *Lecture Notes in Computer Science*, pp. 186–200. Springer Berlin Heidelberg, 2009.

[AW04] Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *Information and Communications Security, 6th International Conference, ICICS 2004, Malaga, Spain, October 27-29, 2004, Proceedings*, Vol. 3269 of *Lecture Notes in Computer Science*, pp. 1–13. Springer Berlin Heidelberg, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, Vol. 21, No. 2, pp. 149–177, April 2008.

[BBH06] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, Vol. 3860 of *Lecture Notes in Computer Science*, pp. 226–243. Springer Berlin Heidelberg, 2006.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, Vol. 3152 of *Lecture Notes in Computer Science*, pp. 41–55. Springer Berlin Heidelberg, 2004.

[BCN$^+$10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010, Proceedings*, Vol. 6280 of *Lecture Notes in Computer Science*, pp.

381–398. Springer Berlin Heidelberg, 2010.

[BCPZ08] Julien Bringer, Hervé Chabanne, David Pointcheval, and Sébastien Zimmer. An application of the boneh and shacham group signature scheme to biometric authentication. In Kanta Matsuura and Eiichiro Fujisaki, editors, *Advances in Information and Computer Security, Third International Workshop on Security, IWSEC 2008, Kagawa, Japan, November 25-27, 2008, Proceedings*, Vol. 5312 of *Lecture Notes in Computer Science*, pp. 219–230. Springer Berlin Heidelberg, 2008.

[BD09] Mihir Bellare and Shanshan Duan. Partial signatures and their applications. Cryptology ePrint Archive, Report 2009/336, 2009. `http://eprint.iacr.org/`.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, Vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522. Springer Berlin Heidelberg, 2004.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, Vol. 3378 of *Lecture Notes in Computer Science*, pp. 325–341. Springer Berlin Heidelberg, 2005.

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, Vol. 3376 of *Lecture Notes in Computer Science*, pp. 87–103. Springer Berlin Heidelberg, 2005.

[BKP11] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011, 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011, Proceedings*, Vol. 7073 of *Lecture Notes in Computer Science*, pp. 590–609. Springer Berlin Heidelberg, 2011.

[BMG07] Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual Symposium on Foundations of Computer Science, 20-23 October, 2007, Providence, RI, USA*, pp. 680–688, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, Vol. 2656 of *Lecture Notes in Computer Science*, pp. 614–629. Springer Berlin Heidelberg, 2003.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, May 2-4, 1988*, pp. 1–10, New York, NY, USA, 1988. ACM.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceedings*, Vol. 1807 of *Lecture Notes in Computer Science*, pp. 431–444. Springer Berlin Heidelberg, 2000.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, Vol. 3376 of *Lecture*

Notes in Computer Science, pp. 136–153. Springer Berlin Heidelberg, 2005.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 – June 1, 2006, Proceedings*, Vol. 4004 of *Lecture Notes in Computer Science*, pp. 427–444. Springer Berlin Heidelberg, 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, Vol. 4450 of *Lecture Notes in Computer Science*, pp. 1–15. Springer Berlin Heidelberg, 2007.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, Vol. 1976 of *Lecture Notes in Computer Science*, pp. 331–345. Springer Berlin Heidelberg, 2000.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, Vol. 839 of *Lecture Notes in Computer Science*, pp. 174–187. Springer Berlin Heidelberg, 1994.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack (extended abstract). In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99, Interna-*

tional Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceedings, Vol. 1592 of *Lecture Notes in Computer Science*, pp. 90–106. Springer Berlin Heidelberg, 1999.

[CG05] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, Vol. 3352 of *Lecture Notes in Computer Science*, pp. 120–133. Springer Berlin Heidelberg, 2005.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pp. 209–218, New York, NY, USA, 1998. ACM.

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 – June 1, 2006, Proceedings*, Vol. 4004 of *Lecture Notes in Computer Science*, pp. 1–11. Springer Berlin Heidelberg, 2006.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, Vol. 3027 of *Lecture Notes in Computer Science*, pp. 207–222. Springer Berlin Heidelberg, 2004.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002, Revised Papers*, Vol. 2576 of *Lecture Notes in Computer Science*, pp. 268–289. Springer Berlin Heidelberg, 2003.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, Vol. 3152 of *Lecture Notes in Computer Science*, pp. 56–72. Springer Berlin Heidelberg, 2004.

