論文 / 著書情報
Article / Book Information

題目(和文)	アドホックグループ上の匿名認証
Title(English)	Anonymous Authentication over Ad-Hoc Groups
著者(和文)	原啓祐
Author(English)	Keisuke Hara
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第11712号, 授与年月日:2022年3月26日, 学位の種別:課程博士, 審査員:田中 圭介,伊東 利哉,尾形 わかは,鹿島 亮,安永 憲司
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第11712号, Conferred date:2022/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Anonymous Authentication over Ad-Hoc Groups
（アドホックグループ上の匿名認証）

Keisuke Hara
Supervisor: Keisuke Tanaka

Department of Mathematical and Computing Science
School of Computing
Tokyo Institute of Technology

Acknowledgement

First of all, I am deeply grateful to my supervisor Professor Keisuke Tanaka. He gave me a lot of advices and encouragements through the daily activities in his laboratory. Throughout these six years, his advice was great not only for my research but also for my life. Also, he gave me a lot of opportunities to experience important things in my life.

I would like to thank the referees of my doctoral thesis, Prof. Toshiya Itoh, Prof. Wakaha Ogata, Prof. Ryo Kashima, and Prof. Kenji Yasunaga for taking their time to reading this thesis and to listening to my presentations. Their feedbacks helped me to improve my work.

I also would like to thank current members and ex-members of Tanaka laboratory. I spent delightful time with them in daily life. Especially, I thank to my colleagues, Yusuke Yoshida and Masayuki Tezuka, who shared much time and developed through friendly competition with me. His kind and great help was crucial for completing my research.

I am also grateful to Goichiro Hanaoka who is a prime senior researcher at Cyber Physical Security Research Center (CPSEC) of National Institute of Advanced Industrial Science and Technology (AIST) and the members of "Shin-Akarui-Angou-Benkyou-Kai". They gives me grateful comments and suggestions for my research. I have a lot of experience in the discussion with them. Expressly, I would deeply like to thank Shuichi Katsumata, Takahiro Matsuda, Yusuke Sakai, Jacob Schuldt, and Shota Yamada. Their comments and advices are indispensable for my thesis. Without their guidance and persistent help, this thesis would not be done.

Last but not least, I would also like to thank my father Kazuo Hara and my mother Noriko Hara for their grateful support and encouragements. Without their warm help, this thesis would not have been possible and I would not decide to go to the doctoral course after completing the master's course.

Contents

Chapter 1

Introduction

1.1 Background

In recent years, people are increasingly sending out information on the Internet, for example, through the social networking services. When we communicate with others through the Internet, we need to make sure that the counter party is the right person and the communication is authenticated. For realizing this, an authentication protocol is one of the most important cryptographic primitives in an Internet-based communication. By using an authentication protocol, a receiver of a message, Bob, can verify that the message received is indeed the one sent by the sender, Alice.

While it is important to consider the authenticity for users, it is also required to ensure the opposite properties, that is, *anonymity* for users in some real-world applications. In particular, users' anonymity on the Internet is one of the most important requirements for protecting the freedom of personal speech. In fact, in order to meet the demand for anonymity, some technologies providing users with (unlimited) anonymity on the Internet (such as, the Tor network) have been developed.

In order to solve this dilemma between these two properties (authenticity and anonymity), various anonymous authentication primitives have been proposed so far (e.g., group signature [Cv91], attribute-based signature [MPR11], direct anonymous attestation [BCC04], and so on). Among these primitives, in this thesis, we focus on two anonymous authentication primitives *over ad-hoc groups* of users: *ring signature* and *(deniable) ring authentication*.

In the following, we look back on these two primitives briefly.

1.1.1 Ring Signature

Rivest, Shamir, and Tauman-Kalai [RST01] introduced a novel concept of signature with anonymity, which is called *ring signature*. Ring signature allows a user to sign messages as a member of a set of users R, which is called a *ring*. The required security properties for ring signature are unforgeability and anonymity. Firstly, unforgeability ensures that an adversary cannot forge a signature on behalf of an honest ring of signers. Secondly, anonymity ensures that any signature does not reveal any information of the signer's identity. Combining these two properties, we can ensure that all users can verify the signature is made by someone in the ring but cannot detect that which member in the ring sign the message.

When ring signature has been introduced in [RST01], the primary motivation was whistle-blowing, where a party can anonymously leak a secret by using a ring including members who know the secret information. In addition to this motivation, ring signature was recently made into practice used by one of the largest cryptocurrency: Monero. Monero is a cryptocurrency launched in April 2014 to provide users appropriate privacy by using ring signature. Due to the anonymity of ring signature, nobody can know which of the users in the ring spent the coin.

Compared to the other major anonymous authentication primitives (such as, group signature and attribute-based signature), ring signature has some attractive features. The first one is that we do not need trusted key managers and users can generate their keys by themselves. The second one is that a user can choose its ring in an ad-hoc manner. Thus, it can set the size of its ring adaptively (that is, the anonymity level of its ring signature). Due to these features, it is the most desirable for us to construct a ring signature scheme in the *plain* model. Here, the plain model is the model that we do not need any trusted setup assumption (e.g., the existence of some trusted common reference string (CRS)) and any heuristic technique (e.g., the random oracle (RO) methodology [BR93]). In general, since it is difficult to construct an efficient ring signature scheme in the plain model, a lot of works have been proposed various ring signature schemes in the RO model or the CRS model so far. In the following, we introduce some major ring signature schemes in the RO model, the CRS model, and the plain model.

The Schemes in the RO Model. The concept of ring signature was introduced by Rivest, Shamir, and Tauman-Kalai [RST01]. They proposed a construction based on trapdoor permutations. Then, a number of ring signature schemes provided in the RO model based on the various computational assumptions [AOS02, BGLS03, HS03]. Dodis et al. [DKNS04] firstly

proposed a constant-size ring signature scheme based on the RSA accumulators [CL02]. Groth and Kohlweiss [GK15] proposed a logarithmic-size ring signature scheme by combining Σ protocols and the Fiat-Shamir heuristic [FS87]. They succeeded to prove the security of this scheme under the standard computational assumptions in the RO model. However, their security reduction was very loose due to using the Forking Lemma [PS96]. Based on this result, Libert et al. [LPQ18] gave the first tightly-secure ring signature scheme with logarithmic signature size under the DDH assumption in the RO model. The number of users and the number of oracle queries do not affect their reduction cost at all. Recently, Derler, Ramacher, and Slamanig [DRS18] proposed the first cryptographic accumulator based sorely on the symmetric-key primitives and by using this new primitive, showed how to construct a logarithmic-size ring signature scheme sorely from symmetric-key primitives.

The Schemes in the CRS Model. Ring signature in the CRS model has been also studied well so far. Shacham and Waters [SW07] proposed the first scheme based on the composite order groups with bilinear maps. Then, Schäge and Schwenk [SS10] proposed more efficient construction based on the computational Diffie-Hellman (CDH) assumption over the pairing-friendly group, but their scheme achieves only a weaker notion of unforgeability (against the chosen subring attacks in the terminology of [BKM06]). However, the signature size of these constructions grows linearly with respect to the size of a ring. Then, Chandran, Groth, and Sahai [CGS07] proposed the first sub-linear-size ring signature scheme based on the composite order groups with bilinear maps. Moreover, Gonzalez [Gon19] recently improved the signature size of their construction.

The Schemes in the Plain Model. Bender et al. [BKM06] introduced rigorous and desirable security notions for ring signature and provide a construction satisfying the security properties in the plain model. Their construction is based on a PKE scheme, and a signature scheme, and a ZAP argument with computational privacy [DN00]. Then, Chow et al. [CWLY06] proposed a ring signature scheme with unconditional anonymity in the plain model under the tailored assumption over the pairing group. However, their scheme supports only rings of constant size. Recently, the direction for constructing ring signature schemes in the plain model has been extensively studied. Malavolta et al. [MS17] proposed a ring signature scheme in the plain model based on the variant of the Diffie-Hellman knowledge assumption. Then, Backes, Hanzlik, Kluczniak, and Schneider [BHKS18] provided a ring signature scheme in the plain model with sub-linear size signatures from a new cryptographic primitive: *signatures with flexible public key*. Moreover, Backes et al. [BDH+19] recently show

the breakthrough result which provided the first ring signature scheme in the plain model with logarithmic-size signatures from standard assumptions over bilinear groups. Very recently, Chatterjee, Garg, Hajiabadi, Khurana, Liang, Malavolta, Pandey, and Shiehian [CGH+21] proposed the first (post-quantum) logarithmic-sized ring signature scheme in the plain model based (solely) on the learning with errors (LWE) assumption over lattices.

1.1.2 (Deniable) Ring Authentication

One of the natural ways for authentication is to utilize a (digital) signature scheme in a communication. In this case, the authentication of a message is Alice's signature generated by her secret signing key. The unforgeability of the signature scheme ensures the soundness of the authentication. Additionally, this approach has a *non-repudation* property since the signature can be verified by anyone using Alice's public verification key. In other words, once Alice signs the message, she is bound to it. Then, everybody can know that she signed it. For contracts, for example, this property is useful since conditions must be enforced in case of dispute.

Here, however, we have a natural question: Is there any case in which the non-repudiation property is not desirable? In some cases, the non-repudiation property could raise serious privacy issues. For example, consider a situation that Alice wants to tell something privately to Bob, in a way that Bob believes it comes from her, but also in a way that Bob cannot convince a third party that Alice told anything. (In other words, Alice wants to communicate with Bob in an "Off-The-Record" manner.) Clearly, a signature scheme is not a proper primitive for this situation.

Due to the above problem, deniable authentication [DNS98, Kat03, Pas03, DG05, DGK06] has always been a central concern in personal and business communications. In general, we say that an authentication protocol provides deniability if the receiver of a message could have generated all the transcripts in the communication by itself. On the practical side, for example, in the internet key exchange (IKE) protocol [IKE] or in the recent Signal protocol [Signal], (an appropriate form of) deniability is identified as a desirable property.

Deniable Ring Authentication. In deniable authentication, while Alice can deny her participation in a communication to a third party, she is not anonymous to Bob. In some cases, we need to consider a situation that Alice wants to hide her identity and Bob just needs to confirm that Alice is a legitimate person in some group (e.g., a board member of some company) without knowing which one. To address this problem while preserving deniability

simultaneously in an authentication, Naor [Nao02] proposed a cryptographic primitive called *deniable ring authentication*. Deniable ring authentication enables a prover in some group (called a *ring*) to authenticate a message to a verifier using its secret key while at the same time allowing the prover to deny ever having interacted with the verifier. This primitive furthermore guarantees the anonymity of the prover in the sense that the verifier will learn nothing about the identity of the prover except that it is included in the ring. This property is called *source hiding*. Note that source hiding is a security notion against a (malicious) verifier, while deniability is one against any third party. As observed in [ZCTH17], one of the novel applications of deniable ring authentication is a privacy-preserving protocol for a location-based service (LBS) in a vehicular ad-hoc network (VANET).

1.2 Our Contribution

In this thesis, we give some new results for ring signature in the plain model and deniable ring authentication.

1.2.1 Ring Signature with Unconditional Anonymity in the Plain Model

As the first result, we propose the first generic construction of ring signature with unconditional anonymity in the plain model based on the standard assumption. Our scheme is constructed based on a statistical ZAP argument, a lossy encryption scheme, and a standard signature scheme. It is known that all of our building blocks can be instantiated based on the quasi-polynomial LWE assumption.

The anonymity for ring signature can be classified into two types: *computational anonymity* and *unconditional anonymity*. Computational anonymity states that the anonymity holds only against a computationally bounded adversary. In many cases, the computational anonymity is ensured under some mathematical computational hard problem (e.g., the factorization problem, the discrete logarithm problem, and the computational Diffie-Hellman problem). Here, if there exists an adversary who can solve an underlying hard problem efficiently, then the computational anonymity no longer holds. Conversely, unconditional anonymity states that the anonymity holds even against a computationally unbounded adversary.

In the applications of ring signature, we can suppose that some systems are used for a long time. In such cases, if we adopt a ring signature scheme with computational anonymity, we should predict that its anonymity for users in the systems can be broken in the future

due to recent developments of algorithms and computer technologies. Thus, from a practical point of view, we can say that unconditional anonymity is more desirable than computational anonymity.

So far, a lot of ring signature schemes with unconditional anonymity has been proposed in the common reference string (CRS) model and in the random oracle model (*e.g.*, [DKNS04, DRS18, CGS07, Gon19, GK15, LPQ18, MS17, RST01]). On the other hand, in the plain model, we have only one scheme proposed by Malavolta and Schröder [MS17].[1] However, their scheme is only secure under the variant of the Diffie-Hellman knowledge assumption, which is one of the non-falsifiable assumptions. That is, we did not have any ring signature scheme with unconditional anonymity in the plain model based on the standard falsifiable assumption so far.

1.2.2 Tightly Secure Ring Signature in the Plain Model

Next, as the second result, we propose a generic constructions of tightly secure ring signature in the plain model. Through our generic construction, we obtain the first tightly secure and logarithmic-sized ring signature scheme in the plain model under the DLIN assumption over bilinear groups. In the following, we explain the motivation for this result.

Tight Security. In general, when we prove the security of a cryptographic primitive, we usually construct a reduction algorithm $\mathcal{B}$ who try to solve an underlying problem assumed to be hard by using an algorithm $\mathcal{A}$ who attacks the security of the cryptographic primitive. We say that the cryptographic primitive is secure if the probability $p_{\mathcal{A}}$ that $\mathcal{A}$ succeeds to break the security is as small as the probability $p_{\mathcal{B}}$ that $\mathcal{B}$ can solve the hard problem. Ideally, both of the probabilities should be the same, that is, $p_{\mathcal{A}} = p_{\mathcal{B}}$ should be held. However, it often occurs that the reductions suffer from a loss in the success probability. That is, we have $p_A = L \cdot p_{\mathcal{B}}$, where $L = L(\lambda)$, which is called a *reduction cost*, is some polynomial in the

[1] In fact, Malavolta et al. [MS17] proposed a ring signature scheme with unconditional anonymity in the plain model under the some standard computational assumptions over the pairing group. However, this scheme only achieves basic and weak anonymity considered in [BKM06], which is not preferable for the real-world applications. Furthermore, Chow, Wei, Liu, and Yuen [CWLY06] proposed a ring signature scheme with unconditional anonymity in the plain model under the tailored assumption over the pairing group. However, their scheme supports only rings of constant size. In this thesis, we focus on ring signature satisfying the strongest and standard security properties considered in [BKM06] and supporting a-priori unbounded polynomial-sized rings.

security parameter λ.[1] In the setting of ring signatures, we consider as the crucial reduction cost L, the number of all users $N := N(\lambda)$ (in the system) and the numbers of oracle queries made by an adversary (e.g., random oracle queries $Q_H := Q_H(\lambda)$ and signing oracle queries $Q_{sig} := Q_{sig}(\lambda)$). In some cases, the reduction cost L depends on N, Q_H, Q_{sig} linearly. (That is, we have $L = N \cdot Q_H \cdot Q_{sig}$.) Asymptotically, even if we have the reduction cost L, we can prove that a cryptographic primitive is secure. However, when we consider its exact security in a realistic setting, we have the following problem. Here, if we want to obtain 128-bit security for the resulting ring signature scheme (that is, we require that $p_{\mathcal{A}} = 2^{-128}$ holds), and say we had $p_{\mathcal{A}} = 2^{-128}$, $N = 2^{14}$, $Q_H = 2^{30}$, and $Q_{sig} = 2^{20}$ as the realistic setting, we would have to require at least 192-bit security for the underlying problem (since $p_{\mathcal{B}} = \frac{p_{\mathcal{A}}}{N \cdot Q_H \cdot Q_{sig}} = 2^{-192}$ holds), which incurs a significant blowup of the parameters. Thus, it is essential to care about the reduction cost L because smaller L ensures a better security level for a fixed security parameter. We say that a cryptographic primitive is *tightly secure* if the reduction cost is constant, *i.e.*, $L = \mathcal{O}(1)$.

Tight Security for Ring Signatures. Recently, Libert, Peters, and Qian [LPQ18] proposed the first tightly secure ring signature scheme in the random oracle (RO) model [BR93] based on the decisional Diffie-Hellman (DDH) assumption. The signature size of their scheme depends on the size of a ring only logarithmically. Thanks to the random oracle methodology, Libert et al.'s scheme [LPQ18] is highly efficient and practical. However, since the results from several papers, such as [CGH98], this methodology has been well known to be open to some debate.

So far, it is well studied to provide tight security proofs for standard cryptographic primitives in the standard model (e.g., standard signature schemes [Sch11, HJ12, ADK+13, CW13, BKP14, LJYP14, Hof17, AHN+17, JOR18, GHKP18] and public-key encryption schemes [HJ12, LJYP14, LPJY15, Hof17]). However, in a context of ring signatures, there exists no scheme given a tight security proof in the standard model.

1.2.3 Round-Optimal Deniable Ring Authentication

Finally, as the third result, we propose a new generic construction of two-round concurrently deniable ring authentication in the random oracle model. Our generic construction is based

[1] Formally, when estimating a loss of a success probability, we also have to consider a runtime of the reduction algorithm $\mathcal{B}$. However, in order to simplify an explanation, we omit a security loss by a runtime here.

on any IND-CPA secure broadcast encryption (BE) scheme. Instantiating the underlying IND-CPA secure BE scheme with the schemes proposed by Agrawal and Yamada [AY20] or Agrawal, Wichs, and Yamada [AWY20], we obtain the first two-round concurrently deniable ring authentication scheme with optimal efficiency in an asymptotic sense. Here, by optimal efficiency, we mean that all of the sizes of a public parameter and secret keys, the communication costs, and the number of pairing operations are independent of n, where n is the number of users in a ring. In addition to these main instantiations, through our generic construction, we further obtain various two-round concurrently deniable ring authentication schemes. In the following, we provide the motivation for this result.

We have the technical challenges for deniable ring authentication from two perspectives: the security requirement and the efficiency requirement. In terms of the security requirement, one of the important challenges is to achieve concurrent deniability [DNS98]. Intuitively, concurrent deniability ensures that a (malicious) verifier of a message cannot convince a third party that the corresponding prover sends the message even if the verifier can open and schedule sessions in an arbitrary way. This is a natural desirable requirement for an Internet-based communication.

Next, in terms of the efficiency requirement, we should estimate the efficiency of deniable ring authentication with four aspects: *round complexity*, *parameter size*, *communication complexity*, and *computational complexity*. Regarding round complexity, clearly, it is infeasible to achieve soundness and deniability simultaneously in the non-interactive (that is, one round) setting. Therefore, an optimal scheme is one having only *two rounds*.[1] Then, on the aspects of parameter size, communication complexity, and computational complexity, one important measure is whether they depend on the number of users in a ring. Here, let n be the number of users in a ring. All of these complexity measures naturally depend on n linearly (that is, $\mathcal{O}(n)$) when we consider a naive construction. Regarding the parameter size, we have to take care of the sizes of a public parameter and a secret key. Finally, regarding the computational complexity, *a pairing operation* is one of the heavy computations for schemes based on bilinear maps. (In fact, most of the previous schemes [DHIN11, YAS+12, ZMYH17, ZCTH17] are constructed over bilinear groups.) Thus, we would like its number to be as small as possible.

From the above observations, it is desirable to construct a *two-round concurrently* deniable ring authentication scheme with *optimal efficiency*. Here, optimal efficiency we target in this

[1] We note that some previous works (e.g., [SM04b, SM04a]) proposed non-interactive deniable ring authentication schemes with *partial* deniability. Partial deniability only ensures that a user in the authentication protocol can deny the contents of its communications. That is, it cannot deny its involvement in the authentication protocol. In this work, we focus only on deniability in the sense of [DGK06].

thesis means that the public parameter size, the secret key size, communication complexity, and the number of pairing operations are all independent of n.[2]

1.3 Organization

The rest of this thesis is organized as follows. In Chapter 2, we review some notations and the models and security requirements of cryptographic primitives used in this thesis (signature, public key encryption, lossy encryption, broadcast encryption, ZAP, non-interactive proof system in the plain model, collision-resistant hash function, and somewhere perfectly binding hash function with private local opening).

In Chapter 3, we review the model of ring signature and (deniable) ring authentication. Firstly, the syntax and security requirements (unforgeability and anonymity) of ring signature are described in Section 3.1. Secondly, we give the syntax and security requirements (soundness, source hiding, and deniability) of deniable ring authentication in Section 3.2.

In Chapter 4, we provide our ring signature scheme with unconditional anonymity in the plain model. Firstly, we give the technical overview of this construction in Section 4.1. Then, in Section 4.2, we describe the formal description of our scheme. Lastly, in Section 4.3, we provide the security proofs for unforgeability and anonymity of our scheme.

In Chapter 5, we provide our tightly secure and logarithmic-sized ring signature scheme in the plain model. Similar to Chapter 4, we give the technical overview of this construction in Section 5.1, describe the formal description of our scheme in Section 5.2, and provide the security proofs for unforgeability and anonymity of our scheme in Section 5.3.

In Chapter 6, we present our generic transformation from a BE scheme to a DRA scheme in the RO model. Similar to Chapter 4 and 5, we give the technical overview of this construction in Section 6.2. In this chapter, as an intermediate cryptographic primitive for obtaining our round-optimal DRA scheme, we introduce a new security notion for BE, which is called *plaintext awareness*, in Section 6.3. More precisely, we provide the definition of plaintext awareness for BE in Section 6.3.1 and show that a BE scheme with plaintext awareness and IND-CPA security satisfies IND-CCA security in Section 6.3.2. Then, in Section 6.3.3, we propose our BE scheme with plaintext awareness and IND-CPA security in the RO model. Next, we formally describe our DRA scheme in Section 6.4 and give its security proofs in Section 6.5. Lastly, in

[2] The public parameter size, the secret key size, and communication complexity of our schemes actually have $\mathsf{poly}(\log n)$ factors. Here, however, we ignore them since $\mathsf{poly}(\log n)$ factors are asymptotically absorbed into $\mathsf{poly}(\lambda)$ factors, where λ is a security parameter.

Section 6.6, to better understand our scheme, we present a simple and efficient instantiation of our generic construction of round-optimal deniable ring authentication.

Finally, in Chapter 7, we conclude this thesis and leave future works.

Chapter 2

Preliminaries

In this section, we define notations and cryptographic primitives used in Section 4, 5, and 6.

2.1 Notations

In this thesis, $x \leftarrow X$ denotes sampling an element x from a finite set X uniformly at random. $y \leftarrow \mathcal{A}(x; r)$ denotes that a probabilistic algorithm $\mathcal{A}$ outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathcal{A}(x)$ when we need not write an internal randomness explicitly. For interactive Turing machines $\mathcal{A}$ and $\mathcal{B}$, $v \leftarrow \langle \mathcal{A}(x_a), \mathcal{B}(x_b) \rangle (y)$ denotes that $\mathcal{B}$ outputs v at the end of an execution of an interactive protocol between $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ take x_a and x_b as a private input respectively and y denotes the common input for both $\mathcal{A}$ and $\mathcal{B}$. When $\mathcal{A}$ (resp., $\mathcal{B}$) does not take private inputs, we simply denote $v \leftarrow \langle \mathcal{A}, \mathcal{B}(x_b) \rangle (y)$ (resp., $v \leftarrow \langle \mathcal{A}(x_a), \mathcal{B} \rangle (y)$). For strings x and y, $x \| y$ denotes the concatenation of x and y. Also, $x := y$ denotes that x is defined by y. λ denotes a security parameter. A function $f(\lambda)$ is a negligible function in λ, if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. $\mathsf{negl}(\lambda)$ denotes an unspecified negligible function. poly denotes an unspecified polynomial. We use $\mathbf{Time}(\mathcal{A})$ to denote the running time of an algorithm $\mathcal{A}$. PPT stands for probabilistic polynomial time. $\emptyset$ denotes an empty set. If n is a natural number, $[n]$ denotes the set of integers $\{1, \cdots, n\}$. Also, if a and b are integers such that $a \leq b$, $[a, b]$ denotes the set of integers $\{a, \cdots, b\}$.

2.2 Signature

A signature scheme with a message space $\mathcal{M}$ consists of a tuple of the following three PPT algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$.

Gen: The key generation algorithm, given a security parameter 1^λ, outputs a verification key vk and a signing key sk.

Sign: The signing algorithm, given a signing key sk and a message $m \in \mathcal{M}$, outputs a signature σ.

Ver: The (deterministic) verification algorithm, given a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, outputs either 1 (meaning "accept") or 0 (meaning "reject").

As the correctness for SIG, we require that $\mathsf{Ver}(vk, m, \mathsf{Sign}(sk, m)) = 1$ holds for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, and $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.

Next, we define existential unforgeability under chosen-message attacks in the multi-user setting with corruptions (MU-EUF-CMA$^{\mathrm{Corr}}$ security) for a signature scheme.

Definition 1 (MU-EUF-CMA$^{\mathrm{Corr}}$ Security)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of verification keys n.*

1. *$\mathcal{C}$ generates $(vk_i, sk_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for all $i \in [n]$. Then, $\mathcal{C}$ gives a set of the verification keys $\mathbf{vk} := (vk_1, \cdots, vk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_i := \emptyset$ for all $i \in [n]$.*
2. *$\mathcal{A}$ is allowed to make a signing query of the form (j, m), where $j \in [n]$ and $m \in \mathcal{M}$ is a message. When $\mathcal{C}$ receives (j, m), $\mathcal{C}$ computes $\sigma \leftarrow \mathsf{Sign}(sk_j, m)$, gives the signature σ to $\mathcal{A}$, and appends (m, σ) to S_j. Moreover, $\mathcal{A}$ is allowed to make a corruption query of the form $j \in [n]$. When $\mathcal{C}$ receives j, $\mathcal{C}$ returns sk_j to $\mathcal{A}$ and appends j to $\mathsf{S}_{\mathsf{Corr}}$.*
3. *$\mathcal{A}$ outputs a tuple (i^*, m^*, σ^*).*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr[(1 = \mathsf{Ver}(vk_{i^*}, m^*, \sigma^*)) \wedge ((m^*, \cdot) \notin \mathsf{S}_{i^*}) \wedge (i^* \notin \mathsf{S}_{\mathsf{Corr}})].$$

We say that SIG satisfies MU-EUF-CMA^{Corr} security if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

2.3 Public Key Encryption

A public key encryption (PKE) scheme with a plaintext space $\mathcal{M}$ consists of a tuple of the following three PPT algorithms $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

- KG: The key generation algorithm, given a security parameter 1^λ, outputs a public key pk and a decryption key dk.
- Enc: The encryption algorithm, given a public key pk and a plaintext $m \in \mathcal{M}$, outputs a ciphertext c.
- Dec: The (deterministic) decryption algorithm, given a public key pk, a decryption key dk, and a ciphertext c, outputs a plaintext $m \in \{\bot\} \cup \mathcal{M}$.

For the correctness of Π, we require that $\mathsf{Dec}(pk, dk, \mathsf{Enc}(pk, m)) = m$ holds for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, and $(pk, dk) \leftarrow \mathsf{KG}(1^\lambda)$.

Next, we define the security notions for a PKE scheme: *pseudorandomness of public keys* and *pseudorandomness of ciphertexts*.

Definition 2 (Pseudorandomness of public keys)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of public keys n. In the following, let $\mathcal{PK}$ be a space of public keys of Π.*

1. *$\mathcal{C}$ chooses a challenge bit $b \leftarrow \{0,1\}$ and generates $(pk_{i0}, dk_{i0}) \leftarrow \mathsf{KG}(1^\lambda)$ and $pk_{i1} \leftarrow \mathcal{PK}$ for all $i \in [n]$. Then, $\mathcal{C}$ sets $pk_i := pk_{ib}$ for all $i \in [n]$ and gives a set of the public keys $\mathbf{pk} := (pk_1, \cdots, pk_n)$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{key}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that Π satisfies pseudorandomness of public keys if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{key}}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

Definition 3 (Pseudorandomness of ciphertexts)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of public keys n. In the following, let $\mathcal{CT}$ the (whole) ciphertext space of Π.*

1. *$\mathcal{C}$ chooses a challenge bit $b \leftarrow \{0,1\}$ and samples a public key $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{key} for all $i \in [n]$. Then, $\mathcal{C}$ gives $(r_i^{\mathsf{key}})_{i\in[n]}$ to $\mathcal{A}$.*

2. *$\mathcal{A}$ can make queries of the form $(j, m) \in [n] \times \mathcal{M}$ to $\mathcal{C}$. If $b = 0$ holds, $\mathcal{C}$ generates a challenge ciphertext $c \leftarrow \mathsf{Enc}(pk_j, m)$. Otherwise, $\mathcal{C}$ samples a ciphertext $c \leftarrow \mathcal{CT}$. Then, $\mathcal{C}$ gives the ciphertext c to $\mathcal{A}$.*

3. *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{ct}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that Π satisfies pseudorandomness of ciphertexts if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ct}}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

Remark 2.1. *We note that the ElGamal encryption scheme [ElG84] is a PKE scheme satisfying the both of the above properties. Remarkably, the pseudorandomness of public keys is ensured information-theoretically and the pseudorandomness of ciphertexts can be tightly reduced to the DDH assumption by using the self-reducibility.*

2.4 Lossy Encryption

A lossy encryption scheme with a plaintext space $\mathcal{M}$ consists of a tuple of the following four PPT algorithms $\mathsf{LE} = (\mathsf{KG}, \mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$.

KG: The ordinary key generation algorithm, given a security parameter 1^λ, outputs an ordinary encryption key ek and a decryption key dk.

LKG: The lossy key generation algorithm, given a security parameter 1^λ, outputs a lossy encryption key ek.

Enc: The encryption algorithm, given an encryption key ek and a plaintext $m \in \mathcal{M}$, outputs a ciphertext c.

Dec: The (deterministic) decryption algorithm, given an encryption key ek, a decryption key dk, and a ciphertext c, outputs a plaintext $m \in \{\perp\} \cup \mathcal{M}$.

As the correctness for LE, we require that $\mathsf{Dec}(ek, dk, \mathsf{Enc}(ek, m)) = m$ holds for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, and $(ek, dk) \leftarrow \mathsf{KG}(1^\lambda)$.

Next, we define the security notions for a lossy encryption scheme: *indistinguishability of ordinary/lossy keys* and *statistical lossiness*.

Definition 4 (Indistinguishability of ordinary/lossy keys)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of encryption keys n.*

1. *$\mathcal{C}$ chooses a challenge bit $b \leftarrow \{0,1\}$ and generates $(ek_{i0}, dk_{i0}) \leftarrow \mathsf{KG}(1^\lambda)$ and $ek_{i1} \leftarrow \mathsf{LKG}(1^\lambda)$ for all $i \in [n]$. Then, $\mathcal{C}$ sets $ek_i := ek_{ib}$ for all $i \in [n]$ and gives a set of the encryption keys $\mathbf{ek} := (ek_1, \cdots, ek_n)$ to $\mathcal{A}$.*

2. *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{A}}(\lambda) := 2 \cdot \left|\Pr[b = b'] - \frac{1}{2}\right|$. We say that LE satisfies indistinguishability of ordinary/lossy keys if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

Remark 2.2. *The above definition of the indistinguishability of ordinary/lossy keys is different from the original one proposed in [BHY09] in the sense that we consider a multi-user setting. We note that our definition and the original one proposed in [BHY09] are equivalent except for a polynomial reduction loss based on the number of users.*

Definition 5 (Statistical lossiness)**.** *Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of encryption keys n. In the following, we let $\mathcal{R}^{\mathsf{LKG}}$ be the randomness space for the lossy key generation algorithm LKG and $\mathcal{CT}$ the (whole) ciphertext space of LE.*

1. *For all $i \in [n]$, $\mathcal{C}$ samples randomnesses $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{\mathsf{LKG}}$ and generates lossy encrytion keys $ek_i \leftarrow \mathsf{LKG}(1^\lambda; r_i^{\mathsf{LKG}})$. Then, $\mathcal{C}$ chooses a challenge bit $b \leftarrow \{0,1\}$ and gives $(r_i^{\mathsf{LKG}})_{i \in [n]}$ to $\mathcal{A}$.*

2. *$\mathcal{A}$ is allowed to make a challenge query of the form (j, m) to $\mathcal{C}$. Next, if $b = 0$ holds, $\mathcal{C}$ generates a challenge ciphertext $c \leftarrow \mathsf{Enc}(ek_j, m)$. Otherwise, $\mathcal{C}$ samples a challenge ciphertext $c \leftarrow \mathcal{CT}$. Then, $\mathcal{C}$ gives the challenge ciphertext c to $\mathcal{A}$.*

3. *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{A}}(\lambda) := 2 \cdot \left|\Pr[b = b'] - \frac{1}{2}\right|$. *We say that* LE *satisfies statistical lossiness if for any computationally unbounded adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ *holds.*

Remark 2.3. *The above definition of the statistical lossiness is slightly different from the original one proposed by [BHY09] in the following two sense.*

1. *We consider a multi-user setting rather than a single-user setting.*
2. *We consider an indistinguishability between* $c \leftarrow \mathsf{Enc}(ek_i, m)$ *and* $c \leftarrow \mathcal{CT}$ *instead of one between* $c \leftarrow \mathsf{Enc}(ek_j, m_0)$ *and* $c \leftarrow \mathsf{Enc}(ek_j, m_1)$, *where* $j \in [n]$ *and* $m_0, m_1 \in \mathcal{M}$ *are chosen by* $\mathcal{A}$ *given a lossy encryption key* $(ek_i \leftarrow \mathsf{LKG}(1^\lambda))_{i\in[n]}$.

We note that some constructions based on the learning with errors (LWE) assumption proposed in the previous works [BHY09, HLL+15] satisfies the above security definition.

2.5 Broadcast Encryption

In this section, we review the basic definitions for BE. A BE scheme consists of the following three PPT algorithms $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$.

Setup: The setup algorithm, given a security parameter 1^λ and the number of users n, outputs a public parameter pp and a set of secret keys $\mathbf{sk} := (sk_i)_{i\in[n]}$.

Enc: The encryption algorithm, given a public parameter pp, an index set S, and a plaintext m, outputs a ciphertext c.

Dec: The (deterministic) decryption algorithm, given a public parameter pp, an index set S, a secret key sk_i, and a ciphertext c, outputs a plaintext m, which could be the special symbol $\perp$.

As the correctness for BE, we require that $\mathsf{Dec}(pp, \mathsf{S}, sk_i, c) = m$ hold for all $\lambda \in \mathbb{N}$, $n = \mathsf{poly}(\lambda)$, m, $\mathsf{S} \subseteq [n]$, $i \in \mathsf{S}$, $(pp, \mathbf{sk} = (sk_1, \cdots, sk_n)) \leftarrow \mathsf{Setup}(1^\lambda, n)$, and $c \leftarrow \mathsf{Enc}(pp, \mathsf{S}, m)$.

Decryption Uniqueness. We give the definition of decryption uniqueness. Intuitively, decryption uniqueness is a property ensuring that it is impossible to generate an invalid ciphertext which is decrypted to different plaintexts among users in an authorized set. (In the previous work [HK08], the same property was called *verifiability*.)

Definition 6 (Decryption Uniqueness)**.** *We say that a BE scheme* $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies decryption uniqueness if for any* $\lambda \in \mathbb{N}$*,* $n = \mathsf{poly}(\lambda)$*,* $(pp, \mathbf{sk} = (sk_1, \cdots, sk_n)) \leftarrow \mathsf{Setup}(1^\lambda, n)$*,* $\mathsf{S}^* \subseteq [n]$*, and* c^**, there are no distinct indices* $i, j \in \mathsf{S}^*$ *such that* $\mathsf{Dec}(pp, \mathsf{S}^*, sk_i, c^*) \neq \mathsf{Dec}(pp, \mathsf{S}^*, sk_j, c^*)$ *holds.*

Smoothness. We provide the definition of smoothness for BE, which is a natural BE-analogue of smoothness defined for PKE by Bellare, Hofheinz, and Kiltz [BHK15]. Informally, we say that a BE scheme satisfies smoothness if the number of possible ciphertexts is super-polynomially large for any plaintext. We note that many known BE schemes secure in the sense of indistinguishability (such as, BE schemes used in the instantiations of our deniable ring authentication scheme) have smoothness unconditionally. Moreover, it is easy to convert any BE scheme to one satisfying this property (say, attaching a randomness to a ciphertext). Its formal definition is as follows.

Definition 7 (Smoothness)**.** *Let* $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *be a BE scheme. For any* $\lambda \in \mathbb{N}$*, we define* Smth *as*

$$\mathsf{Smth}(\lambda) := \mathop{\mathbf{E}}_{(pp,\mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, n)} \left[\max_{m, \mathsf{S}, c'} \Pr_{c \leftarrow \mathsf{Enc}(pp, \mathsf{S}, m)} [c = c'] \right].$$

We say that BE *satisfies smoothness if we have* $\mathsf{Smth}(\lambda) = \mathsf{negl}(\lambda)$*.*

IND-CCA Security. We give the definition of IND-CCA security for BE.

Definition 8 (IND-CCA Security)**.** *Let* $n := n(\lambda)$ *be a polynomial which denotes the number of users. Consider the following experiment for a (stateful) adversary* $\mathcal{A}$*.*

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda)$:
- $\mathsf{S}_{\mathsf{Corr}} := \emptyset$
- $(pp, \mathbf{sk} = (sk_i)_{i \in [n]}) \leftarrow \mathsf{Setup}(1^\lambda, n)$
- $(\mathsf{S}^*, m_0^*, m_1^*) \leftarrow \mathcal{A}^{\mathcal{DO}, \mathcal{CO}}(pp)$
- $b \leftarrow \{0, 1\}$
- $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{S}^*, m_b^*)$
- $b' \leftarrow \mathcal{A}^{\mathcal{DO}, \mathcal{CO}}(c^*)$
- *If* $b = b'$ *then return* 1 *else return* 0*.*

In the above experiment, we require that $|m_0^*| = |m_1^*|$ *and* $\mathsf{S}^* \subseteq [n] \backslash \mathsf{S}_{\mathsf{Corr}}$ *hold. Moreover, the decryption oracle* $\mathcal{DO}$ *and the corruption oracle* $\mathcal{CO}$ *are defined as follows:*

Decryption oracle. *When $\mathcal{A}$ accesses to the decryption oracle $\mathcal{DO}$ by making a query (i, S, c), $\mathcal{DO}$ computes $m \leftarrow \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ and returns m to $\mathcal{A}$. After the challenge, $\mathcal{A}$ is not allowed to make a query (i, S, c) such that $c = c^*$.*

Corruption oracle. *When $\mathcal{A}$ accesses to the corruption oracle $\mathcal{CO}$ by making a query i, $\mathcal{CO}$ returns sk_i to $\mathcal{A}$ and appends i to $\mathsf{S}_{\mathsf{Corr}}$. After the challenge, $\mathcal{A}$ is not allowed to make a query i such that $i \in \mathsf{S}^*$.*

We say that BE *satisfies IND-CCA security if for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

holds. IND-CPA security of BE is defined analogously, except that $\mathcal{A}$ is not given access to the decryption oracle $\mathcal{DO}$. We denote the IND-CPA advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{A}}(\lambda)$.

Remark 2.4. *In the above definition of IND-CCA (IND-CPA) security, if all algorithms of* BE *and $\mathcal{A}$ are given access to a random oracle, we say that* BE *satisfies IND-CCA (IND-CPA) security in the RO model.*

2.6 ZAP

Let $\mathcal{R}$ be an efficiently computable binary relation and $\mathcal{L} := \{x \mid \exists w \,\text{s.t.}\, (x, w) \in \mathcal{R}\}$. A ZAP argument for $\mathcal{L}$ consists of a tuple of the following two PPT algorithms $\mathsf{ZAP} = (\mathsf{Prove}, \mathsf{Verify})$ associated with a parameter $\ell := \ell(\lambda)$, where $\ell(\lambda)$ is some polynomial in λ.

Prove: The proving algorithm, given a string $r \in \{0,1\}^\ell$, a statement $x \in \mathcal{L}$, and a witness w for the fact that $x \in \mathcal{L}$, outputs a proof π.

Verify: The verification algorithm, given a string $r \in \{0,1\}^\ell$, a statement x, and a proof π, outputs either 1 (meaning "accept") or 0 (meaning "reject").

As the correctness for ZAP, we require that $\mathsf{Verify}(r, x, \mathsf{Prove}(r, x, w)) = 1$ holds for all $\lambda \in \mathbb{N}$, all $r \leftarrow \{0,1\}^\ell$, all statements $x \in \mathcal{L}$, and all witnesses w for the fact that $x \in \mathcal{L}$.

Next, we define the security notions for a ZAP argument: *computational soundness* and *statistical witness indistinguishability.*

Definition 9 (Computational Soundness)**.** *We say that a ZAP argument* ZAP = (Prove, Verify) *satisfies computational soundness if for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{A}}(\lambda) := \Pr[r \leftarrow \{0,1\}^\ell; (x^*, \pi^*) \leftarrow \mathcal{A}(r) : (x^* \notin \mathcal{L}) \wedge (\mathsf{Verify}(r, x^*, \pi^*) = 1)] = \mathsf{negl}(\lambda).$$

Definition 10 (Statistical Witness Indistinguishability)**.** *We say that a ZAP argument* $\mathsf{ZAP} = (\mathsf{Prove}, \mathsf{Verify})$ *satisfies witness indistinguishability if for any computationally unbounded adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{A}}(\lambda) := |\Pr[r \leftarrow \mathcal{A}(1^\lambda) : \mathcal{A}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(r) = 1] - \Pr[r \leftarrow \mathcal{A}(1^\lambda) : \mathcal{A}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(r) = 1]| = \mathsf{negl}(\lambda),$$

where $\mathcal{O}_b(\cdot,\cdot,\cdot)$ *is an oracle that takes* (x, w_0, w_1) *as input and answers* $\pi \leftarrow \mathsf{Prove}(r, x, w_b)$ *for* $b \in \{0,1\}$ *to* $\mathcal{A}$*, where* $(x, w_0), (x, w_1) \in \mathcal{R}$*.*

2.7 Non-interactive Proof System in the Plain Model

Let $\mathcal{R}$ be an efficiently computable binary relation and $\mathcal{L} := \{x \mid \exists w \,\text{s.t.}\, (x,w) \in \mathcal{R}\}$. A non-interactive proof system in the plain model for $\mathcal{L}$ consists of a tuple of the following two PPT algorithms $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$.

Prove: The proving algorithm, given a statement $x \in \mathcal{L}$ and a witness w for the fact that $x \in \mathcal{L}$, outputs a proof π.

Verify: The verification algorithm, given a statement x and a proof π, outputs either 1 (meaning "accept") or 0 (meaning "reject").

For the correctness of NIWI, we require that $\mathsf{Verify}(x, \mathsf{Prove}(x,w)) = 1$ holds for all $\lambda \in \mathbb{N}$, all statements $x \in \mathcal{L}$, and all witnesses w for the fact that $x \in \mathcal{L}$.

Next, we define the security notions for a non-interactive proof system in the plain model: *Computational witness indistinguishability* and *perfect soundness.*

Definition 11 (Computational Witness Indistinguishability)**.** *We say that a non-interactive proof system in the plain model* $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ *satisfies computational witness indistinguishability if for any PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = \mathsf{negl}(\lambda),$$

where $\mathcal{O}_b(\cdot,\cdot,\cdot)$ *is an oracle that takes* (x, w_0, w_1) *as input and answers* $\pi \leftarrow \mathsf{Prove}(x, w_b)$ *for* $b \in \{0,1\}$*, where* $(x, w_0), (x, w_1) \in \mathcal{R}$*.*

Definition 12 (Perfect Soundness)**.** *We say that a non-interactive proof system in the plain model* $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ *satisfies perfect soundness if for any computationally unbounded adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{A}}(\lambda) := \Pr[(x^*, \pi^*) \leftarrow \mathcal{A}(1^\lambda) : (x^* \notin \mathcal{L}) \wedge (\mathsf{Verify}(x^*, \pi^*) = 1)] = 0$$

holds.

From the previous work [GOS06], we can construct a non-interactive proof system in the plain model satisfying computational witness indistinguishability and perfect soundness based on the DLIN assumption over the bilinear group.

Remark 2.5 (Tightness of computational witness indistinguishability in [GOS06])**.** *The original definition of computational witness indistinguishability given in [GOS06] captures the situation that an adversary makes single challenge query. Then, they showed that the computational witness indistinguishability of their non-interactive proof system in the plain model holds under the DLIN assumption.*

Unlike their definition, in Definition 11, we consider the setting that an adversary $\mathcal{A}$ can make multiple challenge queries. We stress that even if we require that their scheme [GOS06] satisfies Definition 11, the computational witness indistinguishability of their scheme can be reduced to the DLIN assumption tightly (without depending on the number of challenge queries) by using the self-reducibility of the DLIN assumption.

2.8 Collision-Resistant Hash Function

Here, we recall the definition of a collision-resistant hash function. A hash function consists of a pair of PPT algorithms $\mathsf{CRHF} = (\mathsf{HKG}, \mathsf{Hash})$. HKG is the hash key generation algorithm that, given a security parameter 1^λ, outputs a hash key hk. Hash is the (deterministic) hashing algorithm that, given a hash key hk and a string $x \in \{0,1\}^*$, outputs a hash value h.

Definition 13 (Collision-resistance)**.** *We say that* $\mathsf{CRHF} = (\mathsf{HKG}, \mathsf{Hash})$ *is a collision-resistant hash function if for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{A}}(\lambda) := \Pr[hk \leftarrow \mathsf{HKG}(1^\lambda); (x, x^*) \leftarrow \mathcal{A}(hk) : \mathsf{Hash}(hk, x) = \mathsf{Hash}(hk, x^*) \wedge x \neq x^*] = \mathsf{negl}(\lambda).$$

2.9 Somewhere Perfectly Binding Hash Function with Private Local Opening

A somewhere perfectly binding hash function with private local opening consists of a tuple of the following four PPT algorithms $\Gamma = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{HVer})$.

HGen: The hashing/private key generation algorithm, given a security parameter 1^λ, a database size n, and an index i, outputs a hashing key hk and a private key shk.

Hash: The hashing algorithm, given a hashing key hk and a database $\mathcal{DB}$, outputs a digest h.

Open: The opening algorithm, given a hashing key hk, a private key shk, a database $\mathcal{DB}$, and an index i, outputs a witness τ.

HVer: The (deterministic) verification algorithm, given a hashing key hk, a digest h, an index i, a value x, and a witness τ, outputs either 1 (meaning "accept") or 0 (meaning "reject").

For the correctness of Π, for all $\lambda \in \mathbb{N}$, all $n = \mathsf{poly}(\lambda)$, all databases $\mathcal{DB} = \{x_i\}_{i\in[n]}$, and all indices $i \in [n]$, we require that $\mathsf{HVer}(hk, h, i, x_i, \tau) = 1$ holds, where $(hk, shk) \leftarrow \mathsf{HGen}(1^\lambda, n, i)$, $h \leftarrow \mathsf{Hash}(hk, \mathcal{DB})$, and $\tau \leftarrow \mathsf{Open}(hk, shk, \mathcal{DB}, i)$.

Next, we define the security notions for a somewhere perfectly binding hash function with private local opening: *somewhere perfectly binding* and *index hiding*.

Definition 14 (Somewhere perfectly binding)**.** *Let $n := n(\lambda)$ be a polynomial in λ. For all databases $\mathcal{DB} = \{x_i\}_{i\in[n]}$, all indices $i \in [n]$, all hashing keys hk, all values x, and all witnesses τ, we say that Γ satisfies somewhere perfectly binding if $h = \mathsf{Hash}(hk, \mathcal{DB})$ and $1 = \mathsf{HVer}(hk, h, i, x, \tau)$, then $x = x_i$ holds.*

Definition 15 (Index hiding)**.** *Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

1. *$\mathcal{A}$ sends a tuple (n, i_0, i_1) to $\mathcal{C}$.*
2. *$\mathcal{C}$ chooses a challenge bit $b \leftarrow \{0, 1\}$ and generates a pair of a hashing/private key $(hk, shk) \leftarrow \mathsf{HGen}(1^\lambda, n, i_b)$. Then, $\mathcal{C}$ gives the hashing key hk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{hide}}_{\Gamma,\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that Γ satisfies index hiding if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{hide}}_{\Gamma,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

Chapter 3

Formal Definitions of Ring Signature and Deniable Ring Authentication

In this chapter, we provide the formal definitions of ring signature and (deniable) ring authentication.

Firstly, the syntax and security requirements (unforgeability and anonymity) of ring signature are described in Section 3.1. More specifically, we introduce the most standard and strongest security notions: *unforgeability w.r.t. insider corruptions* and *anonymity under full key exposure*, which are proposed by Bender, Katz, and Morselli [BKM06]. Both of our ring signature schemes satisfy this security requirements.

Secondly, we give the syntax and security requirements (soundness, source hiding, and deniability) of deniable ring authentication in Section 3.2. The definitions of soundness and source hiding are based on the previous work by Yamada, Attrapadung, Santoso, Schuldt, Hanaoka, and Kunihiro [YAS+12]. Moreover, the definition of deniability is obtained by extending the definition of deniability for deniable authentication [DGK06] into the ring setting.

3.1 Ring Signature

In this section, we provide the syntax and security requirements of ring signature. A ring signature scheme with a message space $\mathcal{M}$ consists of a tuple of the following three PPT algorithms $\mathsf{RS} = (\mathsf{RGen}, \mathsf{RSign}, \mathsf{RVer})$.

RGen: The key generation algorithm, given a security parameter 1^λ, outputs a verification key rvk and a signing key rsk.

RSign: The signing algorithm, given a signing key rsk, a message $m \in \mathcal{M}$, and a ring R, outputs a signature σ.

RVer: The (deterministic) verification algorithm, given a ring R, a message $m \in \mathcal{M}$, and a signature σ, outputs either 1 (meaning "accept") or 0 (meaning "reject").

As the correctness for RS, we require that $\mathsf{RVer}(\mathsf{R}, m, \mathsf{RSign}(rsk, m, \mathsf{R})) = 1$ holds for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(rvk, rsk) \leftarrow \mathsf{RGen}(1^\lambda)$, and R such that $rvk \in \mathsf{R}$.

Next, we define anonymity and unforgeability for a ring signature scheme. We adopt the strongest security notions of anonymity and unforgeability proposed by Bender et al. [BKM06].

Definition 16 (Anonymity under full key exposure)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of verification keys n.*

1. *For all $i \in [n]$, $\mathcal{C}$ generates $(rvk_i, rsk_i) \leftarrow \mathsf{RGen}(1^\lambda; r_i)$, where r_i is a randomness for generating a pair of keys (rvk_i, rsk_i). Then, $\mathcal{C}$ gives a set of the randomnesses $(r_i)_{i\in[n]}$ to $\mathcal{A}$.*

2. *$\mathcal{A}$ requests a challenge to $\mathcal{C}$ by sending a tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, where i_0 and i_1 are indices such that $rvk_{i_0} \in \mathsf{R}^*$ and $rvk_{i_1} \in \mathsf{R}^*$. Then, $\mathcal{C}$ samples a challenge bit $b \leftarrow \{0,1\}$, computes $\sigma^* \leftarrow \mathsf{RSign}(rsk_{i_b}, m^*, \mathsf{R}^*)$, and gives σ^* to $\mathcal{A}$.*

3. *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) := 2 \cdot \left|\Pr[b = b'] - \frac{1}{2}\right|$. We say that RS satisfies unconditional anonymity under full key exposure if for any computationally unbounded adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds.