[CM98] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency (extended abstract). In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT '98, International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, Vol. 1514 of *Lecture Notes in Computer Science*, pp. 160–174. Springer Berlin Heidelberg, 1998.

[CP95] L. Chen and T.P. Pedersen. New group signature schemes (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, Vol. 950 of *Lecture Notes in Computer Science*, pp. 171–181. Springer Berlin Heidelberg, 1995.

[Cra96] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI and University of Amsterdam, November 1996.

[CS03a] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, Vol. 2729 of *Lecture Notes in Computer Science*, pp. 126–144. Springer Berlin Heidelberg, 2003.

[CS03b] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, Vol. 33, No. 1, pp. 167–226, 2003.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991,* *Proceedings*, Vol. 547 of *Lecture Notes in Computer Science*, pp. 257–265. Springer Berlin Heidelberg, 1991.

[Dam] Ivan Damgård. On Σ-protocol. Cryptologic Protocol Theory, CPT 2010, v.2, `http://www.daimi.au.dk/~ivan/Sigma.pdf`.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89, Proceedings*, Vol. 435 of *Lecture Notes in Computer Science*, pp. 307–315. Springer New York, 1990.

[DHKT08] Ivan Damgård, Dennis Hofheinz, Eike Kiltz, and Rune Thorbek. Public-key encryption with non-interactive opening. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008, Proceedings*, Vol. 4964 of *Lecture Notes in Computer Science*, pp. 239–255. Springer Berlin Heidelberg, 2008.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, Vol. 3378 of *Lecture Notes in Computer Science*, pp. 188–209. Springer Berlin Heidelberg, 2005.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology – VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, Vol. 4341 of *Lecture Notes in Computer Science*, pp. 193–210. Springer Berlin Heidelberg, 2006.

[DP08] Cécile Delerablée and David Pointcheval. Dynamic threshold public-key encryption. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008, Proceedings*, Vol. 5157 of *Lecture Notes in Computer Science*, pp. 317–334. Springer Berlin Heidelberg, 2008.

[DSDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pp. 522–533, New York, NY, USA, 1994. ACM.

[EHS10] Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. Group signature implies PKE with non-interactive opening and threshold PKE. In Isao Echizen, Noboru Kunihiro, and Ryoichi Sasaki, editors, *Advances in Information and Computer Security, 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22-24, 2010, Proceedings*, Vol. 6434 of *Lecture Notes in Computer Science*, pp. 181–198. Springer Berlin Heidelberg, 2010.

[EHSS13] Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Jacob C. N. Schuldt. Group signature implies public-key encryption with non-interactive opening. *International Journal of Information Security*, June 2013. Online first.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In Colin Boyd and Juan M. González Nieto, editors, *Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005, Proceedings*, Vol. 3574 of *Lecture Notes in Computer Science*, pp. 455–467. Springer Berlin Heidelberg, 2005.

[FI06] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. *IEICE Trans. Fundamentals.*, Vol. E89-A, No. 5, pp. 1328–1338, May 2006.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, October 22-24, 1990, Volume I, St. Louis, Missouri, USA*, pp. 308–317, Los Alamitos, CA, USA, 1990. IEEE Computer Society.

[FP09] Georg Fuchsbauer and David Pointcheval. Proofs on encrypted values in bilinear groups and an application to anonymity of signatures. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography – Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceed-*

ings, Vol. 5671 of *Lecture Notes in Computer Science*, pp. 132–149. Springer Berlin Heidelberg, 2009.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86, Proceedings*, Vol. 263 of *Lecture Notes in Computer Science*, pp. 186–194. Springer Berlin Heidelberg, 1987.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pp. 416–426, New York, NY, USA, 1990. ACM.

[Gal09] David Galindo. Breaking and repairing damgård et al. public key encryption scheme with non-interactive opening. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009, Proceedings*, Vol. 5473 of *Lecture Notes in Computer Science*, pp. 389–398. Springer Berlin Heidelberg, 2009.