Definition 17 (Unforgeability w.r.t. insider corruptions)**.** *Let $n := n(\lambda)$ be a polynomial in λ. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of verification keys n.*

1. *$\mathcal{C}$ generates $(rvk_i, rsk_i) \leftarrow \mathsf{RGen}(1^\lambda)$ for all $i \in [n]$. Then, $\mathcal{C}$ gives a set of the verification keys $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.*

2. *$\mathcal{A}$ is allowed to make a signing query of the form (j, R, m), where $m \in \mathcal{M}$ is a message, R is a set of verification keys, and $j \in [n]$ is an index such that $rvk_j \in \mathsf{R}$. When $\mathcal{C}$ receives (j, R, m), $\mathcal{C}$ computes $\sigma \leftarrow \mathsf{RSign}(rsk_j, m, \mathsf{R})$, gives the signature σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$. Moreover, $\mathcal{A}$ is allowed to make a corruption query of the form $j \in [n]$. When $\mathcal{C}$ receives j, $\mathcal{C}$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.*

3. $\mathcal{A}$ *outputs a tuple* $(\mathsf{R}^*, m^*, \sigma^*)$.

In this game, we define the advantage of the adversary $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) := \Pr[(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})].$$

We say that RS *satisfies unforgeability w.r.t. insider corruptions if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ *holds.*

3.2 Deniable Ring Authentication

In this section, we review the definitions for deniable ring authentication in the RO model. Here, we assume that all algorithms introduced in the following can access to an RO. A deniable ring authentication scheme consists of a tuple of the following PPT algorithms $\mathsf{DRA} = (\mathsf{DRA.Setup}, \langle \mathsf{DRA.Prove}, \mathsf{DRA.Verify} \rangle)$.

$\mathsf{DRA.Setup}(1^\lambda, n)$**:** The setup algorithm, given a security parameter 1^λ and the number of users n, outputs a public parameter pp and n secret keys $\mathbf{sk} := (sk_1, \cdots, sk_n)$.

$\langle \mathsf{DRA.Prove}(sk), \mathsf{DRA.Verify} \rangle(pp, \mathsf{R}, m)$**:** This is an interactive protocol between the prover algorithm DRA.Prove and the verifier algorithm DRA.Verify, in which DRA.Prove takes sk as a secret input and both of the algorithms take a public parameter pp, a ring $\mathsf{R} \subseteq [n]$, and a message m as input. As a result of the interaction, DRA.Verify locally outputs a bit $v \in \{0, 1\}$.

As the correctness for DRA, we require that $1 = \langle \mathsf{DRA.Prove}(sk_i), \mathsf{DRA.Verify} \rangle(pp, \mathsf{R}, m)$ hold for all $\lambda \in \mathbb{N}$, $n = \mathsf{poly}(\lambda)$, $i \in [n]$, m, $(pp, \mathbf{sk}(= (sk_1, \cdots, sk_n))) \leftarrow \mathsf{DRA.Setup}(1^\lambda, n)$, and $\mathsf{R} \subseteq [n]$ such that $i \in \mathsf{R}$.

Remark 3.1 (On the setup algorithm DRA.Setup)**.** *Similar to the previous works [DHIN11, YAS+12], when generating a public parameter pp and n secret keys* $\mathbf{sk} := (sk_1, \cdots, sk_n)$ *in the setup algorithm* DRA.Setup*, we require that the maximum number of users in the system n is fixed at the setup phase (by a trusted third party). In other words, we do not consider the setting that new users dynamically join to this system.*

As mentioned in Section 6.1, some previous works [ZMYH17, ZCTH17] do not need such a setup procedure and a public/secret key of each user is generated by itself. Thus, in their schemes, the maximum number of users is not fixed and a new user can dynamically join the system.

Next, we define security properties for deniable ring authentication in the RO model: *concurrent soundness*, *source hiding*, and *concurrent deniability*. Although we propose a deniable ring authentication scheme in the RO model in Section 6.4, we omit an RO in the following definitions of concurrent soundness and source hiding since the RO does not play an essential role.

Concurrent Soundness. Firstly, we provide the definition of concurrent soundness which is basically based on the definition proposed in [YAS+12]. Roughly, in the definition of concurrent soundness, we define an adversary $\mathcal{A}$ as a *man-in-the-middle* attacker such that $\mathcal{A}$ interacts with provers as a verifier in left sessions, and at the same time interacts with an honest verifier in a right session as a prover, in a concurrent manner. For capturing this situation, in the security definition, the adversary $\mathcal{A}$ is given access to the three oracles: the *prover oracle* $\mathcal{PO}$, the *execution oracle* $\mathcal{EO}$, and the *corruption oracle* $\mathcal{CO}$. Concretely, $\mathcal{PO}$ and $\mathcal{EO}$ are used for capturing the interactions between provers and $\mathcal{A}$, and $\mathcal{CO}$ is used for capturing the corruptions of (non-target) provers. Regarding such an adversary $\mathcal{A}$ (as a verifier), concurrent soundness guarantees that $\mathcal{A}$ will not be able to make an honest verifier accept as a valid prover in the right session (except for some unavoidable trivial attacks). The formal definition is given as follows.

Definition 18 (Concurrent Soundness)**.** *Let* $\mathsf{DRA} = (\mathsf{DRA.Setup}, \langle \mathsf{DRA.Prove}, \mathsf{DRA.Verify} \rangle)$ *be a deniable ring authentication scheme. Let* $n := n(\lambda)$ *be a polynomial which denotes the number of users. Consider the following experiment for a (stateful) adversary* $\mathcal{A}$*.*

$\mathsf{Exp}^{\mathsf{cs}}_{\mathsf{DRA},\mathcal{A}}(\lambda)$:
$\mathsf{cnt} := 1$, $\mathsf{List}_{\mathsf{PO}} := \emptyset$, $\mathsf{S}_{\mathsf{Corr}} := \emptyset$
$(pp, \mathbf{sk} = (sk_i)_{i\in[n]}) \leftarrow \mathsf{DRA.Setup}(1^\lambda, n)$
$(\mathsf{R}^*, m^*) \leftarrow \mathcal{A}^{\mathcal{PO},\mathcal{EO},\mathcal{CO}}(pp)$
If $\mathsf{R}^* \not\subseteq [n]\backslash\mathsf{S}_{\mathsf{Corr}} \vee (\cdot,\cdot,\mathsf{R}^*,m^*) \in \mathsf{List}_{\mathsf{PO}}$ *then return* 0
$v \leftarrow \langle \mathcal{A}^{\mathcal{PO},\mathcal{EO},\mathcal{CO}}, \mathsf{DRA.Verify} \rangle (pp, \mathsf{R}^*, m^*)$
If $v = 1 \wedge \mathsf{R}^* \subseteq [n]\backslash\mathsf{S}_{\mathsf{Corr}} \wedge (\cdot,\cdot,\mathsf{R}^*,m^*) \notin \mathsf{List}_{\mathsf{PO}}$
then return 1 *else return* 0

In the above experiment, the prover oracle $\mathcal{PO}$*, the execution oracle* $\mathcal{EO}$*, and the corruption oracle* $\mathcal{CO}$ *are defined as follows:*

Prover oracle. $\mathcal{PO}$ *takes* (i, R, m) *such that* $i \in \mathsf{R}$ *as input, then initiates* $\mathcal{P}_{\mathsf{cnt}}$ *as a stateful instance of* $\mathsf{DRA.Prove}(sk_i, pp, \mathsf{R}, m)$ *that can be accessed only via the*

execution oracle $\mathcal{EO}$ *below, appends* $(\mathsf{cnt}, i, \mathsf{R}, m)$ *to* $\mathsf{List}_{\mathsf{PO}}$*, and sets* $\mathsf{cnt} := \mathsf{cnt}+1$*. Concretely,* $\mathcal{P}_j$ *(where* $j \in [\mathsf{cnt}]$*) takes a message* msg *as a protocol message from a verifier as input and computes the next message function of* $\mathsf{DRA.Prove}$ *(which could accompany an update of the internal state of* $\mathcal{P}_j$*). We denote this procedure as* $\mathsf{msg}' \leftarrow \mathcal{P}_j(\mathsf{msg})$*.*[1]

Execution oracle. $\mathcal{EO}$ *takes* (j, msg) *such that* $j \in [\mathsf{cnt}]$ *as input, then computes* $\mathsf{msg}' \leftarrow \mathcal{P}_j(\mathsf{msg})$ *and returns* msg' *to* $\mathcal{A}$*.*

Corruption oracle. $\mathcal{CO}$ *takes* $i \in [n]$ *as input, then returns* sk_i *to* $\mathcal{A}$ *and appends* i *to* $\mathsf{S}_{\mathsf{Corr}}$*.*

We say that DRA *satisfies concurrent soundness if for any PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{cs}}_{\mathsf{DRA},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{cs}}_{\mathsf{DRA},\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda).$$

Remark 3.2 (On the prover oracle $\mathcal{PO}$)**.** *In the prover oracle* $\mathcal{PO}$*,* $\mathcal{A}$ *can start new sessions as many times as it wants with any prover* i *with a ring* R *and a message* m *chosen by* $\mathcal{A}$ *for each session. We note that a session launched by* $\mathcal{A}$ *will not be closed even if another session is started with (possibly the same) prover* i*. Moreover, the internal state of each* $\mathcal{P}_j$ *is not shared with other instances, where* $j \in [\mathsf{cnt}]$*.*

Remark 3.3 (Selective Variant of Concurrent Soundness)**.** *In the above definition of concurrent soundness, an adversary* $\mathcal{A}$ *can choose the challenge ring* R^* *adaptively after it is given a public parameter* pp *and accesses to* $\mathcal{PO}$*,* $\mathcal{EO}$*, and* $\mathcal{CO}$*. We can also consider a selective variant of the above definition in the sense that* $\mathcal{A}$ *is required to choose the challenge ring* R^* *before it is given* pp*.*

Source Hiding. Next, we provide the definition of source hiding [Nao02]. Roughly, source hiding guarantees that a malicious verifier can be convinced only that a message is authenticated by some member in a ring R, without knowing which one is the actual prover. In line with previous works [DHIN11, Nao02, YAS+12, ZMYH17, ZCTH17], we require that the above property be guaranteed even if all of the secret keys in the system are revealed to the malicious verifier. (This setting is sometimes called the *big brother setting*.)

[1] In a protocol in which the prover first speaks and a prover instance is invoked for the first time, we only allow msg to be an empty string.

Definition 19 (Source Hiding)**.** *Let* $\mathsf{DRA} = (\mathsf{DRA.Setup}, \langle \mathsf{DRA.Prove}, \mathsf{DRA.Verify} \rangle)$ *be a deniable ring authentication scheme. Let* $n := n(\lambda)$ *be a polynomial which denotes the number of users. Consider the following experiment for a (stateful) adversary* $\mathcal{A}$*.*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{sh}}_{\mathsf{DRA},\mathcal{A}}(\lambda): \\
&\quad (pp, \mathbf{sk} = (sk_i)_{i\in[n]}) \leftarrow \mathsf{DRA.Setup}(1^\lambda, n) \\
&\quad (i_0, i_1, \mathsf{R}^*, m^*) \leftarrow \mathcal{A}(pp, \mathbf{sk}) \\
&\quad b \leftarrow \{0,1\} \\
&\quad b' \leftarrow \langle \mathsf{DRA.Prove}(sk_{i_b}), \mathcal{A} \rangle (pp, \mathsf{R}^*, m^*) \\
&\quad \textit{If } b = b' \textit{ then return } 1 \textit{ else return } 0
\end{aligned}
$$

In the above experiment, we require that $i_0, i_1 \in \mathsf{R}^* \subseteq [n]$ *hold. We say that* DRA *satisfies source hiding if for any computationally unbounded adversary* $\mathcal{A}$*,*

$$
\mathsf{Adv}^{\mathsf{sh}}_{\mathsf{DRA},\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr[\mathsf{Exp}^{\mathsf{sh}}_{\mathsf{DRA},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)
$$

holds.

Concurrent Deniability. Finally, we present the definition of concurrent deniability in the RO model. (In our construction, an RO is needed essentially in the proof of concurrent deniability.) In a nutshell, we extend the definition of concurrent deniability for deniable authentication [DGK06] into the deniable ring authentication setting.

Informally, for a deniable ring authentication scheme, concurrent deniability ensures that a (malicious) verifier of a message m under a ring R cannot convince a third party that m was authenticated by any of the provers in R even if during the interaction with the prover, the verifier can open and schedule sessions in an arbitrary way. We require that this property hold even if the malicious verifier gathers auxiliary information, which especially includes transcripts which are eavesdropped on honestly executed protocols between other honest parties. For capturing this ability of $\mathcal{A}$, in the definition, we allow $\mathcal{A}$ to access to the transcript oracle $\mathcal{TO}$. Using the transcript oracle $\mathcal{TO}$, $\mathcal{A}$ can obtain transcripts on executions of the protocol between any prover and an (honest) verifier for any message and any ring including the prover. The formal definition is as follows.

Definition 20 (Concurrent Deniability)**.** *Let* $\mathsf{DRA} = (\mathsf{DRA.Setup}, \langle \mathsf{DRA.Prove}, \mathsf{DRA.Verify} \rangle)$ *be a deniable ring authentication scheme in the RO model, and suppose it uses an RO with the input length* ℓ_{in} *and the output length* ℓ_{out}*. Let* $\mathcal{H}_{\mathcal{RO}} := \{\mathcal{RO} : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\}$ *be the set of all functions with the input length* ℓ_{in} *and the output length* ℓ_{out}*. Let* $n := n(\lambda)$

be a polynomial which denotes the number of users. Let $\mathcal{A}$ *be a (stateful) adversary and* $\mathcal{S}$ *a PPT simulator. Consider the following real experiment* $\mathsf{Exp}^{\mathsf{cd\text{-}real}}_{\mathsf{DRA},\mathcal{A}}(\lambda)$ *and ideal experiment* $\mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$.

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{cd\text{-}real}}_{\mathsf{DRA},\mathcal{A}}(\lambda):\\
&\quad \mathsf{cnt} := 1,\ \mathsf{List}_{\mathsf{PO}} := \emptyset,\\
&\quad \mathsf{List}_{\mathsf{TO}} := \emptyset,\ \mathsf{S}_{\mathsf{Corr}} := \emptyset,\ \mathsf{List}_{\mathsf{RO}} := \emptyset\\
&\quad \mathcal{RO} \leftarrow \mathcal{H}_{\mathcal{RO}},\\
&\quad (pp, \mathbf{sk} = (sk_i)_{i\in[n]}) \leftarrow \mathsf{DRA.Setup}(1^\lambda, n)\\
&\quad \mathsf{out} \leftarrow \mathcal{A}^{\mathcal{PO},\mathcal{EO},\mathcal{TO},\mathcal{CO},\mathcal{RO}}(pp)\\
&\quad \mathsf{view} := (pp, \mathsf{List}_{\mathsf{TO}}, \mathsf{S}_{\mathsf{Corr}}, \mathsf{List}_{\mathsf{RO}}, \mathsf{out})\\
&\quad \textit{Return } (\mathsf{view}, \mathcal{RO})
\end{aligned}
$$

In the above experiment, the prover oracle $\mathcal{PO}$*, the execution oracle* $\mathcal{EO}$*, and the corruption oracle* $\mathcal{CO}$ *are defined in the same way as in the experiment for the concurrent soundness. The transcript oracle* $\mathcal{TO}$ *and the random oracle* $\mathcal{RO}$ *are defined as follows:*

Transcript oracle. $\mathcal{TO}$ *takes* (i, R, m) *(such that* $i \in \mathsf{R}$*) as input, then executes* $\langle \mathsf{DRA.Prove}(sk_i), \mathsf{DRA.Verify}\rangle(pp, \mathsf{R}, m)$*, returns the transcript* tr *of the execution to* $\mathcal{A}$*, and appends* $((i, \mathsf{R}, m), \mathsf{tr})$ *to* $\mathsf{List}_{\mathsf{TO}}$.

Random oracle. $\mathcal{RO}$ *takes* x *as input, then checks whether* $(x, y) \in \mathsf{List}_{\mathsf{RO}}$ *holds for some* y*. If this is the case, then* $\mathcal{RO}$ *returns* y *to* $\mathcal{A}$*. Otherwise,* $\mathcal{RO}$ *samples* $y \leftarrow \{0,1\}^{\ell_{\mathsf{RO}}}$*, returns* y *to* $\mathcal{A}$*, and appends* (x, y) *to* $\mathsf{List}_{\mathsf{RO}}$.

The ideal experiment $\mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$ *is defined in the same way as in the real experiment in which* $\mathcal{A}$ *is replaced with* $\mathcal{S}$*, except that* $\mathcal{S}$ *is not allowed to access to the prover oracle* $\mathcal{PO}$ *and the execution oracle* $\mathcal{EO}$.

We say that DRA *satisfies concurrent deniability in the RO model if for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *such that for any PPT distinguisher* $\mathcal{D}$*,*

$$\mathsf{Adv}^{\mathsf{cd}}_{\mathsf{DRA},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) := |\Pr[\mathcal{D}^{\mathcal{RO}_{\mathsf{real}}}(\mathsf{view}_{\mathsf{real}}) = 1] - \Pr[\mathcal{D}^{\mathcal{RO}_{\mathsf{ideal}}}(\mathsf{view}_{\mathsf{ideal}}) = 1]| = \mathsf{negl}(\lambda)$$

holds, where $(\mathsf{view}_{\mathsf{real}}, \mathcal{RO}_{\mathsf{real}}) \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}real}}_{\mathsf{DRA},\mathcal{A}}(\lambda)$ *and* $(\mathsf{view}_{\mathsf{ideal}}, \mathcal{RO}_{\mathsf{ideal}}) \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$.

We note that providing the distinguisher $\mathcal{D}$ with access to the RO is essential, since otherwise $\mathcal{D}$ might not be able to even execute the protocol by itself. This treatment of ROs in defining deniability is due to the work by Pass [Pas03].

Remark 3.4 (Comparison with Prior Definitions)**.** *If we consider only one user ($n = 1$) and remove $\mathcal{CO}$, our definition is the same as the one proposed by Di Raimondo et al. [DGK06]. Actually, although [DGK06, Definition 1] says that the auxiliary information ($\mathsf{List}_{\mathsf{TO}}$) that $\mathcal{A}$ (and $\mathcal{S}$) is given is written as something that is fixed outside the real/ideal executions, their proof suggests that the authors seem to treat it in the same way as ours.*

Moreover, we should compare our definition to the one proposed by Zeng et al. [ZCTH17]. In contrast to ours, their definition does not explicitly allow that an adversary can corrupt honest users or get auxiliary information ($\mathsf{List}_{\mathsf{TO}}$), and thus we can say that our definition is stronger than theirs in this sense. (However, for fairness, we note that in their proof for concurrent deniability, they also seem to consider the situation that an adversary can obtain transcripts of honest executions of the protocol in the same way as ours.)

Chapter 4

A Ring Signature Scheme with Unconditional Anonymity in the Plain Model

We propose a generic construction of ring signature with unconditional anonymity in the plain model. Our construction is based on a two-message public coin witness indistinguishable proof with statistical privacy (which is called a statistical ZAP argument), lossy encryption, and an existential unforgeable under the chosen-message attacks in the multi-user setting with corruptions (MU-EUF-CMA$^{\mathrm{Corr}}$ secure) signature. From the previous works [BFJ+20, BHJ+15, BHY09, GJJM20], all of these building blocks can be instantiated under the quasi-polynomial learning with errors (LWE) assumption.

4.1 Technical Overview

In this section, we give the technical overview of our construction of ring signature with unconditional anonymity in the plain model. Our starting point is the generic construction of ring signature in the plain model proposed by Bender, Katz, and Morselli [BKM06]. We call this ring signature scheme the BKM scheme. The BKM scheme consist of a (standard) signature scheme, a public key encryption (PKE) scheme, and a ZAP argument with computational privacy. In the BKM scheme, a user's verification key $rvk = (vk, ek, r)$ consists of a verification key vk for a signature scheme, an encryption key ek for a PKE scheme, and a first message r for a ZAP argument. Conversely, a user's signing key $rsk = sk$ consists of only a signing key sk for the underlying signature scheme (does not include a decryption key dk

corresponding to ek). When a user signs a message m using its signing key $rsk = sk$ and a ring $\mathsf{R} = (rvk_1, \cdots, rvk_n)$ (which includes a user's verification key $rvk_{i^*} = (vk_{i^*}, ek_{i^*}, r_{i^*})$), it generates a ring signature σ as follows. Firstly, it computes a signature σ_m of m using the signing key sk. Then, it generates a ciphertext c_{i^*} of σ_m under his encryption key ek. Next, for all $i \in \{1, \cdots, n\} \backslash \{i^*\}$, it generates dummy ciphertexts c_i of $0^{|\sigma_m|}$ under other users' encryption key ek_i, where $rvk_i = (vk_i, ek_i, r_i)$. Finally, by using a ZAP argument under the first message r_1 of the first user in the ring R, it generates a proof π for the statement $(m, \mathsf{R}, c_1, \cdots, c_n)$ to show the following facts:

1. There exists an index i^* such that c_{i^*} encrypts a signature σ_m.

2. The signature σ_m verifies m under vk_{i^*}.

The resulting ring signature consists of $\sigma = (c_1, \cdots, c_n, \pi)$. When verifying a ring signature $\sigma = (c_1, \cdots, c_n, \pi)$ for a message m and a ring R, a verifier just checks a ZAP proof π for the statement $(m, \mathsf{R}, c_1, \cdots, c_n)$ under the first message r_1 of the first user in R. Regarding its anonymity, the BKM scheme only achieves a computational anonymity due to the computational privacy properties of the underlying PKE scheme and ZAP argument.

In the following, we modify the BKM scheme for upgrading the computational anonymity into unconditional one. The first step is to replace a ZAP argument used in BKM scheme as a statistical ZAP argument. By this change, a proof π in a ring signature σ provides statistical privacy while holding its computational soundness property, which is required to ensure the unforgeability for the ring signature.

As the second step, we consider how to solve a problem due to the computational privacy of the underlying PKE scheme. Looking closely the BKM scheme, we see that a decryption key dk (corresponding to an encryption key ek in a verification key rvk) is not needed when running the actual scheme. In fact, a decryption key dk is only needed in the proof of unforgeability in order to reduce the unforgeability of the BKM scheme to the unforgeability of the underlying signature scheme. Due to this feature, as the second step, we find that the underlying PKE scheme can be replaced by a *lossy encryption* scheme. A lossy encryption scheme is a special PKE scheme having two types of encryption keys: an ordinary encryption key ek and a lossy encryption key lk. When encrypting a plaintext under an ordinary encryption key ek, we can generate a standard ciphertext which is decrypted by the corresponding decryption key. On the other hand, when encrypting a plaintext under a lossy encryption key lk, we can statistically eliminate the information of the plaintext from the ciphertext (that is, the corresponding decryption key does not exist). As a basic property for lossy encryption, we require that an

ordinary encryption key and a lossy encryption key is computationally indistinguishable. In our ring signature scheme, a user's verification key rvk contains a lossy encryption key lk instead of an encryption key ek for a PKE scheme. Now, from the statistical privacy of a lossy encryption scheme, we can eliminate the information of the index i^* from the ciphertext c_{i^*} in a ring signature σ. In the proof of the unforgeability, based on the property of lossy encryption, we switch from a lossy encryption key lk to an ordinary encryption key ek in order to extract a signature σ_m from the ciphertext c_{i^*} using the corresponding decryption key dk. Note that since we only consider a computational unforgeability for ring signature, this computational key switching process does not make any problem in the security proof.

Applying these two modifications to the BKM scheme, we can obtain the first ring signature scheme with unconditional anonymity in the plain model. In addition to the above our modifications, in our actual construction, we adopt the technique for reducing the number of ciphertexts in BKM scheme proposed by Backes, Döttling, Hanzlik, Kluczniak, and Schneider [BDH+19]. By utilizing this technique, the number of ciphertexts in a ring signature reduces from n to 2. Informally, in our actual construction, a user computes a ciphertext c_0 by encrypting the signature σ_m under the lossy encryption key lk and sampling another ciphertext c_1 from the ciphertext space uniformly at random. Then, by using the underlying ZAP argument, we generate a proof π for the statement $(m, \mathsf{R}, c_0, c_1)$ to show that either c_0 OR c_1 is a ciphertext of a signature σ_m for a message m under the verification key vk.

4.2 Description

In this section, we formally describe our ring signature scheme. Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme with the message space $\mathcal{M}_{\mathsf{SIG}} = \{0,1\}^*$ and the signature space $\mathcal{S}^{\mathsf{SIG}}$. Let $\mathsf{LE} = (\mathsf{KG}, \mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a lossy encryption scheme with the plaintext space $\mathcal{M}^{\mathsf{LE}} = \mathcal{S}^{\mathsf{SIG}}$, the ciphertext space $\mathcal{CT}^{\mathsf{LE}}$, and the randomness space $\mathcal{R}^{\mathsf{Enc}}$ for Enc. Let $\mathsf{ZAP} = (\mathsf{Prove}, \mathsf{Verify})$ be a ZAP argument for $\mathcal{L}$, where

$$\mathcal{L} := \Big\{ x = (m, c_0, c_1, \mathsf{R} = (rvk_1, \cdots, rvk_n)) \Big| \ \exists w = (i, rvk = (vk, ek, r), \sigma_m, r^{\mathsf{Enc}}) \text{ s.t.} \\ rvk = rvk_i \wedge 1 = \mathsf{Ver}(vk, m \| \mathsf{R}, \sigma_m) \wedge \big(c_0 = \mathsf{Enc}(ek, \sigma_m; r^{\mathsf{Enc}}) \vee c_1 = \mathsf{Enc}(ek, \sigma_m; r^{\mathsf{Enc}})\big) \Big\}$$

Then, we construct a ring signature scheme $\mathsf{RS} = (\mathsf{RGen}, \mathsf{RSign}, \mathsf{RVer})$ with the message space $\mathcal{M}$ as described in Figure 4.1. We note that the correctness of RS is straightforward due to the correctness of SIG, LE, and ZAP.

$\mathsf{RGen}(1^\lambda)$:	$\mathsf{RSign}(rsk, m, \mathsf{R})$:	$\mathsf{RVer}(\mathsf{R}, m, \sigma)$:
$(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	Parse $rsk := (sk, rvk)$	Parse $\mathsf{R} := (rvk_1, \cdots, rvk_n)$
$ek \leftarrow \mathsf{LKG}(1^\lambda)$	Parse $rvk := (vk, ek, r)$	Parse $rvk_1 := (vk_1, ek_1, r_1)$
$r \leftarrow \{0,1\}^{\ell(\lambda)}$	Parse $\mathsf{R} := (rvk_1, \cdots, rvk_n)$	Parse $\sigma := (\pi, c_0, c_1)$
$rvk := (vk, ek, r)$	Parse $rvk_1 := (vk_1, ek_1, r_1)$	$x := (m, c_0, c_1, \mathsf{R})$
$rsk := (sk, rvk)$	If $rvk \notin \mathsf{R}$ then Return $\perp$	$b \leftarrow \mathsf{Verify}(r_1, x, \pi)$
Return (rvk, rsk)	$\sigma_m \leftarrow \mathsf{Sign}(sk, m \Vert \mathsf{R})$	Return b
	$r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$	
	$c_0 \leftarrow \mathsf{Enc}(ek, \sigma_m; r^{\mathsf{Enc}})$	
	$c_1 \leftarrow \mathcal{CT}^{\mathsf{LE}}$	
	$x := (m, c_0, c_1, \mathsf{R})$	
	$w := (i, rvk, \sigma_m, r^{\mathsf{Enc}})$	
	$\pi \leftarrow \mathsf{Prove}(r_1, x, w)$	
	Return $\sigma := (\pi, c_0, c_1)$	

Figure 4.1: Our construction of ring signature with unconditional anonymity in the plain model RS.

4.3 Security Proof

In this section, we show the following two theorems.

Theorem 4.1. *If* ZAP *satisfies statistical witness indistinguishability and* LE *satisfies statistical lossiness, then* RS *satisfies unconditional anonymity under full key exposure.*

Proof of Theorem 4.1. Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of key pairs. Let $\mathcal{A}$ be any adversary that attacks the anonymity under full key exposure of RS. We proceed the proof via a sequence of games by introducing the following games: $\mathbf{Game}_i$ for $i \in [0, 6]$.

$\mathbf{Game}_0$: This game is the original game of anonymity under full key exposure for RS conditioned on $b = 0$. The detailed description is as follows:

1. The challenger $\mathcal{C}$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{C}$ samples randomnesses $r_i^{\mathsf{Gen}} \leftarrow \mathcal{R}^{\mathsf{Gen}}$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{\mathsf{LKG}}$, and $r_i \leftarrow \{0,1\}^\ell$.
   (b) For all $i \in [n]$, $\mathcal{C}$ generates $(vk_i, sk_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i^{\mathsf{Gen}})$ and $ek_i \leftarrow \mathsf{LKG}(1^\lambda; r_i^{\mathsf{LKG}})$.
   (c) For all $i \in [n]$, $\mathcal{C}$ sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.
   (d) $\mathcal{C}$ gives the randomnesses $(r_i^{\mathsf{Gen}}, r_i^{\mathsf{LKG}})_{i \in [n]}$ to $\mathcal{A}$
2. When $\mathcal{A}$ requests a challenge to $\mathcal{C}$ by sending a tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{C}$ proceeds as follows:

(a) $\mathcal{C}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, and computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_0}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{C}$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*; r^{\mathsf{Enc}})$, and chooses $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$.

(c) $\mathcal{C}$ sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$ and $w^* := (i_0, rvk_{i_0}, \sigma_m^*, r^{\mathsf{Enc}})$, and computes $\pi^* \leftarrow \mathsf{Prove}(r_1^*, x^*, w^*)$.

(d) $\mathcal{C}$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$, and gives σ^* to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

$\mathbf{Game}_1$: This game is identical to $\mathbf{Game}_0$ except that $\mathcal{C}$ samples $r'^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and computes $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*; r'^{\mathsf{Enc}})$ and $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r')$ instead of $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$.

$\mathbf{Game}_2$: This game is identical to $\mathbf{Game}_1$ except that $\mathcal{C}$ sets $w^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$ instead of $w^* := (i_0, rvk_{i_0}, \sigma_m^*, r^{\mathsf{Enc}})$.

$\mathbf{Game}_3$: This game is identical to $\mathbf{Game}_2$ except that $\mathcal{C}$ samples $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ instead of computing $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*)$.

$\mathbf{Game}_4$: This game is identical to $\mathbf{Game}_3$ except that $\mathcal{C}$ computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$ and $c_0^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma_m^*; r^{\mathsf{Enc}})$ instead of $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$.

$\mathbf{Game}_5$: This game is identical to $\mathbf{Game}_4$ except that $\mathcal{C}$ sets $w^* := (i_1, rvk_{i_1}, \sigma_m^*, r^{\mathsf{Enc}})$ instead of $w^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$.

$\mathbf{Game}_6$: This game is identical to $\mathbf{Game}_5$ except that $\mathcal{C}$ samples $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ instead of computing $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r'^{\mathsf{Enc}})$. Note that this game is exactly the same as the original game of anonymity under full key exposure for RS conditioned on $b = 1$.

Let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ outputs $b' = 0$ in $\mathbf{Game}_i$ for $i \in [0, 6]$. By using triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) = 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right| = |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_6]| \leq \sum_{i=0}^{5} |\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]|$$

It remains to show how each $|\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]|$ is upper-bounded. To this end, we will show the following lemmata.

- There exist adversaries $\mathcal{B}_1^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{los}}}(\lambda)$ (Lemma 4.1).
- There exist an adversary $\mathcal{B}_1^{\mathsf{wi}}$ against the statistical witness indistinguishability of ZAP such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_1^{\mathsf{wi}}}(\lambda)$ (Lemma 4.2).
- There exist adversaries $\mathcal{B}_2^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_2^{\mathsf{los}}}(\lambda)$ (Lemma 4.3).
- There exist adversaries $\mathcal{B}_3^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_3^{\mathsf{los}}}(\lambda)$ (Lemma 4.4).
- There exist an adversary $\mathcal{B}_2^{\mathsf{wi}}$ against the statistical witness indistinguishability of ZAP such that $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{wi}}}(\lambda)$ (Lemma 4.5).
- There exist adversaries $\mathcal{B}_4^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_4^{\mathsf{los}}}(\lambda)$ (Lemma 4.6).

Lemma 4.1. *There exists an adversary $\mathcal{B}_1^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{los}}}(\lambda)$.*

Proof of Lemma 4.1. We construct an adversary $\mathcal{B}_1^{\mathsf{los}}$ that attacks the statistical lossiness of LE so that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{los}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i^{\mathsf{LKG}})_{i\in[n]}$ from the challenger, $\mathcal{B}_1^{\mathsf{los}}$ proceeds as follows:
   (a) $\mathcal{B}_1^{\mathsf{los}}$ generates $ek_i \leftarrow \mathsf{LKG}(1^\lambda; r_i^{\mathsf{LKG}})$ for all $i \in [n]$.
   (b) For all $i \in [n]$, $\mathcal{B}_1^{\mathsf{los}}$ samples randomness $r_i^{\mathsf{Gen}} \leftarrow \mathcal{R}^{\mathsf{Gen}}$ and generates $(sk_i, vk_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i^{\mathsf{Gen}})$.
   (c) For all $i \in [n]$, $\mathcal{B}_1^{\mathsf{los}}$ samples $r_i \leftarrow \{0,1\}^\ell$ and sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.
   (d) $\mathcal{B}_1^{\mathsf{los}}$ gives the randomness $(r_i^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, r_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ requests a challenge tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_1^{\mathsf{los}}$ proceeds as follows:
   (a) $\mathcal{B}_1^{\mathsf{los}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, and computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_0}, m^* \| \mathsf{R}^*)$ and $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$.
   (b) $\mathcal{B}_1^{\mathsf{los}}$ samples $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and generates $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*; r^{\mathsf{Enc}})$.

(c) $\mathcal{B}_1^{\mathsf{los}}$ makes its own challenge query (i_1, σ'_m) to the challenger. Upon receiving c_1^* from the challenger, it sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$ and $w^* := (i_0, rvk_{i_0}, \sigma_m^*, r^{\mathsf{Enc}})$, and computes $\pi^* \leftarrow \mathsf{Prove}(r_1^*, x^*, w^*)$.

(d) $\mathcal{B}_1^{\mathsf{los}}$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$ and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_1^{\mathsf{los}}$ outputs $\beta' := 1$ to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_1^{\mathsf{los}}$ outputs 0 to the challenger and terminates.

In the following, we let β be the challenge bit for $\mathcal{B}_1^{\mathsf{los}}$. We can see that $\mathcal{B}_1^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_0$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_1^{\mathsf{los}}$ outputs 1 given $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ is exactly the same as the probability that $\mathbf{Succ}_0$ happens in $\mathbf{Game}_0$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_0]$ holds.

On the other hand, we can see that $\mathcal{B}_1^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m)$ from its challenger. This ensures that the probability that $\mathcal{B}_1^{\mathsf{los}}$ outputs 1 given $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m)$ is exactly the same as the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_1]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{los}}}(\lambda) = |\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]| = |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]|.$$

□ (**Lemma 4.1**)

Lemma 4.2. *There exists an adversary $\mathcal{B}_1^{\mathsf{wi}}$ against the statistical witness indistinguishability of* ZAP *such that* $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{wi}}}(\lambda)$.

Proof of Lemma 4.2. We construct an adversary $\mathcal{B}_1^{\mathsf{wi}}$ that attacks the statistical witness indistinguishability of ZAP so that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_1^{\mathsf{wi}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_1^{\mathsf{wi}}$ proceeds as follows:

(a) For all $i \in [n]$, $\mathcal{B}_1^{\mathsf{wi}}$ samples randomness $r_i^{\mathsf{Gen}} \leftarrow \mathcal{R}^{\mathsf{Gen}}$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{\mathsf{LKG}}$, and $r_i \leftarrow \{0,1\}^\ell$.

(b) For all $i \in [n]$, $\mathcal{B}_1^{\mathsf{wi}}$ generates $(vk_i, sk_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i^{\mathsf{Gen}})$ and $ek_i \leftarrow \mathsf{LKG}(1^\lambda; r_i^{\mathsf{LKG}})$.

(c) For all $i \in [n]$, $\mathcal{B}_1^{\mathsf{wi}}$ sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.

(d) $\mathcal{B}_1^{\mathsf{wi}}$ gives the randomness $(r_i^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, r_i)_{i \in [n]}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_1^{\mathsf{wi}}$ proceeds as follows:

   (a) $\mathcal{B}_1^{\mathsf{wi}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_0}, m^* \| \mathsf{R}^*)$ and $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$, and sends r_1^* to the challenger.

   (b) $\mathcal{B}_1^{\mathsf{wi}}$ samples randomness $r^{\mathsf{Enc}}, r'^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*; r^{\mathsf{Enc}})$ and $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r'^{\mathsf{Enc}})$, and sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$.

   (c) $\mathcal{B}_1^{\mathsf{wi}}$ sets $w_0^* := (i_0, rvk_{i_0}, \sigma_m^*, r^{\mathsf{Enc}})$ and $w_1^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$.

   (d) $\mathcal{B}_1^{\mathsf{wi}}$ makes a query (x^*, w_0^*, w_1^*) to its oracle, gets the corresponding proof π^*, and sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$.

   (e) $\mathcal{B}_1^{\mathsf{wi}}$ gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and terminates, $\mathcal{B}_1^{\mathsf{wi}}$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_1^{\mathsf{wi}}$ outputs 0 to the challenger and terminates.

We can see that $\mathcal{B}_1^{\mathsf{wi}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_1^{\mathsf{wi}}$ outputs 1 given the proof π from the oracle $\mathcal{O}_0$ is exactly the same as the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[{\mathcal{B}_1^{\mathsf{wi}}}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(r_1^*) = 1] = \Pr[\mathbf{Succ}_1]$ holds.

On the other hand, $\mathcal{B}_1^{\mathsf{wi}}$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_1^{\mathsf{wi}}$ outputs 1 given the proof π from the oracle $\mathcal{O}_1$ is exactly the same as the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[{\mathcal{B}_1^{\mathsf{wi}}}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(r_1^*) = 1] = \Pr[\mathbf{Succ}_2]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_1^{\mathsf{wi}}}(\lambda) = |\Pr[{\mathcal{B}_1^{\mathsf{wi}}}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(r_1^*) = 1] - \Pr[{\mathcal{B}_1^{\mathsf{wi}}}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(r_1^*) = 1]| = |\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]|.$$

□ (**Lemma 4.2**)

Lemma 4.3. *There exists an adversary $\mathcal{B}_2^{\mathsf{los}}$ against the statistical lossiness of LE such that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{los}}}(\lambda)$.*

Proof of Lemma 4.3. We construct an adversary $\mathcal{B}_2^{\mathsf{los}}$ that attacks the statistical lossiness of LE so that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_2^{\mathsf{los}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. $\mathcal{B}_2^{\mathsf{los}}$ runs in the same way as the Step 1 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

2. When $\mathcal{A}$ requests a challenge tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_2^{\mathsf{los}}$ proceeds as follows:

(a) $\mathcal{B}_2^{\mathsf{los}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, and computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_0}, m^* \| \mathsf{R}^*)$ and $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{B}_2^{\mathsf{los}}$ samples $r'^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and generates $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r'^{\mathsf{Enc}})$.

(c) $\mathcal{B}_2^{\mathsf{los}}$ makes its own challenge query (i_0, σ_m^*) to the challenger. Upon receiving c_0^* from the challenger, it sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$ and $w^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$, and computes $\pi^* \leftarrow \mathsf{Prove}(r_1^*, x^*, w^*)$.

(d) $\mathcal{B}_2^{\mathsf{los}}$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$ and gives σ^* to $\mathcal{A}$.

3. $\mathcal{B}_2^{\mathsf{los}}$ runs in the same way as the Step 3 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

In the following, we let β be the challenge bit for $\mathcal{B}_2^{\mathsf{los}}$. We can see that $\mathcal{B}_2^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*)$ from its challenger. This ensures that the probability that $\mathcal{B}_2^{\mathsf{los}}$ outputs 1 given $c_0^* \leftarrow \mathsf{Enc}(ek_{i_0}, \sigma_m^*)$ is exactly the same as the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_2]$ holds.

On the other hand, we can see that $\mathcal{B}_2^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_2^{\mathsf{los}}$ outputs 1 given $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ is exactly the same as the probability that $\mathbf{Succ}_3$ happens in $\mathbf{Game}_3$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_3]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE}, \mathcal{B}_2^{\mathsf{los}}}(\lambda) = |\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]| = |\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]|.$$

□ (**Lemma 4.3**)

Lemma 4.4. *There exists an adversary* $\mathcal{B}_3^{\mathsf{los}}$ *against the statistical lossiness of* LE *such that* $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE}, \mathcal{B}_3^{\mathsf{los}}}(\lambda)$.

Proof of Lemma 4.4. We construct an adversary $\mathcal{B}_3^{\mathsf{los}}$ that attacks the statistical lossiness of LE so that $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE}, \mathcal{B}_3^{\mathsf{los}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. $\mathcal{B}_3^{\mathsf{los}}$ runs in the same way as the Step 1 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

2. When $\mathcal{A}$ requests a challenge tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_3^{\mathsf{los}}$ proceeds as follows:

(a) $\mathcal{B}_3^{\mathsf{los}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, and computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$ and $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{B}_3^{\mathsf{los}}$ samples $r'^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and generates $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r'^{\mathsf{Enc}})$.

(c) $\mathcal{B}_3^{\mathsf{los}}$ makes its own challenge query (i_1, σ_m^*) to the challenger. Upon receiving c_0^* from the challenger, it sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$ and $w^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$, and computes $\pi^* \leftarrow \mathsf{Prove}(r_1^*, x^*, w^*)$.

(d) $\mathcal{B}_3^{\mathsf{los}}$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$ and gives σ^* to $\mathcal{A}$.

3. $\mathcal{B}_3^{\mathsf{los}}$ runs in the same way as the Step 3 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

In the following, we let β be the challenge bit for $\mathcal{B}_3^{\mathsf{los}}$. We can see that $\mathcal{B}_3^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_3^{\mathsf{los}}$ outputs 1 given $c_0^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ is exactly the same as the probability that $\mathbf{Succ}_3$ happens in $\mathbf{Game}_3$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_3]$ holds.

On the other hand, we can see that $\mathcal{B}_3^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_4$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_0^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma_m^*)$ from its challenger. This ensures that the probability that $\mathcal{B}_3^{\mathsf{los}}$ outputs 1 given $c_0^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma_m^*)$ is exactly the same as the probability that $\mathbf{Succ}_4$ happens in $\mathbf{Game}_4$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_4]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_3^{\mathsf{los}}}(\lambda) = |\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]| = |\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]|.$$

□ (**Lemma 4.4**)

Lemma 4.5. *There exists an adversary* $\mathcal{B}_2^{\mathsf{wi}}$ *against the statistical witness indistinguishability of* ZAP *such that* $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{wi}}}(\lambda)$.

Proof of Lemma 4.5. We construct an adversary $\mathcal{B}_2^{\mathsf{wi}}$ that attacks the statistical witness indistinguishability of ZAP so that $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{wi}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. $\mathcal{B}_2^{\mathsf{wi}}$ runs in the same way as the Step 1 of $\mathcal{B}_1^{\mathsf{wi}}$ in the proof of Lemma 4.2.

2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_2^{\mathsf{wi}}$ proceeds as follows:

   (a) $\mathcal{B}_2^{\mathsf{wi}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_0}, m^* \| \mathsf{R}^*)$ and $\sigma'_m \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$, and sends r_1^* to the challenger.

   (b) $\mathcal{B}_2^{\mathsf{wi}}$ samples randomness $r^{\mathsf{Enc}}, r'^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma_m^*; r^{\mathsf{Enc}})$ and $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m; r'^{\mathsf{Enc}})$, and sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$.

   (c) $\mathcal{B}_2^{\mathsf{wi}}$ sets $w_0^* := (i_1, rvk_{i_1}, \sigma'_m, r'^{\mathsf{Enc}})$ and $w_1^* := (i_1, rvk_{i_1}, \sigma_m^*, r^{\mathsf{Enc}})$.

(d) $\mathcal{B}_2^{\mathsf{wi}}$ makes a query (x^*, w_0^*, w_1^*) to its oracle, gets the corresponding proof π^*, and sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$.

(e) $\mathcal{B}_2^{\mathsf{wi}}$ gives σ^* to $\mathcal{A}$.

3. $\mathcal{B}_2^{\mathsf{wi}}$ runs in the same way as the Step 3 of $\mathcal{B}_1^{\mathsf{wi}}$ in the proof of Lemma 4.2.

We can see that $\mathcal{B}_2^{\mathsf{wi}}$ perfectly simulates $\mathbf{Game}_4$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_2^{\mathsf{wi}}$ outputs 1 given the proof π from the oracle $\mathcal{O}_0$ is exactly the same as the probability that $\mathbf{Succ}_4$ happens in $\mathbf{Game}_4$. That is, $\Pr[{\mathcal{B}_2^{\mathsf{wi}}}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_4]$ holds.

On the other hand, $\mathcal{B}_2^{\mathsf{wi}}$ perfectly simulates $\mathbf{Game}_5$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_2^{\mathsf{wi}}$ outputs 1 given the proof π from the oracle $\mathcal{O}_1$ is exactly the same as the probability that $\mathbf{Succ}_5$ happens in $\mathbf{Game}_5$. That is, $\Pr[{\mathcal{B}_2^{\mathsf{wi}}}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_5]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{wi}}}(\lambda) = |\Pr[{\mathcal{B}_2^{\mathsf{wi}}}^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[{\mathcal{B}_2^{\mathsf{wi}}}^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = |\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]|.$$

□ (**Lemma 4.5**)

Lemma 4.6. *There exists an adversary* $\mathcal{B}_4^{\mathsf{los}}$ *against the statistical lossiness of* LE *such that* $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_4^{\mathsf{los}}}(\lambda)$.

Proof of Lemma 4.6. We construct an adversary $\mathcal{B}_4^{\mathsf{los}}$ that attacks the statistical lossiness of LE so that $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_4^{\mathsf{los}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. $\mathcal{B}_4^{\mathsf{los}}$ runs in the same way as the Step 1 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

2. When $\mathcal{A}$ requests a challenge tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_4^{\mathsf{los}}$ proceeds as follows:

   (a) $\mathcal{B}_4^{\mathsf{los}}$ parses $\mathsf{R}^* := (rvk_1^*, \cdots, rvk_{n^*}^*)$ and $rvk_1^* := (vk_1^*, ek_1^*, r_1^*)$, and computes $\sigma_m^* \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$ and $\sigma_m' \leftarrow \mathsf{Sign}(sk_{i_1}, m^* \| \mathsf{R}^*)$.

   (b) $\mathcal{B}_4^{\mathsf{los}}$ samples $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and generates $c_0^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma_m^*; r^{\mathsf{Enc}})$.

   (c) $\mathcal{B}_4^{\mathsf{los}}$ makes its own challenge query (i_1, σ_m') to the challenger. Upon receiving c_1^* from the challenger, $\mathcal{B}_4^{\mathsf{los}}$ sets $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$ and $w^* := (i_1, rvk_{i_1}, \sigma_m^*, r^{\mathsf{Enc}})$, and computes $\pi^* \leftarrow \mathsf{Prove}(r_1^*, x^*, w^*)$.

Finally, $\mathcal{B}_4^{\mathsf{los}}$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*)$ and gives σ^* to $\mathcal{A}$.

3. $\mathcal{B}_4^{\mathsf{los}}$ runs in the same way as the Step 3 of $\mathcal{B}_1^{\mathsf{los}}$ in the proof of Lemma 4.1.

In the following, we let β be the challenge bit for $\mathcal{B}_4^{\mathsf{los}}$. We can see that $\mathcal{B}_4^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_5$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m)$ from its challenger. This ensures that the probability that $\mathcal{B}_4^{\mathsf{los}}$ outputs 1 given $c_1^* \leftarrow \mathsf{Enc}(ek_{i_1}, \sigma'_m)$ is exactly the same as the probability that $\mathbf{Succ}_5$ happens in $\mathbf{Game}_5$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_5]$ holds.

On the other hand, we can see that $\mathcal{B}_4^{\mathsf{los}}$ perfectly simulates $\mathbf{Game}_6$ for $\mathcal{A}$ if it receives the challenge ciphertext $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_4^{\mathsf{los}}$ outputs 1 given $c_1^* \leftarrow \mathcal{CT}^{\mathsf{LE}}$ is exactly the same as the probability that $\mathbf{Succ}_6$ happens in $\mathbf{Game}_6$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_6]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_4^{\mathsf{los}}}(\lambda) = |\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]| = |\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]|.$$

□ (**Lemma 4.6**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_1^{\mathsf{los}}}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_1^{\mathsf{wi}}}(\lambda) + \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_2^{\mathsf{los}}}(\lambda) + \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_3^{\mathsf{los}}}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{ZAP},\mathcal{B}_2^{\mathsf{wi}}}(\lambda) + \mathsf{Adv}^{\mathsf{los}}_{\mathsf{LE},\mathcal{B}_4^{\mathsf{los}}}(\lambda).$$

Since ZAP satisfies statistical witness indistinguishability and LE satisfies statistical lossiness, for any computationally unbounded adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, RS satisfies unconditional anonymity under full key exposure. □ (**Theorem 4.1**)

Theorem 4.2. *If* SIG *is an MU-EUF-CMA*^{Corr} *secure signature scheme,* LE *satisfies correctness and indistinguishability of ordinary/lossy keys, and* ZAP *satisfies computational soundness, then* RS *satisfies unforgeability w.r.t. insider corruptions.*

Proof of Theorem 4.2. Let $\mathcal{A}$ be a PPT adversary that attacks the unforgeability w.r.t. insider corruptions of RS. We proceed the proof via a sequence of games. We introduce the following three games $\mathbf{Game}_i$ for $i \in [0, 2]$.

$\mathbf{Game}_0$: This game is the original game of the unforgeability w.r.t. insider corruptions for RS. The detailed description is as follows.

1. The challenger $\mathcal{C}$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{C}$ samples randomnesses $r_i^{\mathsf{Gen}} \leftarrow \mathcal{R}^{\mathsf{Gen}}$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{\mathsf{LKG}}$, and $r_i \leftarrow \{0,1\}^{\ell}$.

(b) For all $i \in [n]$, $\mathcal{C}$ generates $(vk_i, sk_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i^{\mathsf{Gen}})$ and $ek_i \leftarrow \mathsf{LKG}(1^\lambda; r_i^{\mathsf{LKG}})$.

(c) For all $i \in [n]$, $\mathcal{C}$ sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.

(d) $\mathcal{C}$ sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$, and gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{C}$ proceeds as follows:

Signing Queries.

(a) $\mathcal{C}$ parses $\mathsf{R} := (rvk'_1, \cdots, rvk'_{n'})$ and $rvk'_1 := (vk'_1, ek'_1, r'_1)$, samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, and computes $\sigma_m \leftarrow \mathsf{Sign}(sk_j, m \| \mathsf{R})$ and $c_0 \leftarrow \mathsf{Enc}(ek_j, \sigma_m; r^{\mathsf{Enc}})$.

(b) $\mathcal{C}$ samples $c_1 \leftarrow \mathcal{CT}^{\mathsf{LE}}$, sets $x := (m, c_0, c_1, \mathsf{R})$ and $w := (j, rvk_j, \sigma_m, r^{\mathsf{Enc}})$, and computes $\pi \leftarrow \mathsf{Prove}(r'_1, x, w)$.

(c) $\mathcal{C}$ sets $\sigma := (\pi, c_0, c_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

Corruption Queries.

$\mathcal{C}$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. $\mathcal{A}$ outputs a tuple $(\mathsf{R}^*, m^*, \sigma^*)$.

$\mathbf{Game}_1$: This game is identical to $\mathbf{Game}_0$ except that $\mathcal{C}$ generates ordinary encryption keys $(ek_i, sk_i) \leftarrow \mathsf{KG}(1^\lambda)$ instead of generating lossy encryption keys $ek_i \leftarrow \mathsf{LKG}(1^\lambda)$ for all $i \in [n]$.