[GLF$^+$10] David Galindo, Benoît Libert, Marc Fischlin, Georg Fuchsbauer, Anja Lehmann, Mark Manulis, and Dominique Schröder. Public-key encryption with non-interactive opening: New constructions and stronger definitions. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology – AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010, Proceedings*, Vol. 6055 of *Lecture Notes in Computer Science*, pp. 333–350. Springer Berlin Heidelberg, 2010.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, Vol. 28, No. 2, pp. 270–299, April 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, Vol. 17, No. 2, pp. 281–308, April 1988.

[GNMP$^+$12] Juan Manuel González Nieto, Mark Manulis, Bertram Poettering, Jothi Rangasamy, and Douglas Stebila. Publicly verifiable ciphertexts. In Ivan Visconti and De Roberto Prisco, editors, *Security and Cryptography for Networks, 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012, Proceedings*, Vol. 7485 of *Lecture Notes in Computer Science*, pp. 393–410. Springer Berlin Heidelberg, 2012.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, Vol. 59, No. 3, pp. 11:1–11:35, June 2012.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, Vol. 4284 of *Lecture Notes in Computer Science*, pp. 444–459. Springer Berlin Heidelberg, 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, Vol. 4833 of *Lecture Notes in Computer Science*, pp. 164–180. Springer Berlin Heidelberg, 2007.

[Gro10] Jens Groth. Fully anonymous group signatures without random oracles. Manuscript, May 17, 2010. `http://www.cs.ucl.ac.uk/staff/J.Groth/CertiSignFull.pdf`.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of*

Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008, Proceedings, Vol. 4965 of *Lecture Notes in Computer Science*, pp. 415–432. Springer Berlin Heidelberg, 2008.

[IMS$^+$06] Toshiyuki Isshiki, Kengo Mori, Kazue Sako, Isamu Teranishi, and Shoko Yonezawa. Using group signatures for identity management and its implementation. In *Proceedings of the Second ACM Workshop on Digital Identity Management, Alexandria, VA, USA, November 3, 2006*, pp. 73–78, New York, NY, USA, 2006. ACM.

[ISN93] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *Journal of Cryptology*, Vol. 6, No. 1, pp. 15–20, March 1993.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, Vol. 3876 of *Lecture Notes in Computer Science*, pp. 581–600. Springer Berlin Heidelberg, 2006.

[KP98] Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, Vol. 1462, pp. 169–185. Springer Berlin Heidelberg, 1998.

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, Vol. 3027 of *Lecture Notes in Computer Science*, pp. 571–589. Springer Berlin Heidelberg, 2004.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, Vol.

3494 of *Lecture Notes in Computer Science*, pp. 198–214. Springer Berlin Heidelberg, 2005.

[LDLK10] Junzuo Lai, Robert H. Deng, Shengli Liu, and Weidong Kou. Efficient CCA-secure PKE from identity-based techniques. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010, Proceedings*, Vol. 5985 of *Lecture Notes in Computer Science*, pp. 132–147. Springer Berlin Heidelberg, 2010.

[MDSMP91] Blum Manuel, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal on Computing*, Vol. 20, No. 6, pp. 1084–1118, December 1991.

[MRY04] Philip MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, Vol. 2951 of *Lecture Notes in Computer Science*, pp. 171–190. Springer Berlin Heidelberg, 2004.

[Ms09] Steven Myers and abhi shelat. Bit encryption is complete. In *50th Annual Symposium on Foundations of Computer Science, 25-27 October, 2009, Atlanta, Georgia, USA*, pp. 607–616, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[NFHF10] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. *IEICE Trans. Fundamentals.*, Vol. E93-A, No. 1, pp. 50–62, 2010.

[NS03] Toru Nakanishi and Yuji Sugiyama. An efficient anonymous survey for attribute statistics using a group signature scheme with attribute tracing. *IEICE Trans. Fundamentals.*, Vol. E86-A, No. 10, pp. 2560–2568, October 2003.

[NSA$^+$12] Koji Nuida, Kana Shimizu, Hiromi Arai, Michiaki Hamada, Koji Tsuda, Takatsugu Hirokawa, Goichiro Hanaoka, Jun Sakuma, and Kiyoshi Asai. Privacy-preserving database search protocol for chemical compounds with additive-homomorphic encryption (in Japanese). In *Proc. of Computer Security Symposium 2012*, October 2012. To appear.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pp. 427–437, New York, NY, USA, 1990. ACM.