$\mathbf{Game}_2$: This game is identical to $\mathbf{Game}_1$ except that $\mathcal{C}$ requires an additional condition for the success condition of $\mathcal{A}$. More precisely, we require a forgery $(\mathsf{R}^*, m^*, \sigma^* = (\pi^*, c_0^*, c_1^*))$ output by $\mathcal{A}$ to satisfy $x^* \in \mathcal{L}$, where $x^* := (m^*, c^*, c_1^*, \mathsf{R}^*)$.

For $i \in [0, 2]$, we let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ succeeds in outputting a tuple $(\mathsf{R}^*, m^*, \sigma^*)$ satisfying $1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*) \wedge \mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}} \wedge (\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}}$ in $\mathbf{Game}_i$. By using triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Succ}_0] \leq \sum_{i=0}^{1} |\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]| + \Pr[\mathbf{Succ}_2].$$

It remains to show how each $|\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]|$ for $i \in [0, 1]$ and $\Pr[\mathbf{Succ}_2]$ are upper-bounded. To this end, we show the following lemmata.

- There exists an adversary $\mathcal{B}^{\mathsf{key}}$ against the indistinguishability of ordinary / lossy keys of LE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{B}^{\mathsf{key}}}(\lambda)$ (Lemma 4.7).

- There exists an adversary $\mathcal{B}^{\mathsf{sound}}$ against the computational soundness of ZAP such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}(\lambda)$ (Lemma 4.8).
- There exists an adversary $\mathcal{B}^{\mathsf{unf}}$ against the MU-EUF-CMA$^{\mathrm{Corr}}$ security of SIG such that $\Pr[\mathbf{Succ}_2] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{B}^{\mathsf{unf}}}(\lambda)$ (Lemma 4.9).

Lemma 4.7. *There exists an adversary $\mathcal{B}^{\mathsf{key}}$ against the indistinguishability of ordinary / lossy keys of LE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{B}^{\mathsf{key}}}(\lambda)$.*

Proof of Lemma 4.7. We construct an adversary $\mathcal{B}^{\mathsf{key}}$ that attacks the indistinguishability of ordinary / lossy keys of LE so that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{B}^{\mathsf{key}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving a set of encryption keys $(ek_i)_{i\in[n]}$ from the challenger, $\mathcal{B}^{\mathsf{key}}$ proceeds as follows:
   (a) $\mathcal{B}^{\mathsf{key}}$ generates $(sk_i, vk_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for all $i \in [n]$.
   (b) For all $i \in [n]$, $\mathcal{B}^{\mathsf{key}}$ samples $r_i \leftarrow \{0,1\}^\ell$ and sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.
   (c) $\mathcal{B}^{\mathsf{key}}$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$.
2. $\mathcal{B}^{\mathsf{key}}$ runs in the same way as the Step 2 in $\mathbf{Game}_0$.
3. When $\mathcal{A}$ outputs a tuple $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, if $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ holds, $\mathcal{B}^{\mathsf{key}}$ outputs 1 and terminates. Otherwise, $\mathcal{B}^{\mathsf{key}}$ outputs 0 and terminates.

In the following, we let β be the challenge bit for $\mathcal{B}^{\mathsf{key}}$. We can see that $\mathcal{B}^{\mathsf{key}}$ perfectly simulates $\mathbf{Game}_0$ for $\mathcal{A}$ if it receives the lossy encryption keys $(ek_i)_{i\in[n]}$ generated by $\mathsf{LKG}(1^\lambda)$ from its challenger. This ensures that the probability that $\mathcal{B}^{\mathsf{key}}$ outputs 1 given the lossy encryption keys $(ek_i)_{i\in[n]}$ is exactly the same as the probability that $\mathbf{Succ}_0$ happens in $\mathbf{Game}_0$. That is, $\Pr[\beta' = 1|\beta = 1] = \Pr[\mathbf{Succ}_0]$ holds.

On the other hand, we can see that $\mathcal{B}^{\mathsf{key}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the ordinary encryption keys $(ek_i)_{i\in[n]}$ generated by $\mathsf{KG}(1^\lambda)$ from its challenger. This ensures that the probability that $\mathcal{B}^{\mathsf{key}}$ outputs 1 given the ordinary encryption keys $(ek_i)_{i\in[n]}$ is exactly the

same as the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_1]$ holds. Therefore, we have

$$\mathsf{Adv}^{\mathsf{key}}_{\mathsf{LE},\mathcal{B}^{\mathsf{key}}}(\lambda) = |\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]| = |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]|.$$

□ (**Lemma 4.7**)

Lemma 4.8. *There exists an adversary* $\mathcal{B}^{\mathsf{sound}}$ *against the computational soundness of* ZAP *such that* $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}(\lambda)$.

Proof of Lemma 4.8. For $i \in \{1, 2\}$, we let $\mathbf{Bad}_i$ be the event that $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^* = (\pi^*, c_0^*, c_1^*))$ satisfying $x^* \notin \mathcal{L} \wedge 1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*) \wedge \mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}$ in $\mathbf{Game}_i$, where $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$. In the following, we call such a forgery a *bad forgery*. $\mathbf{Game}_1$ proceeds identically to $\mathbf{Game}_2$ unless $\mathbf{Bad}_1$ happens. Therefore, we have the inequality $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| \leq \Pr[\mathbf{Bad}_1] = \Pr[\mathbf{Bad}_2]$. Then, we construct an adversary $\mathcal{B}^{\mathsf{sound}}$ that attacks the computational soundness of ZAP so that $\Pr[\mathbf{Bad}_1] = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i)_{i \in [n]}$ from the challenger, $\mathcal{B}^{\mathsf{sound}}$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}^{\mathsf{sound}}$ generates $(ek_i, dk_i) \leftarrow \mathsf{KG}(1^\lambda)$ and $(sk_i, vk_i) \leftarrow \mathsf{Gen}(1^\lambda)$.
   (b) For all $i \in [n]$, $\mathcal{B}^{\mathsf{sound}}$ sets $rvk_i := (vk_i, ek_i, r_i)$ and $rsk_i := (sk_i, rvk_i)$.
   (c) $\mathcal{B}^{\mathsf{sound}}$ sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$, and gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$.
2. $\mathcal{B}^{\mathsf{sound}}$ runs in the same way as the Step 2 in $\mathbf{Game}_0$.
3. When $\mathcal{A}$ outputs a tuple $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}^{\mathsf{sound}}$ proceeds as follows:
   (a) $\mathcal{B}^{\mathsf{sound}}$ parses $\mathsf{R}^* := (rvk^*_{i_t})_{t \in [n]}$, $rvk^*_{i_1} := (vk^*_{i_1}, ek^*_{i_1}, r^*_{i_1})$, and $\sigma^* := (\pi^*, c_0^*, c_1^*)$.
   (b) $\mathcal{B}^{\mathsf{sound}}$ sets $x^* := (m^*, c_0^*, c_1^*, \sigma^*)$, outputs $(r^*_{i_1}, x^*, \pi^*)$, and terminates.

From the above construction of $\mathcal{B}^{\mathsf{sound}}$, it is easy to see that $\mathcal{B}^{\mathsf{sound}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$. Recall that the success condition of $\mathcal{B}^{\mathsf{sound}}$ is to output a tuple of a randomness, a statement, and a proof (r^*, x^*, π^*) satisfying $x^* \notin \mathcal{L} \wedge 1 = \mathsf{Verify}(r^*, x^*, \pi^*)$ for some $r^* \in (r_i)_{i \in [n]}$.

If $\mathcal{A}$ outputs a bad forgery $(\mathsf{R}^*, m^*, \sigma^*)$, then $x^* \notin \mathcal{L} \wedge 1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*) \wedge \mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}$ holds. Due to the construction of RS, the condition $1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)$ implies the condition

$1 = \mathsf{Verify}(r^*_{i_1}, x^*, \pi^*)$. Moreover, due to the condition $\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}$, we can ensure that $r^*_{i_1}$ is included in $(r_i)_{i\in[n]}$. Thus, when $\mathcal{A}$ outputs a bad forgery $(\mathsf{R}^*, m^*, \sigma^*)$, $\mathcal{B}^{\mathsf{sound}}$ achieves its success condition by returning $(r^*_{i_1}, x^*, \pi^*)$ to its challenger.

From the above arguments, the probability that $\mathcal{A}$ outputs a bad forgery is exactly the same as the probability that $\mathcal{B}^{\mathsf{sound}}$ breaks the computational soundness of ZAP. Hence, we have $\Pr[\mathbf{Bad}_1] = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}(\lambda)$, which in turn implies $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}(\lambda)$. □ (**Lemma 4.8**)

Lemma 4.9. *There exists an adversary* $\mathcal{B}^{\mathsf{unf}}$ *against the MU-EUF-CMA*^{Corr} *security of* SIG *such that* $\Pr[\mathbf{Succ}_2] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{B}^{\mathsf{unf}}}(\lambda)$.

Proof of Lemma 4.9. We construct an adversary $\mathcal{B}^{\mathsf{unf}}$ that attacks the MU-EUF-CMA$^{\mathrm{Corr}}$ security of SIG so that $\Pr[\mathbf{Succ}_2] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{B}^{\mathsf{unf}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving a set of verification keys $(vk_i)_{i\in[n]}$ from the challenger, $\mathcal{B}^{\mathsf{unf}}$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}^{\mathsf{unf}}$ generates $(ek_i, dk_i) \leftarrow \mathsf{KG}(1^\lambda)$ and samples $r_i \leftarrow \{0,1\}^\ell$.
   (b) $\mathcal{B}^{\mathsf{unf}}$ sets $rvk_i := (vk_i, ek_i, r_i)$ for all $i \in [n]$, $\mathsf{S}_{\mathsf{Sig}} := \emptyset$, and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$, and gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{B}^{\mathsf{unf}}$ proceeds as follows:

   Signing Queries.
   (a) $\mathcal{B}^{\mathsf{unf}}$ parses $\mathsf{R} := (rvk'_1, \cdots, rvk'_{n'})$ and $rvk'_1 := (vk'_1, ek'_1, r'_1)$.
   (b) $\mathcal{B}^{\mathsf{unf}}$ makes a signing query $(j, m\|\mathsf{R})$ to its challenger. Upon receiving a signature σ_m from its challenger, $\mathcal{B}^{\mathsf{unf}}$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and computes $c_0 \leftarrow \mathsf{Enc}(ek_j, \sigma_m; r^{\mathsf{Enc}})$.
   (c) $\mathcal{B}^{\mathsf{unf}}$ samples $c_1 \leftarrow \mathcal{CT}^{\mathsf{LE}}$, sets $x := (m, c_0, c_1, \mathsf{R})$ and $w := (j, rvk_j, \sigma_m, r^{\mathsf{Enc}})$, and computes $\pi \leftarrow \mathsf{Prove}(r'_1, x, w)$.
   (d) $\mathcal{B}^{\mathsf{unf}}$ sets $\sigma := (\pi, c_0, c_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

   Corruption Queries.
   $\mathcal{B}^{\mathsf{unf}}$ makes a corruption query j to its challenger. Upon receiving sk_j from its challenger, $\mathcal{B}^{\mathsf{unf}}$ sets $rsk_j := (sk_j, rvk_j)$ and gives rsk_j to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}^{\mathsf{unf}}$ parses $\mathsf{R}^* := (rvk_{i_1}, \cdots, rvk_{i_t})$ for some $t = |\mathsf{R}^*|$ and $\sigma^* := (\pi^*, c_0^*, c_1^*)$, and computes $\sigma_{j,0}^* \leftarrow \mathsf{Dec}(dk_{i_j}, c_0^*)$ and $\sigma_{j,1}^* \leftarrow \mathsf{Dec}(dk_{i_j}, c_1^*)$ for all $j \in [t]$. Next, $\mathcal{B}^{\mathsf{unf}}$ proceeds as follows:

   (a) $\mathcal{B}^{\mathsf{unf}}$ checks whether $1 = \mathsf{Ver}(vk_{i_j}, m^* \| \mathsf{R}^*, \sigma_{j,0}^*)$ holds for all $j \in [t]$. If the condition holds for some $j \in [n]$, then $\mathcal{B}^{\mathsf{unf}}$ returns $(i_j, m^* \| \mathsf{R}^*, \sigma_{j,0}^*)$ to its challenger and terminates.

   (b) $\mathcal{B}^{\mathsf{unf}}$ checks whether $1 = \mathsf{Ver}(vk_{i_j}, m^* \| \mathsf{R}^*, \sigma_{j,1}^*)$ holds for all $j \in [t]$. If the condition holds for some $j \in [n]$, then $\mathcal{B}^{\mathsf{unf}}$ returns $(i_j, m^* \| \mathsf{R}^*, \sigma_{j,1}^*)$ to its challenger and terminates.

We can see that $\mathcal{B}^{\mathsf{unf}}$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$. Then, we show that $\mathcal{B}^{\mathsf{unf}}$ can output a valid forgery $(i_j, m^* \| \mathsf{R}^*, \sigma_{j,b}^*)$ satisfying $(1 = \mathsf{Ver}(vk_{i_j}, m^* \| \mathsf{R}^*, \sigma_{j,b}^*)) \wedge ((m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_j) \wedge (i_j \notin \mathsf{C})$ for some $j \in [t]$ and $b \in \{0,1\}$ if $\mathcal{A}$ makes a valid forgery $(\mathsf{R}^*, m^*, \sigma^*)$.

If $\mathcal{A}$ makes a valid forgery $(\mathsf{R}^*, m^*, \sigma^*)$ in $\mathbf{Game}_2$, then $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ and $x^* \in \mathcal{L}$ hold, where $x^* := (m^*, c_0^*, c_1^*, \mathsf{R}^*)$.

Firstly, we can see that $(m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_{i_j}$ holds by the fact $(\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}}$.

Secondly, $1 = \mathsf{Verify}(r_{i_1}^*, x^*, \pi^*)$ holds due to the condition $1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)$. Moreover, by combining the additional condition $x^* \in \mathcal{L}$ in $\mathbf{Game}_2$, we can ensure that there exists a witness $(i_j, rvk^* = (vk^*, ek^*), \sigma^*, r^*)$ such that $(rvk^* = rvk_{i_j}^* (= (vk_{i_j}, ek_{i_j}))) \wedge (1 = \mathsf{Ver}(vk^*, m^* \| \mathsf{R}^*, \sigma_m^*)) \wedge (c_0^* = \mathsf{Enc}(ek^*, \sigma_m^*; r^*) \vee c_1^* = \mathsf{Enc}(ek^*, \sigma_m^*; r^{\mathsf{Enc}}))$ for some $j \in [t]$. That is, due to the perfect correctness of LE, for some $b \in \{0,1\}$, we can ensure that $\sigma_m^* = \mathsf{Dec}(dk_{i_j}, c_b^*) = \sigma_{j,b}^*$ and $1 = \mathsf{Ver}(vk_{i_j}, m^* \| \mathsf{R}^*, \sigma_{j,b}^*)$ hold.

Finally, we can see that $\mathcal{A}$ does not make a corruption query i_j because $rvk^* = rvk_{i_j}^*$ and $\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}$ hold. Therefore, we have $i_j \notin \mathsf{C}$.

From the above arguments, we can see that $\mathcal{B}^{\mathsf{unf}}$ can output a valid forgery $(i_j, m^* \| \mathsf{R}^*, \sigma_{j,b}^*)$ for some $b \in \{0,1\}$ if $\mathcal{A}$ outputs a valid forgery $(\mathsf{R}^*, m^*, \sigma^*)$. Therefore, we have $\Pr[\mathbf{Succ}_2] = \mathsf{Adv}_{\mathsf{SIG},\mathcal{B}^{\mathsf{unf}}}^{\mathsf{unf}}(\lambda)$. □ (**Lemma 4.9**)

Putting everything together, we obtain

$$\mathsf{Adv}_{\mathsf{RS},\mathcal{A}}^{\mathsf{unf}}(\lambda) \leq \mathsf{Adv}_{\mathsf{LE},\mathcal{B}^{\mathsf{key}}}^{\mathsf{key}}(\lambda) + \mathsf{Adv}_{\mathsf{ZAP},\mathcal{B}^{\mathsf{sound}}}^{\mathsf{sound}}(\lambda) + \mathsf{Adv}_{\mathsf{SIG},\mathcal{B}^{\mathsf{unf}}}^{\mathsf{unf}}(\lambda).$$

Since LE satisfies correctness and indistinguishability of ordinary / lossy keys, ZAP satisfies computational soundness, and SIG is MU-EUF-CMA$^{\mathrm{Corr}}$ secure, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{RS},\mathcal{A}}^{\mathsf{unf}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, RS satisfies unforgeability w.r.t. insider corruptions. □ (**Theorem 4.2**)

Chapter 5

A Tightly Secure Ring Signature Scheme in the Plain Model

Based on the above motivation, we give an affirmative answer to the question. More specifically, we give a new construction of tightly secure ring signature without depending on the random oracles. For the overview and comparisons with the previous state-of-the-art schemes, we refer to Table 5.1. There, we highlight the merits of our scheme by using the red color.

Scheme	Signature size	Anonymity	Model	Assumption	Tightness	Remark
Libert et al. [LPQ18]	$\mathcal{O}(\log n)$	Unconditional	RO model	DDH	$\mathcal{O}(1)$	Practically Efficient
González [Gon19]	$\mathcal{O}(\sqrt[3]{n})$	Unconditional	CRS model	DLIN	$\mathcal{O}(n^2, Q_{sig})$	Practically Efficient
Backes et al. [BDH+19]	$\mathcal{O}(\log n)$	Computational	Plain model	DLIN	$\mathcal{O}(n)$	
Ours	$\mathcal{O}(\log n)$	Computational	Plain model	DLIN	$\mathcal{O}(1)$	

Table 5.1: Comparison between state-of-the-art ring signature schemes and ours. (n denotes the size of a ring.)

Noteworthily, the signature size of our second scheme is the same as one of the tightly secure ring signature scheme in the random oracle model proposed by Libert et al. [LPQ18] asymptotically. That is, the signature size of our second scheme is $\mathcal{O}(\log n)$, where n is the size of a ring.

Our scheme is inspired by the novel ring signature scheme recently proposed by Backes et al. [BDH+19]. In [BDH+19], they provided the first logarithmic-size ring signature scheme in the plain model under the DLIN assumption over the pairing group. Their building blocks consist of a standard signature scheme [HJ12], a public key encryption (PKE) scheme with pseudorandom public keys and ciphertexts [ElG84], an NIWI proof system in the plain model [GOS06], and a somewhere perfectly binding (SPB) hash function [OPWW15]. Our

second scheme also requires the same building blocks. We stress that all of the security properties of these building blocks can be reduced to the DLIN assumption tightly using the self-reducibility of the DLIN assumption [HJ12] or ensured information-theoretically.

5.1 Technical Overview

The Difficulty of Giving A Tight Security Proof for Unforgeability. In the context of ring signatures, the generally accepted security notion for unforgeability is *unforgeability w.r.t. insider corruptions* proposed by Bender, Katz, and Morselli [BKM06]. Informally, the unforgeability w.r.t. insider corruptions is defined using an experiment that proceeds as follows.

1. An adversary $\mathcal{A}$ is given a set of the verification keys $\mathbf{rvk} = (rvk_1, \cdots, rvk_n)$ generated by the key generation algorithm.

2. $\mathcal{A}$ can make two types of queries: signing queries and corruption queries. If $\mathcal{A}$ makes a signing query (j, R, m), a signature σ made by the signing key rsk_j corresponding to rvk_j, the ring R, and the message m, is given to $\mathcal{A}$. Also, if $\mathcal{A}$ makes a corruption query j, the corresponding signing key rsk_j is given to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a tuple $(\mathsf{R}^*, m^*, \sigma^*)$.

We say that $\mathcal{A}$ breaks the unforgeability w.r.t. insider corruptions if σ^* is a valid signature for a *new* message m^* under the ring R^* containing only *uncorrupted* verification keys. In other words, if $\mathcal{A}$ cannot make such a forgery, we can say that a ring signature scheme satisfies the unforgeability w.r.t. insider corruptions.

In the following, we explain the difficulty of giving a tight security proof for the unforgeability w.r.t. insider corruptions. For explaining the difficulty clearly, let us consider a simple construction of ring signatures based on a standard signature scheme and an NIWI-PoK. In this scheme, a signer generates a signature of his message and ring (including his verification key) by using the underlying signature scheme. Then, he generates a proof which ensures that **his signature is valid** and **his verification key is included in the ring** by using the underlying NIWI-PoK. Finally, he sets the proof as his resulting ring signature σ. A verifier given the ring signature σ checks whether the proof is valid.

Regarding this scheme, thanks to the extractability of the underlying NIWI-PoK, the unforgeability w.r.t. insider corruptions is reduced to the unforgeability of the underlying

signature scheme by constructing a reduction algorithm that simply guesses the index i^* of an uncorrupted user (signing key). However, this straightforward reduction suffers from a reduction cost of $L = \frac{1}{N}$, where N is the number of all users, since an adversary of the unforgeability w.r.t. insider corruptions might make a corruption query for the index i^*. Thus, we cannot obtain a tight security in terms of the number of all users for this scheme. In general, when proving the unforgeability w.r.t. insider corruptions, the similar problem occurs in the other methods for constructing a ring signature scheme.

Our Approach to Overcome the Difficulty. Libert et al. [LPQ18] overcome the above difficulty in the RO model. Their technique highly depends on the programmability of the RO to deal with the corruption queries by an adversary. Unfortunately, we cannot use this technique because we consider the standard model.

To overcome the above difficulty without using the RO methodology, we focus on the work of Bader, Hofheinz, Jager, Kiltz, and Li [BHJ+15]. In [BHJ+15], they showed that a tightly secure signature scheme *with corruptions* can be constructed by a tightly secure signature scheme *without corruptions* and a tightly secure NIWI-PoK. Their idea is simple but powerful. Informally, the description of their scheme is as follows. In their scheme, a verification key is composed of two verification keys vk_0 and vk_1 of the underlying signature scheme. A signing key is composed of either signing key sk_α corresponding to vk_α, where α is sampled from $\{0, 1\}$ uniformly at random. A signature is constructed from an NIWI OR-proof of knowledge which proves that the underlying signature can be verified either under vk_0 OR vk_1.

Regarding the security proof, we consider constructing the following reduction algorithm $\mathcal{B}$ for the unforgeability of an underlying signature scheme. At first, $\mathcal{B}$ embeds his verification key vk into the verification key $vk_{1\oplus\alpha}$ and generates an opposite verification/signing key (vk_α, sk_α) by himself. Here, $\mathcal{B}$ knows one side signing key sk_α, and thus $\mathcal{B}$ can deal with any corruption query made by an adversary $\mathcal{A}$ against the unforgeability with corruptions. Then, $\mathcal{B}$ hopes that $\mathcal{A}$ makes a proof of knowledge π^* of a signature σ^* under $\mathcal{B}$'s verification key $vk_{1\oplus\alpha}$. If $\mathcal{A}$ makes such a proof, then $\mathcal{B}$ can extract the signature σ^* from the proof of knowledge π^* and utilize it to break the unforgeability of the underlying signature scheme.

Our approach is to extend the above technique to a ring signature setting for providing a tight security proof for unforgeability w.r.t. insider corruptions. In other words, our core idea is to combine a membership proof for a ring with the above proof of knowledge for achieving a tightness. Concretely, a signer firstly makes a signature σ for own message and ring using his signing key sk_α of the underlying signature scheme. Then, he generates a proof of knowledge showing that his verification key belongs to the ring AND his signature σ is valid either under

vk_0 OR vk_1, and sets the proof as his resulting signature.

How to Construct A Tightly Secure Ring Signature Scheme in the Plain Model. As mentioned above, our tightly secure ring signature scheme in the plain model is based on the novel ring signature scheme recently proposed by Backes et al. [BDH+19]. Their starting point is the ring signature scheme of Bender et al. [BKM06], and they improved the performance of this scheme in two aspects to obtain logarithmic-sized signatures. One is to reduce the number of ciphertexts of a PKE scheme in a ring signature and the other is to compress the witness for the proof of an NIWI proof system in a ring signature. They achieve the improvement regarding ciphertexts (resp., a witness) by utilizing the pseudorandomness of public keys and ciphertexts of a PKE scheme (resp., the somewhere perfectly binding property of a SPB hash function). Regarding anonymity, their scheme only achieves a computational anonymity due to the computational security properties of a PKE scheme and a somewhere perfectly binding hash function.

As a core technique for proving anonymity, in their construction, they make an OR-proof using the underlying NIWI proof system for ensuring that either of two ciphertexts (resp., hash keys) of an underlying PKE scheme (resp., SPB hash function) is valid. The important point for our second scheme is that the technique for achieving a tightness used in our first scheme is efficiently compatible with the above techniques of the Backes et al.'s ring signature scheme since their scheme already depends on the OR-proof technique. Somewhat surprisingly, while our scheme achieves a tight security, the efficiency is almost same as one of the original Backes et al.'s ring signature scheme.

In the following sections, we show our ring signature scheme with logarithmic-size signatures in the plain model. First, in Section 5.2, we describe our scheme. Then, in Section 5.3, we give security proofs for our scheme.

5.2 Description

In this section, we formally describe our ring signature scheme with logarithmic-size signatures in the plain model. Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme with the message space $\{0,1\}^*$ and the randomness space $\mathcal{R}^{\mathsf{Gen}}$ for the key generation algorithm Gen. Let $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with the plaintext space $\mathcal{M}^{\mathsf{PKE}}$, the public key space $\mathcal{PK}^{\mathsf{PKE}}$, the randomness space $\mathcal{R}^{\mathsf{Enc}}$ for the encryption algorithm Enc, and the ciphertext space $\mathcal{CT}^{\mathsf{PKE}}$. We require that $\mathcal{M}^{\mathsf{PKE}}$ is equal to the signature space of SIG. Let

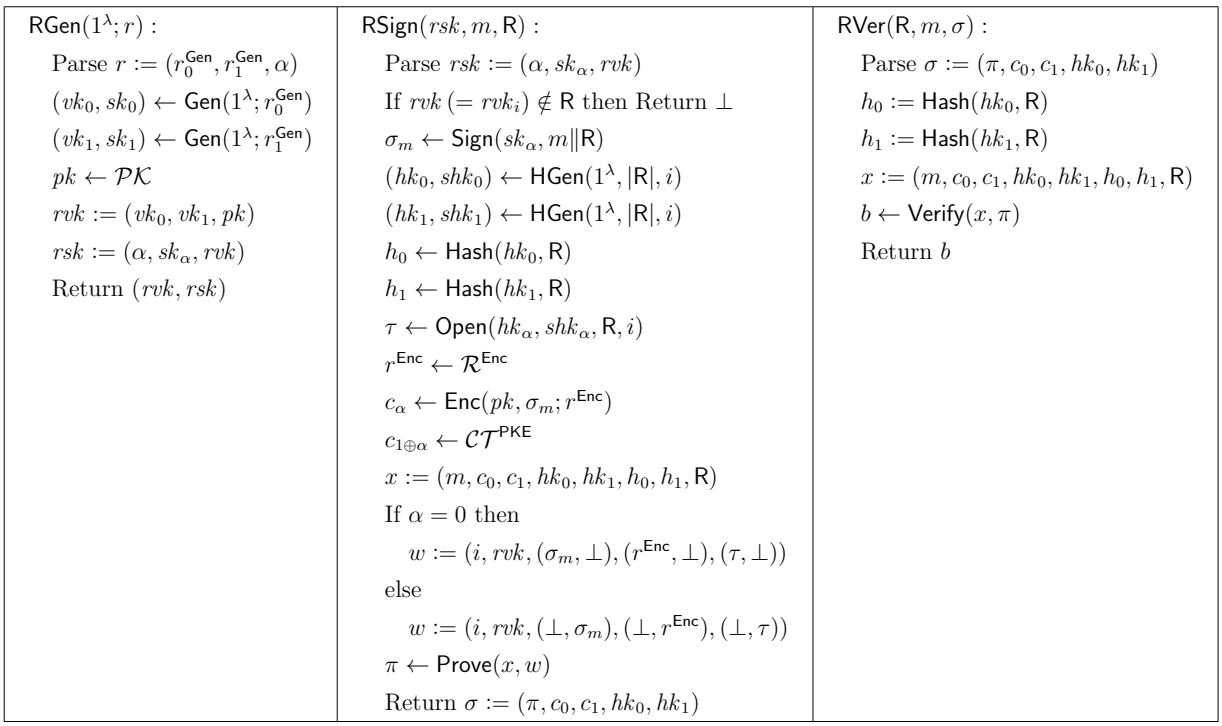

Figure 5.1: Our construction of tightly secure logarithmic-size ring signature in the plain model RS.

$\mathsf{SPBH} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{HVer})$ be a somewhere perfectly binding hash function with private local opening. Let $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ be a non-interactive proof system in the plain model for $\mathcal{L}$, where

$$\mathcal{L} := \Big\{ (m, c_0, c_1, hk_0, hk_1, h_0, h_1, R = (rvk_1, \cdots, rvk_n)) \Big| \exists (i, rvk = (vk_0, vk_1, pk), (\sigma_0, \sigma_1), (r_0, r_1), (\tau_0, \tau_1)) \text{ s.t.} \Big((1 = \mathsf{Ver}(vk_0, m \| \mathsf{R}, \sigma_0)) \wedge (c_0 = \mathsf{Enc}(pk, \sigma_0; r_0)) \wedge (1 = \mathsf{HVer}(hk_0, h_0, i, rvk, \tau_0)) \Big) \bigvee \Big((1 = \mathsf{Ver}(vk_1, m \| \mathsf{R}, \sigma_1)) \wedge (c_1 = \mathsf{Enc}(pk, \sigma_1; r_1)) \wedge (1 = \mathsf{HVer}(hk_1, h_1, i, rvk, \tau_1)) \Big) \Big\}.$$

Then, we construct our ring signature scheme $\mathsf{RS} = (\mathsf{RGen}, \mathsf{RSign}, \mathsf{RVer})$ with the message space $\mathcal{M}$ as described in Figure 5.1. We note that the correctness of RS is straightforward due to the correctness of SIG, PKE, SPBH, and NIWI.

5.3 Security Proof

In this section, we show that our ring signature scheme RS satisfies (unconditional) anonymity under full key exposure (Theorem 5.1) and unforgeability w.r.t. insider corruptions (Theo-

rem 5.2).

Theorem 5.1. *If* NIWI *satisfies computational witness indistinguishability,* PKE *satisfies pseudorandomness of ciphertexts, and* SPBH *satisfies index hiding, then* RS *satisfies computational anonymity under full key exposure. More precisely, for any PPT adversary* $\mathcal{A}$ *against the computational anonymity under full key exposure of* RS*, there exist adversaries* $\mathcal{B}_i$ *with* $\mathbf{Time}(\mathcal{A}) \leq \min_{i\in[9]}\{\mathbf{Time}(\mathcal{B}_i)\}$*, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda) + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda) + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda) + \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda).\end{aligned}$$

Proof of Theorem 5.1 Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of key pairs. Let $\mathcal{A}$ be any PPT adversary that attacks the anonymity under full key exposure of RS. We introduce the following ten games: $\mathbf{Game}_i$ for $i \in [0, 9]$.

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original game of anonymity under full key exposure for RS conditioned on $b = 0$. The detailed description is as follows.

1. The challenger $\mathcal{C}$ firstly proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{C}$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r^{\mathsf{LKG}}_i \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0, 1\}$.
   (b) For all $i \in [n]$, $\mathcal{C}$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$, and samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r^{LKG}_i.
   (c) For all $i \in [n]$, $\mathcal{C}$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{C}$ gives the randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}, r^{\mathsf{LKG}}_i, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ requests a challenge to $\mathcal{C}$ by sending a tuple $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{C}$ proceeds as follows:
   (a) $\mathcal{C}$ computes $\sigma^*_m \leftarrow \mathsf{Sign}(sk_{i_0,\alpha_{i_0}}, m^*\|\mathsf{R}^*)$, $(hk^*_0, shk^*_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $(hk^*_1, shk^*_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h^*_0 \leftarrow \mathsf{Hash}(hk^*_0, \mathsf{R}^*)$, $h^*_1 \leftarrow \mathsf{Hash}(hk^*_1, \mathsf{R}^*)$, and $\tau^* \leftarrow \mathsf{Open}(hk^*_{\alpha_{i_0}}, shk^*_{\alpha_{i_0}}, \mathsf{R}^*, i_0)$.
   (b) $\mathcal{C}$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_0}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma^*_m; r^{\mathsf{Enc}})$, samples $c^*_{1\oplus\alpha_{i_0}} \leftarrow \mathcal{CT}^{\mathsf{PKE}}$, and sets $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.
   (c) If $\alpha_{i_0} = 0$ holds, $\mathcal{C}$ sets $w^* := (i_0, rvk_{i_0}, (\sigma^*_m, \bot), (r^{\mathsf{Enc}}, \bot), (\tau^*, \bot))$. Otherwise, $\mathcal{C}$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma^*_m), (\bot, r^{\mathsf{Enc}}), (\bot, \tau^*))$.

(d) $\mathcal{C}$ computes $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, and gives σ^* to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates.

$\mathbf{Game}_1$: $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except for the following change. When generating the challenge signature $\sigma^* = (\pi^*, c_0^*, c_1^*)$, $\mathcal{C}$ computes both $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^* \| \mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^* \| \mathsf{R}^*)$. Moreover, $\mathcal{C}$ computes both $\tau_0 \leftarrow \mathsf{Open}(hk_0^*, shk_0^*, \mathsf{R}^*, i_0)$ and $\tau_1 \leftarrow \mathsf{Open}(hk_1^*, shk_1^*, \mathsf{R}^*, i_0)$. Then, $\mathcal{C}$ computes $c_{1\oplus\alpha_{i_0}}^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_0}})$ instead of $c_{1\oplus\alpha_{i_0}}^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$.

$\mathbf{Game}_2$: $\mathbf{Game}_2$ is identical to $\mathbf{Game}_1$ except that $\mathcal{C}$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \sigma_1), (r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}}), (\tau_0, \tau_1))$. Here, we can see that the witness w^* does not depend on the randomness α_{i_0}.

$\mathbf{Game}_3$: $\mathbf{Game}_3$ is identical to $\mathbf{Game}_2$ except that $\mathcal{C}$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r_1^{\mathsf{Enc}}), (\bot, \tau_1))$ if $\alpha_{i_1} = 0$ holds. Otherwise, $\mathcal{C}$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r_0^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$.

$\mathbf{Game}_4$: $\mathbf{Game}_4$ is identical to $\mathbf{Game}_3$ except that $\mathcal{C}$ computes $(hk_{\alpha_{i_1}}, shk_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$ instead of $(hk_{\alpha_{i_1}}, shk_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$.

$\mathbf{Game}_5$: $\mathbf{Game}_5$ is identical to $\mathbf{Game}_4$ except that $\mathcal{C}$ samples $c_{\alpha_{i_1}}^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ instead of computing $c_{\alpha_{i_1}}^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{\alpha_{i_1}}; r^{\mathsf{Enc}})$.

$\mathbf{Game}_6$: $\mathbf{Game}_6$ is identical to $\mathbf{Game}_5$ except that $\mathcal{C}$ computes $\sigma_{\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_1,\alpha_{i_1}}, m^* \| \mathsf{R}^*)$ and $c_{\alpha_{i_1}}^* \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}}; r^{\mathsf{Enc}})$ instead of sampling $c_{\alpha_{i_1}}^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$.

$\mathbf{Game}_7$: $\mathbf{Game}_7$ is identical to $\mathbf{Game}_6$ except for the following changes. Firstly, $\mathcal{C}$ computes $\tau_{\alpha_{i_1}} \leftarrow \mathsf{Open}(hk_{\alpha_{i_1}}, shk_{\alpha_{i_1}}, \mathsf{R}^*, i_1)$ instead of $\tau_{\alpha_{i_1}} \leftarrow \mathsf{Open}(hk_{\alpha_{i_1}}, shk_{\alpha_{i_1}}, \mathsf{R}^*, i_0)$. Moreover, if $\alpha_{i_1} = 0$ holds, then $\mathcal{C}$ sets $w^* := (i_1, rvk_{i_1}, (\sigma_0, \bot), (r_0^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{C}$ sets $w^* := (i_1, rvk_{i_1}, (\bot, \sigma_1), (\bot, r_1^{\mathsf{Enc}}), (\bot, \tau_1))$.

$\mathbf{Game}_8$: $\mathbf{Game}_8$ is identical to $\mathbf{Game}_7$ except that $\mathcal{C}$ samples $c_{1\oplus\alpha_{i_1}}^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ instead of computing $c_{1\oplus\alpha_{i_1}}^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}})$.

$\mathbf{Game}_9$: $\mathbf{Game}_9$ is identical to $\mathbf{Game}_8$ except that $\mathcal{C}$ computes $(hk_{1\oplus\alpha_{i_1}}, shk_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$ instead of $(hk_{1\oplus\alpha_{i_1}}, shk_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$. Note that this game is equal to the original game of anonymity under full key exposure for RS conditioned on $b = 1$.

Let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ outputs $b' = 0$ in $\mathbf{Game}_i$ for $i \in [0, 9]$. By using the triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) = 2 \cdot \left|\Pr[b = b'] - \frac{1}{2}\right| = |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_9]| \leq \sum_{i=0}^{8} |\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]| .$$

It remains to show how each $|\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]|$ is upper-bounded. To this end, we show the following lemmata.

- There exists an adversary $\mathcal{B}_1$ against the pseudorandomness of ciphertexts of NIWI such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{NIWI},\mathcal{B}_1}(\lambda)$ (Lemma 5.1).
- There exists an adversary $\mathcal{B}_2$ against the computational witness indistinguishability of NIWI such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$ (Lemma 5.2).
- There exists an adversary $\mathcal{B}_3$ against the computational witness indistinguishability of NIWI such that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda)$ (Lemma 5.3).
- There exists an adversary $\mathcal{B}_4$ against the index hiding of SPBH such that $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda)$ (Lemma 5.4).
- There exists an adversary $\mathcal{B}_5$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda)$ (Lemma 5.5).
- There exists an adversary $\mathcal{B}_6$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda)$ (Lemma 5.6).
- There exists an adversary $\mathcal{B}_7$ against the computational witness indistinguishability of NIWI such that $|\Pr[\mathbf{Succ}_6] - \Pr[\mathbf{Succ}_7]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda)$ (Lemma 5.7).
- There exists an adversary $\mathcal{B}_8$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_7] - \Pr[\mathbf{Succ}_8]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda)$ (Lemma 5.8).
- There exists an adversary $\mathcal{B}_9$ against the index hiding of SPBH such that $|\Pr[\mathbf{Succ}_8] - \Pr[\mathbf{Succ}_9]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda)$ (Lemma 5.9).

Lemma 5.1. *There exists an adversary $\mathcal{B}_1$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$.*

Proof of Lemma 5.1 We construct an adversary $\mathcal{B}_1$ that attacks the pseudorandomness of ciphertexts of PKE so that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i^{\mathsf{LKG}})_{i\in[n]}$ from the challenger, $\mathcal{B}_1$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_1$ samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{LKG}.
   (b) For all $i \in [n]$, $\mathcal{B}_1$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0,1\}$.
   (c) For all $i \in [n]$, $\mathcal{B}_1$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$.
   (d) For all $i \in [n]$, $\mathcal{B}_1$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (e) $\mathcal{B}_1$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_1$ proceeds as follows:
   (a) $\mathcal{B}_1$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^*\|\mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^*\|\mathsf{R}^*)$.
   (b) $\mathcal{B}_1$ computes $(hk_0^*, shk_0^*) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $(hk_1^*, shk_1^*) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk_0^*, shk_0^*, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1^*, shk_1^*, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_1$ sets $m^* := \sigma_{1\oplus\alpha_{i_0}}$ and queries (i_0, m^*) to the challenger.
   (d) Upon receiving c^* from the challenger, $\mathcal{B}_1$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_0}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{\alpha_{i_0}}; r^{\mathsf{Enc}})$, and sets $c^*_{1\oplus\alpha_{i_0}} := c^*$ and $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$.
   (e) If $\alpha_{i_0} = 0$ holds, then $\mathcal{B}_1$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r_0^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{B}_1$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r_1^{\mathsf{Enc}}), (\bot, \tau_1))$.
   (f) $\mathcal{B}_1$ computes a proof $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, and gives σ^* to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_1$ outputs $\beta' := 1$ to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_1$ outputs $\beta' := 0$ to the challenger and terminates.

In the following, let β be the challenge bit for $\mathcal{B}_1$. We can see that $\mathcal{B}_1$ perfectly simulates $\mathbf{Game}_0$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_1$ outputs 1 given $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ is equal to the probability that $\mathbf{Succ}_0$ happens in $\mathbf{Game}_0$. That is, $\Pr[\beta' = 1|\beta = 0] = \Pr[\mathbf{Succ}_0]$ holds. On the other

hand, we can see that $\mathcal{B}_1$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_0}})$ from its challenger. This ensures that the probability that $\mathcal{B}_1$ outputs 1 given $c^* \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{1\oplus\alpha_{i_0}})$ is equal to the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[\beta' = 1|\beta = 1] = \Pr[\mathbf{Succ}_1]$ holds. Therefore, we have $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$. □ **(Lemma 5.1)**

Lemma 5.2. *There exists an adversary $\mathcal{B}_2$ against the computational witness indistinguishability of* NIWI *such that* $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$.

Proof of Lemma 5.2 We construct an adversary $\mathcal{B}_2$ that attacks the computational witness indistinguishability of NIWI so that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_2$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_2$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r^{\mathsf{LKG}}_i \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0,1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_2$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomness r^{LKG}_i.
   (c) For all $i \in [n]$, $\mathcal{B}_2$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_2$ gives the randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}, r^{\mathsf{LKG}}_i, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_2$ proceeds as follows:
   (a) $\mathcal{B}_2$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^*\|\mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^*\|\mathsf{R}^*)$.
   (b) $\mathcal{B}_2$ computes $(hk^*_0, shk^*_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $(hk^*_1, shk^*_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h^*_0 \leftarrow \mathsf{Hash}(hk^*_0, \mathsf{R}^*)$, $h^*_1 \leftarrow \mathsf{Hash}(hk^*_1, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk^*_0, shk^*_0, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk^*_1, shk^*_1, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_2$ samples randomnesses $r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1 \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_0}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma^*_m; r^{\mathsf{Enc}}_{\alpha_{i_0}})$, samples $c^*_{1\oplus\alpha_{i_0}} \leftarrow \mathcal{CT}^{\mathsf{PKE}}$, and sets $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.
   (d) If $\alpha_{i_0} = 0$ holds, then $\mathcal{B}_2$ sets $w^*_0 := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r^{\mathsf{Enc}}_0, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{B}_2$ sets $w^*_0 := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}_1), (\bot, \tau_1))$.
   (e) $\mathcal{B}_2$ sets $w^*_1 := (i_0, rvk_{i_0}, (\sigma_0, \sigma_1), (r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1), (\tau_0, \tau_1))$, makes a query (x^*, w^*_0, w^*_1) to its oracle, and gets the corresponding proof π^*.
   (f) $\mathcal{B}_2$ sets $\sigma^* := (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1)$ and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_2$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_2$ outputs 0 to the challenger and terminates.

We can see that $\mathcal{B}_2$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_2$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_0$ is equal to the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[\mathcal{B}_2^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_1]$ holds. On the other hand, $\mathcal{B}_2$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_2$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_1$ is equal to the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[\mathcal{B}_2^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_2]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) = |\Pr[\mathcal{B}_2^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}_2^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = |\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]|$. □ **(Lemma 5.2)**

Lemma 5.3. *There exists an adversary $\mathcal{B}_3$ against the computational witness indistinguishability of* NIWI *such that* $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda)$.

Proof of Lemma 5.3 We construct an adversary $\mathcal{B}_3$ that attacks the computational witness indistinguishability of NIWI so that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_3$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_3$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0,1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_3$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{LKG}.
   (c) For all $i \in [n]$, $\mathcal{B}_3$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_3$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_3$ proceeds as follows:
   (a) $\mathcal{B}_3$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^* \| \mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^* \| \mathsf{R}^*)$.
   (b) $\mathcal{B}_3$ computes $(hk_0^*, shk_0^*) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $(hk_1^*, shk_1^*) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk_0^*, shk_0^*, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1^*, shk_1^*, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_3$ samples randomnesses $r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_0; r_0^{\mathsf{Enc}})$ and $c_1^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_1; r_1^{\mathsf{Enc}})$, and sets $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$ and $w_0^* := (i_0, rvk_{i_0}, (\sigma_0, \sigma_1), (r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}}), (\tau_0, \tau_1))$.

(d) If $\alpha_{i_1} = 0$ holds, $\mathcal{B}_3$ sets $w_1^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r_1^{\mathsf{Enc}}), (\bot, \tau_1))$. Otherwise, $\mathcal{B}_3$ sets $w_1^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r_0^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$.

(e) $\mathcal{B}_3$ makes a query (x^*, w_0^*, w_1^*) to its oracle and gets the corresponding proof π^*.

(f) $\mathcal{B}_3$ sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$ and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_3$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_3$ outputs 0 to the challenger and terminates.

We can see that $\mathcal{B}_3$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_3$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_0$ is equal to the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[\mathcal{B}_3^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_2]$ holds. On the other hand, $\mathcal{B}_3$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_3$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_1$ is equal to the probability that $\mathbf{Succ}_3$ happens in $\mathbf{Game}_3$. That is, $\Pr[\mathcal{B}_3^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_3]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda) = |\Pr[\mathcal{B}_3^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}_3^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = |\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]|$. □ **(Lemma 5.3)**

Lemma 5.4. *There exists an adversary $\mathcal{B}_4$ against the index hiding of* SPBH *such that* $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda)$.

Proof of Lemma 5.4 We construct an adversary $\mathcal{B}_4$ that attacks the index hiding of SPBH so that $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_4$ proceeds as follows:

   (a) For all $i \in [n]$, $\mathcal{B}_4$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}_{\mathsf{PKE}}^{key}$, and $\alpha_i \leftarrow \{0,1\}$.

   (b) For all $i \in [n]$, $\mathcal{B}_4$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomnesses r_i^{LKG}.

   (c) For all $i \in [n]$, $\mathcal{B}_4$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.

   (d) $\mathcal{B}_4$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_4$ proceeds as follows:

   (a) $\mathcal{B}_4$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^* \| \mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{B}_4$ sends $(|\mathsf{R}^*|, i_0, i_1)$ to its challenger. Upon receiving hk from the challenger, $\mathcal{B}_4$ sets $hk^*_{\alpha_{i_1}} := hk$, computes $(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h^*_0 \leftarrow \mathsf{Hash}(hk^*_0, \mathsf{R}^*)$, $h^*_1 \leftarrow \mathsf{Hash}(hk^*_1, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk^*_0, shk^*_0, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk^*_1, shk^*_1, \mathsf{R}^*, i_0)$.

(c) $\mathcal{B}_4$ samples randomnesses $r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1 \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_0 \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_0; r^{\mathsf{Enc}}_0)$ and $c^*_1 \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_1; r^{\mathsf{Enc}}_1)$, and sets $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.

(d) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_4$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}_1), (\bot, \tau_1))$. Otherwise, $\mathcal{B}_4$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r^{\mathsf{Enc}}_0, \bot), (\tau_0, \bot))$.

(e) $\mathcal{B}_4$ computes a proof $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1)$, and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_4$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_4$ outputs 0 to the challenger and terminates.

In the following, let $\beta \in \{0,1\}$ be the challenge bit for $\mathcal{B}_4$. We can see that $\mathcal{B}_4$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$. This ensures that the probability that $\mathcal{B}_4$ outputs 1 given the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$ is equal to the probability that $\mathbf{Succ}_3$ happens in $\mathbf{Game}_3$. That is, $\Pr[1 = \mathcal{B}_4(hk)|\beta = 0] = \Pr[\mathbf{Succ}_3]$ holds. On the other hand, $\mathcal{B}_4$ perfectly simulates $\mathbf{Game}_4$ for $\mathcal{A}$ if it receives the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$. This ensures that the probability that $\mathcal{B}_4$ outputs 1 given the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$ is equal to the probability that $\mathbf{Succ}_4$ happens in $\mathbf{Game}_4$. That is, $\Pr[1 = \mathcal{B}_4(hk)|\beta = 1] = \Pr[\mathbf{Succ}_4]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda) = |\Pr[1 = \mathcal{B}_4(hk)|\beta = 0] - \Pr[1 = \mathcal{B}_4(hk)|\beta = 1]| = |\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]|$. □ **(Lemma 5.4)**

Lemma 5.5. *There exists an adversary $\mathcal{B}_5$ against the pseudorandomness of ciphertexts of* PKE *such that* $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda)$.

Proof of Lemma 5.5 We construct an adversary $\mathcal{B}_5$ that attacks the pseudorandomness of ciphertexts of PKE so that $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r^{\mathsf{LKG}}_i)_{i\in[n]}$ from the challenger, $\mathcal{B}_5$ proceeds as follows:

(a) For all $i \in [n]$, $\mathcal{B}_5$ samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r^{LKG}_i.

(b) For all $i \in [n]$, $\mathcal{B}_5$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0,1\}$.

(c) For all $i \in [n]$, $\mathcal{B}_5$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$.

(d) For all $i \in [n]$, $\mathcal{B}_5$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.

(e) $\mathcal{B}_5$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_5$ proceeds as follows:

(a) $\mathcal{B}_5$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{i_0,0}, m^* \| \mathsf{R}^*)$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{i_0,1}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{B}_5$ computes $(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$, $(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk_0^*, shk_0^*, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1^*, shk_1^*, \mathsf{R}^*, i_0)$.

(c) $\mathcal{B}_5$ sets $m^* := \sigma_{\alpha_{i_1}}$ and queries (i_0, m^*) to the challenger.

(d) Upon receiving c^* from the challenger, $\mathcal{B}_5$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}}; r^{\mathsf{Enc}})$, and sets $c^*_{\alpha_{i_1}} := c^*$ and $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$.

(e) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_5$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau_1))$. Otherwise, $\mathcal{B}_5$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$.

(f) $\mathcal{B}_5$ computes $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and terminates, $\mathcal{B}_5$ outputs $\beta' := 1$ to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_5$ outputs $\beta' := 0$ to the challenger and terminates.

In the following, let β be the challenge bit for $\mathcal{B}_5$. We can see that $\mathcal{B}_5$ perfectly simulates $\mathbf{Game}_4$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{\alpha_{i_1}})$ from its challenger. This ensures that the probability that $\mathcal{B}_5$ outputs 1 given $c^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{\alpha_{i_1}})$ is equal to the probability that $\mathbf{Succ}_4$ happens in $\mathbf{Game}_4$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_4]$ holds. On the other hand, we can see that $\mathcal{B}_5$ perfectly simulates $\mathbf{Game}_5$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_5$ outputs 1 given $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ is equal to the probability that $\mathbf{Succ}_5$ happens in $\mathbf{Game}_5$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_5]$ holds. Therefore, we have $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = |\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda)$. □ **(Lemma 5.5)**

Lemma 5.6. *There exists an adversary $\mathcal{B}_6$ against the pseudorandomness of ciphertexts of* PKE *such that* $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda)$.

Proof of Lemma 5.6 We construct an adversary $\mathcal{B}_6$ that attacks the pseudorandomness of ciphertexts of PKE so that $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i^{\mathsf{LKG}})_{i\in[n]}$ from the challenger, $\mathcal{B}_6$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_6$ samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{LKG}.
   (b) For all $i \in [n]$, $\mathcal{B}_6$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0,1\}$.
   (c) For all $i \in [n]$, $\mathcal{B}_6$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$.
   (d) For all $i \in [n]$, $\mathcal{B}_6$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (e) $\mathcal{B}_6$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_6$ proceeds as follows:
   (a) $\mathcal{B}_6$ computes $\sigma_{\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_1,\alpha_{i_1}}, m^*\|\mathsf{R}^*)$ and $\sigma_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_0,1\oplus\alpha_{i_1}}, m^*\|\mathsf{R}^*)$.
   (b) $\mathcal{B}_6$ computes $(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$, $(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, $\tau_0 \leftarrow \mathsf{Open}(hk_0^*, shk_0^*, \mathsf{R}^*, i_0)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1^*, shk_1^*, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_6$ sets $m^* := \sigma_{\alpha_{i_1}}$ and queries (i_1, m^*) to the challenger.
   (d) Upon receiving c^* from the challenger, $\mathcal{B}_6$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}}; r^{\mathsf{Enc}})$, and sets $c^*_{\alpha_{i_1}} := c^*$ and $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$.
   (e) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_6$ sets $w^* := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau_1))$. Otherwise, $\mathcal{B}_6$ sets $w^* := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$.
   (f) $\mathcal{B}_6$ computes $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, and gives σ^* to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_6$ outputs $\beta' := 1$ to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_6$ outputs $\beta' := 0$ to the challenger and terminates.