[OFHO09] Go Ohtake, Arisa Fujii, Goichiro Hanaoka, and Kazuto Ogawa. On the theoretical gap between group signatures with and without unlinkability. In Bart Preneel, editor, *Progress in Cryptology – AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009, Proceedings*, Vol. 5580 of *Lecture Notes in Computer Science*, pp. 149–166. Springer Berlin Heidelberg, 2009.

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008, Proceedings*, Vol. 5209 of *Lecture Notes in Computer Science*, pp. 57–74. Springer Berlin Heidelberg, 2008.

[PKO10] Le Trieu Phong, Kaoru Kurosawa, and Wakaha Ogata. Provably secure convertible undeniable signatures with unambiguity. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010, Proceedings*, Vol. 6280 of *Lecture Notes in Computer Science*, pp. 291–308. Springer Berlin Heidelberg, 2010.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pp. 387–394, New York, NY, USA, 1990. ACM.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPT0 '91, Proceedings*, Vol. 576 of *Lecture Notes in Computer Science*, pp. 433–444. Springer Berlin Heidelberg, 1992.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, 17-18 October, 1999, New York, NY, USA*, pp. 543–553, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[SG02] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, Vol. 15, No. 2, pp. 75–96, January 2002.

[SHI$^+$12] Yumi Sakemi, Goichiro Hanaoka, Tetsuya Izu, Masahiko Takenaka, and Masaya Yasuda. Solving a discrete logarithm problem with auxiliary input on a 160-bit elliptic curve. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012, 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012, Proceedings*, Vol. 7293 of *Lecture Notes in Computer Science*, pp. 595–608. Springer Berlin Heidelberg, 2012.

[Sho04] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[SSE$^+$12a] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012, 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012, Proceedings*, Vol. 7293 of *Lecture Notes in Computer Science*, pp. 715–732. Springer Berlin Heidelberg, 2012.

[SSE$^+$12b] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and

Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. Cryptology ePrint Archive, 2012. `http://eprint.iacr.org/`.

[TFS09] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-times anonymous authentication. *IEICE Trans. Fundamentals.*, Vol. E92-A, No. 1, pp. 147–165, January 2009.

[TV09] Stephen R. Tate and Roopa Vishwanathan. Improving cut-and-choose in verifiable encryption and fair exchange protocols using trusted computing technology. In Ehud Gudes and Jaideep Vaidya, editors, *Data and Applications Security XXIII 23rd Annual IFIP WG 11.3 Working Conference, Montreal, Canada, July 12-15, 2009, Proceedings*, Vol. 5645 of *Lecture Notes in Computer Science*, pp. 252–267. Springer Berlin Heidelberg, 2009.

[ZHSI04] Rui Zhang, Goichiro Hanaoka, Junji Shikata, and Hideki Imai. On the security of multiple encryption or CCA-security+CCA-security=CCA-security? In Feng Bao, Robert Deng, and Jianying Zhou, editors, *Public Key Cryptography – PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004, Proceedings*, Vol. 2947 of *Lecture Notes in Computer Science*, pp. 360–374. Springer Berlin Heidelberg, 2004.

[ZL06] Sujing Zhou and Dongdai Lin. Shorter verifier-local revocation group signatures from bilinear maps. In David Pointcheval, Yi Mu, and Kefei Chen, editors, *Cryptology and Network Security, 5th International Conference, CANS 2006, Suzhou, China, December 8-10, 2006, Proceedings*, Vol. 4301 of *Lecture Notes in Computer Science*, pp. 126–143. Springer Berlin Heidelberg, 2006.

Acknowledgments

First and foremost, I would like to express my great acknowledgment to the supervisor Prof. Kazuo Ohta. His advice makes me learn a lot of things, including a desirable attitude toward research activities. I am also grateful to Prof. Kazuo Sakiyama and Associate Prof. Mitsugu Iwamoto. They had many interesting discussions with me, which is always helpful for improving the research result.

I am also grateful to the members of Ohta-lab. and Sakiyama-lab. I had a lot of discussions with them on cryptography and on other things. Such discussions sometimes brought me interesting ideas, and sometimes provided nice refreshment.