In the following, let β be the challenge bit for $\mathcal{B}_6$. We can see that $\mathcal{B}_6$ perfectly simulates $\mathbf{Game}_5$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_6$ outputs 1 given $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ is equal to the probability that $\mathbf{Succ}_5$ happens in $\mathbf{Game}_5$. That is, $\Pr[\beta' = 1|\beta = 0] = \Pr[\mathbf{Succ}_5]$ holds. On the other hand, we can see that $\mathcal{B}_6$ perfectly simulates $\mathbf{Game}_6$ for $\mathcal{A}$ if it receives the challenge ciphertext

$c^* \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}})$ from its challenger. This ensures that the probability that $\mathcal{B}_6$ outputs 1 given $c^* \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}})$ is equal to the probability that $\mathbf{Succ}_6$ happens in $\mathbf{Game}_6$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_6]$ holds. Therefore, we have $|\Pr[\mathbf{Succ}_5] - \Pr[\mathbf{Succ}_6]| = |\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda)$. □ **(Lemma 5.6)**

Lemma 5.7. *There exists an adversary $\mathcal{B}_7$ against the computational witness indistinguishability of* NIWI *such that* $|\Pr[\mathbf{Succ}_6] - \Pr[\mathbf{Succ}_7]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda)$.

Proof of Lemma 5.7 We construct an adversary $\mathcal{B}_7$ that attacks the computational witness indistinguishability of NIWI so that $|\Pr[\mathbf{Succ}_6] - \Pr[\mathbf{Succ}_7]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_7$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_7$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r^{\mathsf{LKG}}_i \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0,1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_7$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomness r^{LKG}_i.
   (c) For all $i \in [n]$, $\mathcal{B}_7$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_7$ gives the randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}, r^{\mathsf{LKG}}_i, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_7$ proceeds as follows:
   (a) $\mathcal{B}_7$ computes $\sigma_{\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_1,\alpha_{i_1}}, m^* \| \mathsf{R}^*)$ and $\sigma_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_0,1\oplus\alpha_{i_1}}, m^* \| \mathsf{R}^*)$.
   (b) $\mathcal{B}_7$ computes $(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$, $(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h^*_0 \leftarrow \mathsf{Hash}(hk^*_0, \mathsf{R}^*)$, $h^*_1 \leftarrow \mathsf{Hash}(hk^*_1, \mathsf{R}^*)$, $\tau_{\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}, \mathsf{R}^*, i_1)$, and $\tau_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_7$ samples randomnesses $r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1 \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}}; r^{\mathsf{Enc}}_{\alpha_{i_1}})$ and $c^*_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}}; r^{\mathsf{Enc}}_{1\oplus\alpha_{i_1}})$, and sets $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.
   (d) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_7$ sets $w^*_0 := (i_0, rvk_{i_0}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}_1), (\bot, \tau_1))$. Otherwise, $\mathcal{B}_7$ sets $w^*_0 := (i_0, rvk_{i_0}, (\sigma_0, \bot), (r^{\mathsf{Enc}}_0, \bot), (\tau_0, \bot))$.
   (e) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_7$ sets $w^*_1 := (i_1, rvk_{i_1}, (\sigma_0, \bot), (r^{\mathsf{Enc}}_0, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{B}_7$ sets $w^*_1 := (i_1, rvk_{i_1}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}_1), (\bot, \tau_1))$.
   (f) $\mathcal{B}_7$ queries (x^*, w^*_0, w^*_1) to its oracle and gets the corresponding proof π^*.
   (g) $\mathcal{B}_7$ sets $\sigma^* := (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1)$ and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_7$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_7$ outputs 0 to the challenger and terminates.

We can see that $\mathcal{B}_7$ perfectly simulates $\mathbf{Game}_6$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_7$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_0$ is equal to the probability that $\mathbf{Succ}_6$ happens in $\mathbf{Game}_6$. That is, $\Pr[\mathcal{B}_7^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_6]$ holds. On the other hand, $\mathcal{B}_7$ perfectly simulates $\mathbf{Game}_7$ for $\mathcal{A}$ if it receives the proof π^* from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_7$ outputs 1 given the proof π^* from the oracle $\mathcal{O}_1$ is equal to the probability that $\mathbf{Succ}_7$ happens in $\mathbf{Game}_7$. That is, $\Pr[\mathcal{B}_7^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_7]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda) = |\Pr[\mathcal{B}_7^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}_7^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = |\Pr[\mathbf{Succ}_6] - \Pr[\mathbf{Succ}_7]|$. □ **(Lemma 5.7)**

Lemma 5.8. *There exists an adversary $\mathcal{B}_8$ against the pseudorandomness of ciphertexts of* PKE *such that* $|\Pr[\mathbf{Succ}_7] - \Pr[\mathbf{Succ}_8]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda)$.

Proof of Lemma 5.8 We construct an adversary $\mathcal{B}_8$ that attacks the pseudorandomness of ciphertexts of PKE so that $|\Pr[\mathbf{Succ}_7] - \Pr[\mathbf{Succ}_8]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i^{\mathsf{LKG}})_{i\in[n]}$ from the challenger, $\mathcal{B}_8$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_8$ samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{LKG}.
   (b) For all $i \in [n]$, $\mathcal{B}_8$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0,1\}$.
   (c) For all $i \in [n]$, $\mathcal{B}_8$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$.
   (d) For all $i \in [n]$, $\mathcal{B}_8$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (e) $\mathcal{B}_8$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_8$ proceeds as follows:
   (a) $\mathcal{B}_8$ computes $\sigma_{\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_1,\alpha_{i_1}}, m^*\|\mathsf{R}^*)$ and $\sigma_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_0,1\oplus\alpha_{i_1}}, m^*\|\mathsf{R}^*)$.
   (b) $\mathcal{B}_8$ computes $(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$, $(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$, $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, $\tau_{\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}, \mathsf{R}^*, i_1)$, and $\tau_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}, \mathsf{R}^*, i_0)$.
   (c) $\mathcal{B}_8$ sets $m^* := \sigma_{1\oplus\alpha_{i_1}}$ and queries (i_0, m^*) to the challenger.
   (d) Upon receiving c^* from the challenger, $\mathcal{B}_8$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}}; r^{\mathsf{Enc}})$, and sets $c^*_{1\oplus\alpha_{i_1}} := c^*$ and $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$.

(e) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_8$ sets $w^* := (i_1, rvk_{i_1}, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{B}_8$ sets $w^* := (i_1, rvk_{i_1}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau_1))$.

(f) $\mathcal{B}_8$ computes $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_8$ outputs $\beta' := 1$ to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_8$ outputs $\beta' := 0$ to the challenger and terminates.

In the following, let β be the challenge bit for $\mathcal{B}_8$. We can see that $\mathcal{B}_8$ perfectly simulates $\mathbf{Game}_7$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}})$ from its challenger. This ensures that the probability that $\mathcal{B}_8$ outputs 1 given $c^* \leftarrow \mathsf{Enc}(pk_{i_0}, \sigma_{1\oplus\alpha_{i_1}})$ is equal to the probability that $\mathbf{Succ}_7$ happens in $\mathbf{Game}_7$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_7]$ holds. On the other hand, we can see that $\mathcal{B}_8$ perfectly simulates $\mathbf{Game}_8$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ from its challenger. This ensures that the probability that $\mathcal{B}_8$ outputs 1 given $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ is equal to the probability that $\mathbf{Succ}_8$ happens in $\mathbf{Game}_8$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_8]$ holds. Therefore, we have $|\Pr[\mathbf{Succ}_7] - \Pr[\mathbf{Succ}_8]| = |\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda)$. □ **(Lemma 5.8)**

Lemma 5.9. *There exists an adversary $\mathcal{B}_9$ against the index hiding of* SPBH *such that* $|\Pr[\mathbf{Succ}_8] - \Pr[\mathbf{Succ}_9]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda)$.

Proof of Lemma 5.9 We construct an adversary $\mathcal{B}_9$ that attacks the index hiding of SPBH so that $|\Pr[\mathbf{Succ}_8] - \Pr[\mathbf{Succ}_9]| = \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_9$ proceeds as follows:

   (a) For all $i \in [n]$, $\mathcal{B}_9$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0,1\}$.

   (b) For all $i \in [n]$, $\mathcal{B}_9$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$ and samples $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{key}.

   (c) For all $i \in [n]$, $\mathcal{B}_9$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.

   (d) $\mathcal{B}_9$ gives the randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}, r_i^{\mathsf{LKG}}, \alpha_i)_{i\in[n]}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes a challenge query $(i_0, i_1, \mathsf{R}^*, m^*)$, $\mathcal{B}_9$ proceeds as follows:

   (a) $\mathcal{B}_9$ computes $\sigma_{\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_1,\alpha_{i_1}}, m^* \| \mathsf{R}^*)$ and $\sigma_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Sign}(sk_{i_0,1\oplus\alpha_{i_1}}, m^* \| \mathsf{R}^*)$.

(b) $\mathcal{B}_9$ sends $(|\mathsf{R}^*|, i_0, i_1)$ to its challenger. Upon receiving hk from the challenger, $\mathcal{B}_9$ sets $hk^*_{1\oplus\alpha_{i_1}} := hk$, computes $(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$, $h^*_0 \leftarrow \mathsf{Hash}(hk^*_0, \mathsf{R}^*)$, $h^*_1 \leftarrow \mathsf{Hash}(hk^*_1, \mathsf{R}^*)$, $\tau_{\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{\alpha_{i_1}}, shk^*_{\alpha_{i_1}}, \mathsf{R}^*, i_1)$, and $\tau_{1\oplus\alpha_{i_1}} \leftarrow \mathsf{Open}(hk^*_{1\oplus\alpha_{i_1}}, shk^*_{1\oplus\alpha_{i_1}}, \mathsf{R}^*, i_0)$.

(c) $\mathcal{B}_9$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c^*_{\alpha_{i_1}} \leftarrow \mathsf{Enc}(pk_{i_1}, \sigma_{\alpha_{i_1}}; r^{\mathsf{Enc}})$, samples $c^*_{1\oplus\alpha_{i_1}} \leftarrow \mathcal{CT}^{\mathsf{PKE}}$, and sets $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.

(d) If $\alpha_{i_1} = 0$ holds, then $\mathcal{B}_9$ sets $w^* := (i_1, rvk_{i_1}, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau_0, \bot))$. Otherwise, $\mathcal{B}_9$ sets $w^* := (i_1, rvk_{i_1}, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau_1))$.

(e) $\mathcal{B}_9$ computes $\pi^* \leftarrow \mathsf{Prove}(x^*, w^*)$, sets $\sigma^* := (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1)$, and gives σ^* to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}_9$ outputs 1 to the challenger and terminates if $b' = 0$ holds. Otherwise, $\mathcal{B}_9$ outputs 0 to the challenger and terminates.

In the following, let $\beta \in \{0,1\}$ be the challenge bit for $\mathcal{B}_9$. We can see that $\mathcal{B}_9$ perfectly simulates $\mathbf{Game}_8$ for $\mathcal{A}$ if it receives the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$. This ensures that the probability that $\mathcal{B}_9$ outputs 1 given the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_0)$ is equal to the probability that $\mathbf{Succ}_8$ happens in $\mathbf{Game}_8$. That is, $\Pr[1 = \mathcal{B}_9(hk)|\beta = 0] = \Pr[\mathbf{Succ}_8]$ holds. On the other hand, $\mathcal{B}_9$ perfectly simulates $\mathbf{Game}_9$ for $\mathcal{A}$ if it receives the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$. This ensures that the probability that $\mathcal{B}_9$ outputs 1 given the hashing key $hk \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}^*|, i_1)$ is equal to the probability that $\mathbf{Succ}_9$ happens in $\mathbf{Game}_9$. That is, $\Pr[1 = \mathcal{B}_9(hk)|\beta = 1] = \Pr[\mathbf{Succ}_9]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda) = |\Pr[1 = \mathcal{B}_9(hk)|\beta = 0] - \Pr[1 = \mathcal{B}_9(hk)|\beta = 1]| = |\Pr[\mathbf{Succ}_8] - \Pr[\mathbf{Succ}_9]|$. □ **(Lemma 5.9)**

Putting everything together, we obtain $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_4}(\lambda) + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_6}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_7}(\lambda) + \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_8}(\lambda) + \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{SPBH},\mathcal{B}_9}(\lambda)$.

Since NIWI satisfies the computational witness indistinguishability, PKE satisfies pseudorandomness of ciphertexts, and SPBH satisfies index hiding, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, RS satisfies computational anonymity under full key exposure. □ **(Theorem 5.1)**

Theorem 5.2. *If* SIG *satisfies MU-EUF-CMA security,* PKE *satisfies pseudorandomness of public keys and pseudorandomness of ciphertexts,* SPBH *satisfies somewhere perfectly binding, and* NIWI *satisfies computational witness indistinguishability and perfect soundness, then* RS *satisfies unforgeability w.r.t. insider corruptions. More precisely, for any PPT adversary* $\mathcal{A}$

against the unforgeability w.r.t. insider corruptions of RS*, there exist adversaries* $\mathcal{B}_i$ *with* $\mathbf{Time}(\mathcal{A}) \leq \min_{i\in[5]}\{\mathbf{Time}(\mathcal{B}_i)\} - Q_{sig} \cdot \mathsf{poly}(\lambda)$*, such that*

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{B}_5}(\lambda)$$

, where Q_{sig} *is the total number of* $\mathcal{A}$*'s signing queries.*

Proof of Theorem 5.2 Let $\mathcal{A}$ be a PPT adversary that attacks the unforgeability w.r.t. insider corruptions of RS. We proceed the proof via a sequence of games. We introduce the following six games $\mathbf{Game}_i$ for $i \in [0, 5]$.

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original game of the unforgeability w.r.t. insider corruptions for RS. The detailed description is as follows.

1. The challenger $\mathcal{C}$ firstly proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{C}$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0, 1\}$.
   (b) For all $i \in [n]$, $\mathcal{C}$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$ and samples $pk_i \leftarrow \mathcal{PK}$.
   (c) For all $i \in [n]$, $\mathcal{C}$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{C}$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{C}$ proceeds as follows:

   Signing queries.

   (a) $\mathcal{C}$ computes $\sigma_m \leftarrow \mathsf{Sign}(sk_{j\alpha_j}, m\|\mathsf{R})$, $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, and $\tau \leftarrow \mathsf{Open}(hk_{\alpha_j}, shk_{\alpha_j}, \mathsf{R}, j)$.
   (b) $\mathcal{C}$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_{\alpha_j} \leftarrow \mathsf{Enc}(pk_j, \sigma_m; r^{\mathsf{Enc}})$, samples $c_{1\oplus\alpha_j} \leftarrow \mathcal{CT}^{\mathsf{PKE}}$, and sets $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$.
   (c) If $\alpha_j = 0$ holds, then $\mathcal{C}$ sets $w := (j, rvk_j, (\sigma_m, \bot), (r^{\mathsf{Enc}}, \bot), (\tau, \bot))$. Otherwise, $\mathcal{C}$ sets $w := (j, rvk_j, (\bot, \sigma_m), (\bot, r^{\mathsf{Enc}}), (\bot, \tau))$.
   (d) $\mathcal{C}$ computes $\pi \leftarrow \mathsf{Prove}(x, w)$, sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

   Corruption queries.

   $\mathcal{C}$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. $\mathcal{A}$ outputs a tuple $(\mathsf{R}^*, m^*, \sigma^*)$.

Game$_1$: **Game**$_1$ is identical to **Game**$_0$ except for the following change. When responding to a signing query (j, R, m), $\mathcal{C}$ generates both signatures $\sigma_{\alpha_j} \leftarrow \mathsf{Sign}(sk_{j,\alpha_j}, m\|\mathsf{R})$ and $\sigma_{1\oplus\alpha_j} \leftarrow \mathsf{Sign}(sk_{j,1\oplus\alpha_j}, m\|\mathsf{R})$ and computes $c_{\alpha_j} \leftarrow \mathsf{Enc}(pk_j, \sigma_{\alpha_j}; r^{\mathsf{Enc}}_{\alpha_j})$ and $c_{1\oplus\alpha_j} \leftarrow \mathsf{Enc}(pk_j, \sigma_{1\oplus\alpha_j}; r^{\mathsf{Enc}}_{1\oplus\alpha_j})$, where $r^{\mathsf{Enc}}_{\alpha_j}, r^{\mathsf{Enc}}_{1\oplus\alpha_j} \leftarrow \mathcal{R}^{\mathsf{Enc}}$. Note that by this change, $c_0 \leftarrow \mathsf{Enc}(pk_j, \sigma_0)$ and $c_1 \leftarrow \mathsf{Enc}(pk_j, \sigma_1)$ holds. That is, c_0 and c_1 does not depend on the randomness α_j, and thus c_0 and c_1 has no information about α_j for all $j \in [n]$.

Game$_2$: **Game**$_2$ is identical to **Game**$_1$ except for the following change. When responding to a signing query (j, R, m), $\mathcal{C}$ computes both $\tau_0 \leftarrow \mathsf{Open}(hk_0, shk_0, \mathsf{R}, j)$ and $\tau_1 \leftarrow \mathsf{Open}(hk_1, shk_1, \mathsf{R}, j)$, and sets a witness w as $w := (j, rvk_j, (\sigma_0, \sigma_1), (r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1), (\tau_0, \tau_1))$.

Game$_3$: **Game**$_3$ is identical to **Game**$_2$ except that we change how to generate public keys of PKE. Concretely, we generate $(pk_i, sk_i) \leftarrow \mathsf{KG}(1^\lambda)$ instead of $pk_i \leftarrow \mathcal{PK}$ for all $i \in [n]$.

Game$_4$: **Game**$_4$ is identical to **Game**$_3$ except that we require an additional condition for the success condition of $\mathcal{A}$. More precisely, we require a forgery $(\mathsf{R}^*, m^*, \sigma^* = (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1))$ output by $\mathcal{A}$ to satisfy $x^* \in \mathcal{L}$, where $x^* := (m^*, c^*_0, c^*_1, hk^*_0, hk^*_1, h^*_0, h^*_1, \mathsf{R}^*)$.

Game$_5$: **Game**$_5$ is identical to **Game**$_4$ except that we require an additional condition for the success condition of $\mathcal{A}$. More precisely, we require a forgery $(\mathsf{R}^*, m^*, \sigma^* = (\pi^*, c^*_0, c^*_1, hk^*_0, hk^*_1))$ output by $\mathcal{A}$ to satisfy $(1 = \mathsf{Ver}(vk_{j0}, m^*\|\mathsf{R}^*, \sigma'_0)) \vee (1 = \mathsf{Ver}(vk_{j1}, m^*\|\mathsf{R}^*, \sigma'_1))$, where $\sigma'_0 = \mathsf{Dec}(pk_j, dk_j, c^*_0)$ and $\sigma'_1 = \mathsf{Dec}(pk_j, dk_j, c^*_1)$, for some $j \in [|\mathsf{R}^*|]$.

Let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ succeeds in outputting a tuple $(\mathsf{R}^*, m^*, \sigma^*)$ satisfying $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ in **Game**$_i$ for $i \in [0, 5]$. By using triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Succ}_0] \leq \sum_{i=0}^{4} |\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]| + \Pr[\mathbf{Succ}_5].$$

It remains to show how each $|\Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}]|$ for $i \in [0, 4]$ and $\Pr[\mathbf{Succ}_5]$ are upper-bounded. To this end, we show the following lemmata.

- There exists an adversary $\mathcal{B}_1$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$ (Lemma 5.10).
- There exists an adversary $\mathcal{B}_2$ against the computational witness indistinguishability of NIWI such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$ (Lemma 5.11).
- There exists an adversary $\mathcal{B}_3$ against the pseudorandomness of public keys of PKE such that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda)$ (Lemma 5.12).
- There exists an adversary $\mathcal{B}_4$ against the perfect soundness of NIWI such that $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda)$ (Lemma 5.13).
- $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = 0$ holds due to the correctness of PKE and the somewhere perfectly binding of SPBH (Lemma 5.14).
- There exists an adversary $\mathcal{B}_5$ against the MU-EUF-CMA security of SIG such that $\Pr[\mathbf{Succ}_5] = 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{B}_5}(\lambda)$ (Lemma 5.15).

Lemma 5.10. *There exists an adversary $\mathcal{B}_1$ against the pseudorandomness of ciphertexts of PKE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$.*

Proof of Lemma 5.10 We construct an adversary $\mathcal{B}_1$ that attacks the pseudorandomness of ciphertexts of PKE so that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| = \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $(r_i^{\mathsf{LKG}})_{i\in[n]}$ from the challenger, $\mathcal{B}_1$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_1$ samples randomnesses $(r_{i0}^{\mathsf{Gen}}, r_{i1}^{\mathsf{Gen}}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0,1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_1$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{LKG}.
   (c) For all $i \in [n]$, $\mathcal{B}_1$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_1$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{B}_1$ proceeds as follows:

   Signing queries.

   (a) $\mathcal{B}_1$ computes both $\sigma_0 \leftarrow \mathsf{Sign}(sk_{j0}, m\|\mathsf{R})$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{j1}, m\|\mathsf{R})$.

(b) $\mathcal{B}_1$ computes $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, and $\tau \leftarrow \mathsf{Open}(hk_{\alpha_j}, shk_{\alpha_j}, \mathsf{R}, j)$.

(c) $\mathcal{B}_1$ samples a randomness $r^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and computes $c_{\alpha_j} \leftarrow \mathsf{Enc}(pk_j, \sigma_m; r^{\mathsf{Enc}})$.

(d) $\mathcal{B}_1$ sets $m^* := \sigma_{1 \oplus \alpha_j}$ and queries (j, m^*) to the challenger. Upon receiving c^* from the challenger, $\mathcal{B}_1$ sets $c_{1 \oplus \alpha_j} := c^*$ and $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$.

(e) If $\alpha_j = 0$ holds, then $\mathcal{B}_1$ sets $w := (j, rvk_j, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau, \bot))$. Otherwise, $\mathcal{B}_1$ sets $w := (j, rvk_j, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau))$.

(f) $\mathcal{B}_1$ computes a proof $\pi \leftarrow \mathsf{Prove}(x, w)$, sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

Corruption queries.

$\mathcal{B}_1$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}_1$ outputs $\beta' := 1$ to the challenger and terminates if $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ holds. Otherwise, $\mathcal{B}_1$ outputs $\beta' := 0$ to the challenger and terminates.

In the following, let β be the challenge bit for $\mathcal{B}_1$. We can see that $\mathcal{B}_1$ perfectly simulates $\mathbf{Game}_0$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$. This ensures that the probability that $\mathcal{B}_1$ outputs 1 given the challenge ciphertext $c^* \leftarrow \mathcal{CT}^{\mathsf{PKE}}$ is equal to the probability that $\mathcal{A}$ outputs 1 in $\mathbf{Game}_0$. That is, $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathbf{Succ}_0]$ holds. On the other hand, $\mathcal{B}_1$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pk_j, m^*)$. This ensures that the probability that $\mathcal{B}_1$ outputs 1 given the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pk_j, m^*)$ is equal to the probability that $\mathcal{A}$ outputs 1 in $\mathbf{Game}_1$. That is, $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathbf{Succ}_1]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda) = 2 \cdot |\Pr[b = b'] - \frac{1}{2}| = |\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]| = |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]|$. □ **(Lemma 5.10)**

Lemma 5.11. *There exists an adversary $\mathcal{B}_2$ against the computational witness indistinguishability of* NIWI *such that* $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$.

Proof of Lemma 5.11 We construct an adversary $\mathcal{B}_2$ that attacks the computational witness indistinguishability of NIWI so that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_2$ proceeds as follows:

(a) For all $i \in [n]$, $\mathcal{B}_2$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r^{\mathsf{LKG}}_i \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0, 1\}$.

(b) For all $i \in [n]$, $\mathcal{B}_2$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i0}^{\mathsf{Gen}})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r_{i1}^{\mathsf{Gen}})$, and $pk_i \leftarrow \mathcal{PK}$ under the randomness r_i^{key}.

(c) For all $i \in [n]$, $\mathcal{B}_2$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.

(d) $\mathcal{B}_2$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.

2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{B}_2$ proceeds as follows:

Signing queries.

(a) $\mathcal{B}_2$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{j0}, m\|\mathsf{R})$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{j1}, m\|\mathsf{R})$.

(b) $\mathcal{B}_2$ computes $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, $\tau_0 \leftarrow \mathsf{Open}(hk_0, shk_0, \mathsf{R}, j)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1, shk_1, \mathsf{R}, j)$.

(c) $\mathcal{B}_2$ samples randomnesses $r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0 \leftarrow \mathsf{Enc}(pk_j, \sigma_0; r_0^{\mathsf{Enc}})$ and $c_1 \leftarrow \mathsf{Enc}(pk_j, \sigma_1; r_1^{\mathsf{Enc}})$, and sets $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$.

(d) If $\alpha_j = 0$ holds, then $\mathcal{B}_2$ sets $w_0 := (j, rvk_j, (\sigma_0, \bot), (r^{\mathsf{Enc}}, \bot), (\tau, \bot))$. Otherwise, $\mathcal{B}_1$ sets $w_0 := (j, rvk_j, (\bot, \sigma_1), (\bot, r^{\mathsf{Enc}}), (\bot, \tau))$.

(e) $\mathcal{B}_2$ sets $w_1 := (j, rvk_j, (\sigma_0, \sigma_1), (r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}}), (\tau_0, \tau_1))$.

(f) $\mathcal{B}_2$ makes a query (x, w_0, w_1) to its oracle and gets the corresponding proof π.