I would like to thank many researchers of RISEC, AIST and other organizations. In particular, Goichiro Hanaoka, Jacob Schuldt, and Keita Emura have co-authored many papers with me. Their comments and suggestions are always insightful and helpful, and make me achieve better results, which are included in this dissertation.

Finally I would like to thank my family for their various kinds of support.

List of Publications Related and Referred to the Dissertation

Related Publications

Journal Papers

[1] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, and Kazumasa Omote. Methods for restricting message space in public-key encryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E96-A, No. 6, pp. 1156–1168, June 2013.

Refereed Conference Papers (with Formal Proceedings)

[2] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012, 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21–23, 2012*, Vol. 7293 of *Lecture Notes in Computer Science*, pp. 715–732. Springer, 2012. Full version available at `http://eprint.iacr.org/2012/431`.

Referred Publications

Journal Papers

[3] Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Jacob C. N. Schuldt. Group signature implies public-key encryption with non-interactive opening. *International Journal of Information Security*, Vol. 13, No. 1, pp. 51–62, February 2014.

Refereed Conference Papers (with Formal Proceedings)

[4] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, and Kazumasa Omote. Towards restricting plaintext space in public key encryption. In Tetsu Iwata and Masakatsu Nishigaki, editors, *Advances in Information and Computer Security, 6th International Workshop, IWSEC 2011, Tokyo, Japan, November 8–10, 2011*, Vol. 7038 of *Lecture Notes in Computer Science*, pp. 193–209. Springer, 2011.

[5] Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. Group signature implies PKE with non-interactive opening and threshold PKE. In Isao Echizen, Noboru Kunihiro, and Ryoichi Sasaki, editors, *Advances in Information and Computer Security, 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22–24, 2010*, Vol. 6434 of *Lecture Notes in Computer Science*, pp. 181–198. Springer, 2010.

Non-refereed Papers

[6] 坂井祐介, 江村恵太, Jacob C. N. Schuldt, 花岡悟一郎, 太田和夫. 非対話開示機能付き公開鍵暗号からの頑健な閾値暗号の一般的構成. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[7] 坂井祐介, 江村恵太, Jacob Schuldt, 花岡悟一郎, 太田和夫. ある弱いモデルの上でロバストな閾値暗号の一般的構成. 2013 年 暗号と情報セキュリティシンポジウム (SCIS2013), January 2013.

List of All Publications

Journal Papers

[1] Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Jacob C. N. Schuldt. Group signature implies public-key encryption with non-interactive opening. *International Journal of Information Security*, Vol. 13, No. 1, pp. 51–62, February 2014.

[2] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, and Kazumasa Omote. Methods for restricting message space in public-key encryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E96-A, No. 6, pp. 1156–1168, June 2013.

[3] Mitsuhiro Hattori, Takato Hirano, Takashi Ito, Nori Matsuda, Takumi Mori, Yusuke Sakai, and Kazuo Ohta. Ciphertext-policy delegatable hidden vector encryption and its application. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E96-A, No. 1, pp. 53–67, January 2013.

[4] Yutaka Kawai, Yusuke Sakai, and Noboru Kunihiro. On the (im)possibility results for strong attack models for public key cryptsystems. *Journal of Internet Services and Information Security*, Vol. 1, No. 2/3, pp. 125–139, August 2011.

[5] Yusuke Sakai, Goichiro Hanaoka, Kaoru Kurosawa, and Kazuo Ohta. How to shorten a ciphertext of reproducible key encapsulation mechanisms in the random oracle model. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E94-A, No. 6, pp. 1293–1305, June 2011.

Refereed Conference Papers (with Formal Proceedings)

[6] Kazuma Ohara, Yusuke Sakai, Keita Emura, and Goichiro Hanaoka. A group signature scheme with unbounded message-dependent opening. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIA CCS'13, Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China, May 8–10, 2013*, pp. 517–522, New York, NY, USA, May 2013. ACM.

[7] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012, 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21–23, 2012*, Vol. 7293 of *Lecture Notes in Computer Science*, pp. 715–732. Springer, 2012. Full version available at `http://eprint.iacr.org/2012/431`.

[8] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, Takahiro Matsuda, and Kazumasa Omote. Group signatures with message-dependent opening. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography – Pairing 2012, 5th International Conference, Cologne, Germany, May 16–18, 2012*, Vol. 7708 of *Lecture Notes in Computer Science*, pp. 270–294. Springer, 2013.

[9] Mitsuhiro Hattori, Takato Hirano, Takashi Ito, Nori Matsuda, Takumi Mori, Yusuke Sakai, and Kazuo Ohta. Ciphertext-policy delegatable hidden vector encryption and its application to searchable encryption in multi-user setting. In Liqun Chen, editor, *Cryptography and Coding, 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12–15, 2011*, Vol. 7089 of *Lecture Notes in Computer Science*, pp. 190–209. Springer, 2011.

[10] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, and Kazumasa Omote. Towards restricting plaintext space in public key encryption. In Tetsu Iwata and

Masakatsu Nishigaki, editors, *Advances in Information and Computer Security, 6th International Workshop, IWSEC 2011, Tokyo, Japan, November 8–10, 2011*, Vol. 7038 of *Lecture Notes in Computer Science*, pp. 193–209. Springer, 2011.

[11] Yusuke Sakai, Goichiro Hanaoka, Kaoru Kurosawa, and Kazuo Ohta. A generic method for reducing ciphertext length of reproducible KEMs in the RO model. In Isao Echizen, Noboru Kunihiro, and Ryoichi Sasaki, editors, *Advances in Information and Computer Security, 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22–24, 2010*, Vol. 6434 of *Lecture Notes in Computer Science*, pp. 55–69. Springer, 2010.

[12] Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. Group signature implies PKE with non-interactive opening and threshold PKE. In Isao Echizen, Noboru Kunihiro, and Ryoichi Sasaki, editors, *Advances in Information and Computer Security, 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22–24, 2010*, Vol. 6434 of *Lecture Notes in Computer Science*, pp. 181–198. Springer, 2010.

Refereed Conference Papers (with No Formal Proceedings)

[13] Keita Emura, Goichiro Hanaoka, Akihisa Kodate, Sanami Nakagawa, and Yusuke Sakai. Secure single sign-on schemes constructed from nominative signatures, revisited. In *IWSEC 2013 (Poster Session), Okinawa, Japan, November 18–20, 2013*, 2012.

[14] Kazuma Ohara, Yusuke Sakai, Mitsugu Iwamoto, and Kazuo Ohta. A t-resilient unconditionally secure first-price auction protocol. In *IWSEC 2012 (Poster Session), Fukuoka, Japan, November 7–9, 2012*, 2012.

Non-refereed Papers

[15] 平野貴人, 伊藤隆, 川合豊, 服部充洋, 松田規, 太田和夫, 坂井祐介. 暗号化索引追加後の Searchable Symmetric Encryption の安全性の関係. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[16] 坂井祐介, 江村恵太, Jacob C. N. Schuldt, 花岡悟一郎, 太田和夫. 非対話開示機能付き公開鍵暗号からの頑健な閾値暗号の一般的構成. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[17] 土屋喬文, 徳重佑樹, 坂井祐介, 岩本貢, 太田和夫. 同時実行攻撃に耐性を持つシンプルな Secret Handshake. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[18] 中川紗菜美, 江村恵太, 坂井祐介, 花岡悟一郎, 小舘亮之. 証明者限定署名による個人情報漏洩を考慮したアクセスログ管理. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[19] 大原一真, 坂井祐介, 江村恵太, 花岡悟一郎, 太田和夫. 署名長の短い削除機能付きグループ署名. 2014 年 暗号と情報セキュリティシンポジウム (SCIS2014), January 2014.

[20] 岩本貢, 大原一真, 坂井祐介, 太田和夫. t 人結託耐性を有する第一価格オークションプロトコルに対する情報理論的解析. 2013 年 暗号と情報セキュリティシンポジウム (SCIS2013), January 2013.

[21] 坂井祐介, 江村恵太, Jacob Schuldt, 花岡悟一郎, 太田和夫. ある弱いモデルの上でロバストな閾値暗号の一般的構成. 2013 年 暗号と情報セキュリティシンポジウム (SCIS2013), January 2013.

[22] 大原一真, 坂井祐介, 岩本貢, 太田和夫. 情報理論的に安全な first-price オークションプロトコル. 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[23] 坂井祐介, Jacob Schuldt, 江村恵太, 花岡悟一郎, 太田和夫. グループ署名の安全性について. 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[24] 平野貴人, 森拓海, 服部充洋, 伊藤隆, 松田規, 川合豊, 坂井祐介, 太田和夫. Searchable symmetric encryption のドキュメント追加後の安全性について. 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[25] 松田和也, 川合豊, 坂井祐介, 太田和夫. より強い再暗号化鍵匿名性を満たすプロキシ再暗号化方式. 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[26] 服部充洋, 平野貴人, 森拓海, 伊藤隆, 松田規, 川合豊, 坂井祐介, 太田和夫. 安全性と高

速性の両立を目指した検索可能暗号 (2). 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[27] 松田規, 服部充洋, 平野貴人, 森拓海, 伊藤隆, 川合豊, 坂井祐介, 太田和夫. 安全性と高速性の両立を目指した検索可能暗号 (1). 2012 年 暗号と情報セキュリティシンポジウム (SCIS2012), January 2012.

[28] 伊藤隆, 服部充洋, 松田規, 坂井祐介, 太田和夫. 頻度分析耐性を持つ高速秘匿検索方式. 電子情報通信学会 技術研究報告, ISEC2010-72, pp. 1–6, March 2011.

[29] 川合豊, 坂井祐介, 國廣昇. 公開鍵暗号方式における強い攻撃モデルの証明 (不) 可能性. 電子情報通信学会 技術研究報告, ISEC2010-132, pp. 395–402, March 2011.

[30] 服部充洋, 平野貴人, 伊藤隆, 松田規, 森拓海, 坂井祐介, 太田和夫. Searchable public-key encryption for hierarchical systems with adaptive join/leave of members. 2011 年 暗号と情報セキュリティシンポジウム (SCIS2011), January 2011.

[31] 坂井祐介, 岩本貢, 駒野雄一, 太田和夫. FDH 署名の安全性証明の再考. 2011 年 暗号と情報セキュリティシンポジウム (SCIS2011), January 2011.

[32] 江村恵太, 花岡悟一郎, 川合豊, 面和成, 坂井祐介. 公開鍵暗号における平文空間の制限の実現に向けて. 2011 年 暗号と情報セキュリティシンポジウム (SCIS2011), January 2011.

[33] 江村恵太, 花岡悟一郎, 川合豊, 松田隆宏, 面和成, 坂井祐介. メッセージ依存開示可能グループ署名と匿名掲示板への応用. 2011 年 暗号と情報セキュリティシンポジウム (SCIS2011), January 2011.

[34] 坂井祐介, 花岡悟一郎, 黒澤馨, 太田和夫. Reproducible KEM の暗号文長を削減するランダムオラクルモデルにおける一般的な変換方式. 2010 年 暗号と情報セキュリティシンポジウム (SCIS2010), January 2010.

[35] 松田和也, 坂井祐介, 太田和夫, 崎山一男. Katz らの leakage resilient *t*-time 署名の解析. 2010 年 暗号と情報セキュリティシンポジウム (SCIS2010), January 2010.

[36] 坂井祐介, バグスサントソ, 奥秋清次, 太田和夫. ブルームフィルタの設計論とその APOP パスワード復元攻撃検知への応用. 2009 年 暗号と情報セキュリティシンポジウム (SCIS2009), January 2009.

[37] 奥秋清次, 坂井祐介, 太田和夫, 國廣昇. APOP 攻撃検知法の提案とシミュレーション

による検証: 目には目を, 歯には歯を. 2009 年 暗号と情報セキュリティシンポジウム (SCIS2009), January 2009.

[38] 北原恵介, 坂井祐介, 太田和夫. より効率的な秘匿情報検索. 2009 年 暗号と情報セキュリティシンポジウム (SCIS2009), January 2009.

Author Biography

Yusuke Sakai was born in Niigata, Japan in 1987. He received his B.E. degree from The University of Electro-Communications, Tokyo, Japan, in 2009 and received his M.E. degree from The University of Electro-Communications, Tokyo, Japan, in 2011. He is a doctor course student in The University of Electro-Communications, Tokyo, Japan. He is presently engaged in research on cryptography. He received The SCIS Paper Prize from IEICE in 2011 and the Best Student Paper Award in IWSEC 2010.