(g) $\mathcal{B}_2$ sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

Corruption queries

$\mathcal{B}_2$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}_2$ outputs 1 to the challenger and terminates if $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ holds. Otherwise, $\mathcal{B}_2$ outputs 0 to the challenger and terminates.

We can see that $\mathcal{B}_2$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_0$. This ensures that the probability that $\mathcal{B}_2$ outputs 1 given the proof π from the oracle $\mathcal{O}_0$ is equal to the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That is, $\Pr[\mathcal{B}_2^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_1]$ holds. On the other hand, $\mathcal{B}_2$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the proof π from the oracle $\mathcal{O}_1$. This ensures that the probability that $\mathcal{B}_2$ outputs 1 given the proof π from the oracle $\mathcal{O}_1$ is equal to the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[\mathcal{B}_2^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1] = \Pr[\mathbf{Succ}_2]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) = |\Pr[\mathcal{B}_2^{\mathcal{O}_0(\cdot,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}_2^{\mathcal{O}_1(\cdot,\cdot,\cdot)}(1^\lambda) = 1]| = |\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]|$. □ **(Lemma 5.11)**

Lemma 5.12. *There exists an adversary $\mathcal{B}_3$ against the pseudorandomness of keys of* PKE *such that* $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda)$.

Proof of Lemma 5.12 We construct an adversary $\mathcal{B}_3$ that attacks the pseudorandomness of keys of PKE so that $|\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]| = \mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving $\mathbf{pk} := (pk_1, \cdots, pk_n)$ from the challenger, $\mathcal{B}_3$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_3$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$ and $\alpha_i \leftarrow \{0,1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_3$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$ and $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$.
   (c) For all $i \in [n]$, $\mathcal{B}_3$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_3$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{B}_3$ proceeds as follows:

   Signing queries.

   (a) $\mathcal{B}_3$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{j0}, m\|\mathsf{R})$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{j1}, m\|\mathsf{R})$.
   (b) $\mathcal{B}_3$ computes $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, $\tau_0 \leftarrow \mathsf{Open}(hk_0, shk_0, \mathsf{R}, j)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1, shk_1, \mathsf{R}, j)$.
   (c) $\mathcal{B}_3$ samples randomnesses $r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1 \leftarrow \mathcal{R}^{\mathsf{Enc}}$ and computes $c_0 \leftarrow \mathsf{Enc}(pk_j, \sigma_0; r^{\mathsf{Enc}}_0)$ and $c_1 \leftarrow \mathsf{Enc}(pk_j, \sigma_1; r^{\mathsf{Enc}}_1)$.
   (d) $\mathcal{B}_3$ sets $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$ and $w := (j, rvk_j, (\sigma_0, \sigma_1), (r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1), (\tau_0, \tau_1))$, and computes $\pi \leftarrow \mathsf{Prove}(x, w)$.
   (e) $\mathcal{B}_3$ sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

   Corruption queries.

   $\mathcal{B}_3$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.
3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}_3$ outputs 1 to the challenger and terminates if $(1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)) \wedge (\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}) \wedge ((\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}})$ holds. Otherwise, $\mathcal{B}_3$ outputs 0 to the challenger and terminates.

In the following, let b be the challenge bit for $\mathcal{B}_3$. We can see that $\mathcal{B}_3$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the public keys $pk_i \leftarrow \mathcal{PK}$ for all $i \in [n]$ from the challenger. This ensures that the probability that $\mathcal{B}_3$ outputs 1 given the the public keys $pk_i \leftarrow \mathcal{PK}$ for all

$i \in [n]$ is equal to the probability that $\mathcal{A}$ outputs 1 in $\mathbf{Game}_2$. That is, $\Pr[1 = \mathcal{B}_3(\mathbf{pk})|b = 1] = \Pr[\mathbf{Succ}_2]$ holds. On the other hand, $\mathcal{B}_3$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$ if it receives the public keys $(pk_i, dk_i) \leftarrow \mathsf{KG}(1^\lambda)$ for all $i \in [n]$. This ensures that the probability that $\mathcal{B}_3$ outputs 1 given the public keys $(pk_i, dk_i) \leftarrow \mathsf{KG}(1^\lambda)$ for all $i \in [n]$ is equal to the probability that $\mathcal{A}$ outputs 1 in $\mathbf{Game}_3$. That is, $\Pr[1 = \mathcal{B}_3(\mathbf{pk})|b = 0] = \Pr[\mathbf{Succ}_3]$ holds. Therefore, it holds that $\mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda) = 2 \cdot |\Pr[b = b'] - \frac{1}{2}| = |\Pr[1 = \mathcal{B}_3(\mathbf{pk})|b = 1] - \Pr[1 = \mathcal{B}_3(\mathbf{pk})|b = 0]| = |\Pr[\mathbf{Succ}_2] - \Pr[\mathbf{Succ}_3]|$. □ **(Lemma 5.12)**

Lemma 5.13. *There exists an adversary $\mathcal{B}_4$ against the perfect soundness of* NIWI *such that* $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda)$.

Proof of Lemma 5.13 For $i \in \{3, 4\}$, let $\mathbf{Bad}_i$ be the event that $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ satisfying $(x^* \notin \mathcal{L}) \wedge (1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*))$ in $\mathbf{Game}_i$. (We call such a forgery a *bad forgery*.) $\mathbf{Game}_3$ proceeds identically to $\mathbf{Game}_4$ unless $\mathbf{Bad}_3$ happens. Therefore, the inequality $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| \leq \Pr[\mathbf{Bad}_3] = \Pr[\mathbf{Bad}_4]$.

In the following, we show that one can construct a PPT adversary $\mathcal{B}_4$ that attacks the perfect soundness of NIWI so that $\Pr[\mathbf{Bad}_3] = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda)$, using the adversary $\mathcal{A}$.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}_4$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_4$ samples randomnesses $(r^{\mathsf{Gen}}_{i0}, r^{\mathsf{Gen}}_{i1}) \leftarrow (\mathcal{R}^{\mathsf{Gen}})^2$, $r^{\mathsf{LKG}}_i \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$, and $\alpha_i \leftarrow \{0, 1\}$.
   (b) For all $i \in [n]$, $\mathcal{B}_4$ generates $(vk_{i0}, sk_{i0}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i0})$, $(vk_{i1}, sk_{i1}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i1})$, and $(pk_i, sk_i) \leftarrow \mathsf{KG}(1^\lambda; r^{\mathsf{LKG}}_i)$.
   (c) For all $i \in [n]$, $\mathcal{B}_4$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (\alpha_i, sk_{i\alpha_i}, rvk_i)$.
   (d) $\mathcal{B}_4$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
2. When $\mathcal{A}$ makes a signing query (j, R, m), $\mathcal{B}_4$ proceeds as follows:

   Signing queries.

   (a) $\mathcal{B}_4$ computes $\sigma_0 \leftarrow \mathsf{Sign}(sk_{j0}, m\|\mathsf{R})$ and $\sigma_1 \leftarrow \mathsf{Sign}(sk_{j1}, m\|\mathsf{R})$.
   (b) $\mathcal{B}_4$ computes $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, $\tau_0 \leftarrow \mathsf{Open}(hk_0, shk_0, \mathsf{R}, j)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1, shk_1, \mathsf{R}, j)$.
   (c) $\mathcal{B}_4$ samples randomnesses $r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1 \leftarrow \mathcal{R}^{\mathsf{Enc}}$, computes $c_0 \leftarrow \mathsf{Enc}(pk_j, \sigma_0; r^{\mathsf{Enc}}_0)$ and $c_1 \leftarrow \mathsf{Enc}(pk_j, \sigma_1; r^{\mathsf{Enc}}_1)$, and sets $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$ and $w := (j, rvk_j, (\sigma_0, \sigma_1), (r^{\mathsf{Enc}}_0, r^{\mathsf{Enc}}_1), (\tau_0, \tau_1))$.

(d) $\mathcal{B}_4$ computes $\pi \leftarrow \mathsf{Prove}(x, w)$, sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

Corruption queries.

$\mathcal{B}_4$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}_4$ parses $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$ and computes $h_0^* \leftarrow \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$ and $h_1^* \leftarrow \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$. Then, $\mathcal{B}_4$ sets $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$ and outputs (x^*, π^*) to the challenger and terminates.

From the above construction of $\mathcal{B}_4$, it is easy to see that $\mathcal{B}_4$ perfectly simulates $\mathbf{Game}_3$ for $\mathcal{A}$. Recall that the success condition of $\mathcal{B}_4$ is to output a statement and a proof (x^*, π^*) satisfying $(x^* \notin \mathcal{L}) \wedge (1 = \mathsf{Verify}(x^*, \pi^*))$. If $\mathcal{A}$ makes a bad forgery $(\mathsf{R}^*, m^*, \sigma^*)$, then $(x^* \notin \mathcal{L}) \wedge (1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*))$ hold. Due to the construction of RS, the condition $1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)$ implies that the condition $1 = \mathsf{Verify}(x^*, \pi^*)$ holds. Thus, when $\mathcal{A}$ makes such a bad forgery $(\mathsf{R}^*, m^*, \sigma^*)$, $\mathcal{B}_4$ achieves its success condition by returning (x^*, π^*) to its challenger. From the above arguments, the probability that $\mathcal{A}$ makes such a bad forgery is equal to the probability that $\mathcal{B}_4$ breaks the perfect soundness of NIWI. Hence, we have $\Pr[\mathbf{Bad}_3] = \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda)$, which in turn implies $|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_4]| \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda)$. □ **(Lemma 5.13)**

Lemma 5.14. $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = 0$ *holds.*

Proof of Lemma 5.14 Firstly, by the definition, the valid forgery $(\mathsf{R}^*, m^*, \sigma^*(= (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)))$ output by $\mathcal{A}$ must satisfy the condition $1 = \mathsf{RVer}(\mathsf{R}^*, m^*, \sigma^*)$. That is, the condition $1 = \mathsf{Verify}(x^*, \pi^*)$ holds, where $h_0^* = \mathsf{Hash}(hk_0^*, \mathsf{R}^*)$, $h_1^* = \mathsf{Hash}(hk_1^*, \mathsf{R}^*)$, and $x^* := (m^*, c_0^*, c_1^*, hk_0^*, hk_1^*, h_0^*, h_1^*, \mathsf{R}^*)$.

Moreover, by the change on $\mathbf{Game}_4$, $x^* \in \mathcal{L}$ holds for the forgery output by $\mathcal{A}$. Thus, by the definition of the language $\mathcal{L}$, we can see that there exists $w^* := (j, rvk^* = (vk_0^*, vk_1^*, pk^*), (\sigma_0, \sigma_1), (r_0, r_1), (\tau_0, \tau_1))$ such that the condition

$$\Big((1 = \mathsf{Ver}(vk_0^*, m^* \| \mathsf{R}^*, \sigma_0)) \wedge (c_0^* = \mathsf{Enc}(pk^*, \sigma_0; r_0)) \wedge (1 = \mathsf{HVer}(hk_0^*, h_0^*, j, rvk^*, \tau_0))\Big)$$
$$\bigvee$$
$$\Big((1 = \mathsf{Ver}(vk_1^*, m^* \| \mathsf{R}^*, \sigma_1)) \wedge (c_1^* = \mathsf{Enc}(pk^*, \sigma_1; r_1)) \wedge (1 = \mathsf{HVer}(hk_1^*, h_1^*, j, rvk^*, \tau_1))\Big)$$

holds.

From the above conditions, $(h_0^* = \mathsf{Hash}(hk_0^*, \mathsf{R}^*)) \wedge (1 = \mathsf{HVer}(hk_0, h_0, j, rvk, \tau_0))$ or $(h_1^* = \mathsf{Hash}(hk_1^*, \mathsf{R}^*)) \wedge (1 = \mathsf{HVer}(hk_1, h_1, j, rvk, \tau_1))$ always holds here. Hence, by the somewhere

perfectly binding of SPBH, $rvk^* = rvk_j$ holds, where $rvk_j \in \mathsf{R}^*$. The condition $rvk^* = rvk_j$ implies $vk_0^* = vk_{j0}$, $vk_1^* = vk_{j1}$, and $pk^* = pk_j$. Thus, $(1 = \mathsf{Ver}(vk_{j0}, m^* \| \mathsf{R}^*, \sigma_0)) \vee (1 = \mathsf{Ver}(vk_{j1}, m^* \| \mathsf{R}^*, \sigma_1))$ holds now.

Furthermore, by using the correctness of PKE, $c_0^* = \mathsf{Enc}(pk_j, \sigma_0; r_0)$ and $c_1^* = \mathsf{Enc}(pk_j, \sigma_1; r_1)$ implies $\sigma_0 = \mathsf{Dec}(pk_j, dk_j, c_0^*)$ and $\sigma_1 = \mathsf{Dec}(pk_j, dk_j, c_1^*)$, respectively. Thus, $(1 = \mathsf{Ver}(vk_{j0}, m^* \| \mathsf{R}^*, \sigma_0)) \vee (1 = \mathsf{Ver}(vk_{j1}, m^* \| \mathsf{R}^*, \sigma_1))$, where $\sigma_0 = \mathsf{Dec}(pk_j, dk_j, c_0^*)$ and $\sigma_1 = \mathsf{Dec}(pk_j, dk_j, c_1^*)$, holds in $\mathbf{Game}_4$. Hence, the change between $\mathbf{Game}_4$ and $\mathbf{Game}_5$ does not affect the view of $\mathcal{A}$. Therefore, $|\Pr[\mathbf{Succ}_4] - \Pr[\mathbf{Succ}_5]| = 0$ holds. □ **(Lemma 5.14)**

Lemma 5.15. *There exists an adversary $\mathcal{B}_5$ against the MU-EUF-CMA security of SIG such that* $\Pr[\mathbf{Succ}_5] \leq 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{B}_5}(\lambda)$.

Proof of Lemma 5.15 We construct an adversary $\mathcal{B}_5$ that attacks the MU-EUF-CMA security of SIG so that $\Pr[\mathbf{Succ}_5] \leq 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{B}_5}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving n verification keys $(vk_i)_{i\in[n]}$ from the challenger, $\mathcal{B}_5$ proceeds as follows:
   (a) For all $i \in [n]$, $\mathcal{B}_5$ samples $\alpha_i \leftarrow \{0,1\}$ and sets $vk_{i\alpha_i} := vk_i$.
   (b) For all $i \in [n]$, $\mathcal{B}_5$ samples randomnesses $r^{\mathsf{Gen}}_{i,(1\oplus\alpha_i)} \leftarrow \mathcal{R}^{\mathsf{Gen}}$ and $r_i^{\mathsf{LKG}} \leftarrow \mathcal{R}^{key}_{\mathsf{PKE}}$ and generates one side of verification/signing keys $(vk_{i(1\oplus\alpha_i)}, sk_{i(1\oplus\alpha_i)}) \leftarrow \mathsf{Gen}(1^\lambda; r^{\mathsf{Gen}}_{i,(1\oplus\alpha_i)})$ for SIG and public/decryption keys $(pk_i, dk_i) \leftarrow \mathsf{KG}(1^\lambda; r_i^{\mathsf{LKG}})$ for PKE.
   (c) For all $i \in [n]$, $\mathcal{B}_5$ sets $rvk_i := (vk_{i0}, vk_{i1}, pk_i)$ and $rsk_i := (1 \oplus \alpha_i, sk_{i(1\oplus\alpha_i)}, rvk_i)$.
   (d) $\mathcal{B}_5$ gives $\mathbf{rvk} := (rvk_1, \cdots, rvk_n)$ to $\mathcal{A}$ and sets $\mathsf{S}_{\mathsf{Sig}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
2. When $\mathcal{A}$ makes a signing query (j, R, m) and a corruption query j, $\mathcal{B}_5$ proceeds as follows:

   Signing queries.

   (a) $\mathcal{B}_5$ computes $\sigma_{1\oplus\alpha_j} \leftarrow \mathsf{Sign}(sk_{j(1\oplus\alpha_j)}, m \| \mathsf{R})$.
   (b) $\mathcal{B}_5$ makes a signing query $(j, m\|\mathsf{R})$ for its challenger, gets a corresponding signature σ, and sets $\sigma_{\alpha_j} := \sigma$.
   (c) $\mathcal{B}_5$ computes $(hk_0, shk_0) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $(hk_1, shk_1) \leftarrow \mathsf{HGen}(1^\lambda, |\mathsf{R}|, j)$, $h_0 \leftarrow \mathsf{Hash}(hk_0, \mathsf{R})$, $h_1 \leftarrow \mathsf{Hash}(hk_1, \mathsf{R})$, $\tau_0 \leftarrow \mathsf{Open}(hk_0, shk_0, \mathsf{R}, j)$, and $\tau_1 \leftarrow \mathsf{Open}(hk_1, shk_1, \mathsf{R}, j)$.
   (d) $\mathcal{B}_5$ samples randomnesses $r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}} \leftarrow \mathcal{R}^{\mathsf{Enc}}$, and computes $c_0 \leftarrow \mathsf{Enc}(pk_j, \sigma_0; r_0^{\mathsf{Enc}})$ and $c_1 \leftarrow \mathsf{Enc}(pk_j, \sigma_1; r_1^{\mathsf{Enc}})$.

(e) $\mathcal{B}_5$ sets $x := (m, c_0, c_1, hk_0, hk_1, h_0, h_1, \mathsf{R})$ and $w := (j, rvk_j, (\sigma_0, \sigma_1), (r_0^{\mathsf{Enc}}, r_1^{\mathsf{Enc}}), (\tau_0, \tau_1))$.

(f) $\mathcal{B}_5$ computes $\pi \leftarrow \mathsf{Prove}(x, w)$, sets $\sigma := (\pi, c_0, c_1, hk_0, hk_1)$, gives σ to $\mathcal{A}$, and appends (j, R, m) to $\mathsf{S}_{\mathsf{Sig}}$.

Corruption queries.

When $\mathcal{A}$ makes a corruption query j, $\mathcal{B}_5$ gives rsk_j to $\mathcal{A}$ and appends rvk_j to $\mathsf{S}_{\mathsf{Corr}}$.

3. When $\mathcal{A}$ outputs a forgery $(\mathsf{R}^*, m^*, \sigma^*)$ and terminates, $\mathcal{B}_5$ parses $\sigma^* := (\pi^*, c_0^*, c_1^*, hk_0^*, hk_1^*)$, computes $\sigma_{j0} \leftarrow \mathsf{Dec}(pk_j, dk_j, c_0^*)$ and $\sigma_{j1} \leftarrow \mathsf{Dec}(pk_j, dk_j, c_1^*)$, and checks whether $(1 = \mathsf{Ver}(vk_{j0}, m^* \| \mathsf{R}^*, \sigma_{j0})) \vee (1 = \mathsf{Ver}(vk_{j1}, m^* \| \mathsf{R}^*, \sigma_{j1}))$ holds for all $j \in [|\mathsf{R}^*|]$. If this holds for some $j \in [|\mathsf{R}^*|]$, then $\mathcal{B}_5$ returns $(j, m^* \| \mathsf{R}^*, \sigma_{j\alpha_j})$ to its challenger and terminates. Otherwise, $\mathcal{B}_5$ gives up and terminates.

We can see that $\mathcal{B}_5$ perfectly simulates $\mathbf{Game}_5$ for $\mathcal{A}$. In the following, we show that $\mathcal{B}_5$ can make a valid forgery $(j, m^* \| \mathsf{R}^*, \sigma_{j\alpha_j})$ satisfying $(1 = \mathsf{Ver}(vk_j (= vk_{j\alpha_j}), m^* \| \mathsf{R}^*, \sigma_{j\alpha_j})) \wedge ((m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_j)$ with the probability $\frac{1}{2}$ if $\mathcal{A}$ makes a valid forgery $(\mathsf{R}^*, m^*, \sigma^*)$. Here, by the definition, if $\mathcal{A}$ outputs a valid forgery $(\mathsf{R}^*, m^*, \sigma^*)$, then $(\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}}$ holds. Thus, we can see that $(m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_j$ holds by the fact $(\cdot, \mathsf{R}^*, m^*) \notin \mathsf{S}_{\mathsf{Sig}}$.

Next, we can say that $rvk^* = rvk_j$ holds now from the same argument in the proof of Lemma 5.14. Hence, $\mathcal{A}$ does not make a corruption query j because $rvk^* = rvk_j$ and $\mathsf{R}^* \subseteq \mathbf{rvk} \backslash \mathsf{S}_{\mathsf{Corr}}$ hold. Therefore, $\mathcal{A}$ does not get the corresponding signing key $rsk_j := ((1 \oplus \alpha_j), sk_{j(1 \oplus \alpha_j)}, rvk_j)$ which is the only element containing the information about α_j in $\mathbf{Game}_5$. Thus, the information of α_j is information-theoretically hidden for $\mathcal{A}$. Furthermore, by the change from $\mathbf{Game}_4$ to $\mathbf{Game}_5$, $(1 = \mathsf{Ver}(vk_{j0}, m^* \| \mathsf{R}^*, \sigma_{j0})) \vee (1 = \mathsf{Ver}(vk_{j1}, m^* \| \mathsf{R}^*, \sigma_{j1}))$ hold for all $j \in [|\mathsf{R}^*|]$, where $\sigma_{j0} = \mathsf{Dec}(pk_j, dk_j, c_0^*)$ and $\sigma_{j1} = \mathsf{Dec}(pk_j, dk_j, c_1^*)$. Hence, $\Pr[1 = \mathsf{Ver}(vk_{j0}, m^* \| \mathsf{R}^*, \sigma_{j0})] = \Pr[1 = \mathsf{Ver}(vk_{j1}, m^* \| \mathsf{R}^*, \sigma_{j1})] = \frac{1}{2}$ holds.

From the above arguments, we can show that $\mathcal{B}_5$ can makes a valid forgery with the probability $\frac{1}{2}$ if $\mathcal{A}$ makes a valid forgery. That is, $\Pr[(1 = \mathsf{Ver}(vk_j (= vk_{j\alpha_j}), m^* \| \mathsf{R}^*, \sigma_{j\alpha_j})) \wedge ((m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_j) | \mathbf{Succ}_5] = \frac{1}{2}$ holds.

In the following, let $\mathbf{Win}$ be the event that $(1 = \mathsf{Ver}(vk_j, m^* \| \mathsf{R}^*, \sigma_{j\alpha_j})) \wedge ((m^* \| \mathsf{R}^*, \cdot) \notin \mathsf{S}_j)$ holds in $\mathbf{Game}_5$. We have the following inequality.

$$\mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG}, \mathcal{B}_5}(\lambda) = \Pr[\mathbf{Win}] \geq \Pr[\mathbf{Win} \wedge \mathbf{Succ}_5] = \Pr[\mathbf{Win} | \mathbf{Succ}_5] \cdot \Pr[\mathbf{Succ}_5] = \frac{1}{2} \cdot \Pr[\mathbf{Succ}_5]$$

That is, $\Pr[\mathbf{Succ}_5] \leq 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG}, \mathcal{B}_5}(\lambda)$ holds. □ **(Lemma 5.15)**

Putting everything together, we obtain $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ct}}_{\mathsf{PKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\mathsf{NIWI},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{key}}_{\mathsf{PKE},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIWI},\mathcal{B}_4}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}unf}}_{\mathsf{SIG},\mathcal{B}_5}(\lambda)$.

Since PKE satisfies pseudorandomness of public keys and pseudorandomness of ciphertexts, NIWI satisfies computational witness indistinguishability and perfect soundness, and SIG is MU-EUF-CMA secure, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{RS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, RS satisfies unforgeability w.r.t. insider corruptions. □ **(Theorem 5.2)**

Chapter 6

Round-Optimal Deniable Ring Authentication in the RO model

In this section, we provide the first two-round concurrently deniable ring authentication schemes with an optimal efficiency in the random oracle (RO) model [BR93]. Towards our goal, as our main technical result, we propose a new generic construction of two-round concurrently deniable ring authentication in the RO model based on any IND-CPA secure BE scheme.[1] This is achieved by the following two steps.

- As the first step, in Section 6.3, we formally define *plaintext awareness* for BE, then provide a generic transformation from an IND-CPA secure BE scheme into a BE scheme satisfying IND-CCA security and plaintext awareness in the RO model, and *decryption uniqueness*[2].
- As the second step, in Section 6.4, we give a generic construction of deniable ring authentication in the RO model based on a BE scheme satisfying IND-CCA security and plaintext awareness in the RO model, and decryption uniqueness. (In Section 6.6, to better understand our scheme, we present a simple and efficient instantiation of our generic construction of two-round concurrently deniable ring authentication.)

We give a technical overview for these contributions in Section 6.2.

[1] Actually, in our generic construction, we require that the underlying IND-CPA secure BE scheme satisfy a subtle additional property called *smoothness*. As mentioned in Section 2.5, many known IND-CPA secure BE schemes have smoothness unconditionally and any BE scheme can be easily converted to one satisfying this property (with essentially no overhead).

[2] This property is called *verifiability* in the previous work [HK08].

Notably, unlike the previous deniable ring authentication schemes based on BE schemes [DHIN11, YAS+12], our generic construction does not require that the underlying BE scheme satisfy decryption uniqueness. Therefore, if more efficient BE schemes satisfying only (standard) IND-CPA security are proposed in the future, we will be able to immediately obtain a more efficient two-round concurrently deniable ring authentication scheme by using our generic construction.

Through our generic construction, we obtain the first two-round concurrently deniable ring authentication scheme with optimal efficiency in the RO model instantiating the underlying IND-CPA secure BE scheme with the scheme recently proposed by Agrawal and Yamada [AY20] based on the learning with errors (LWE) assumption [Reg05] and bilinear maps in the generic bilinear group model (GBGM) [BCFG17]. Additionally, instantiating the underlying IND-CPA secure BE scheme with the more recent scheme proposed by Agrawal, Wichs, and Yamada [AWY20], we can also obtain the first two-round concurrently deniable ring authentication scheme with optimal efficiency in the RO model based on the LWE assumption and the Knowledge of OrthogonALity Assumption (KOALA) [BW19], which is a variant of the knowledge of exponent assumptions over the groups.

In addition to the above main instantiations, we further obtain the following various two-round concurrently deniable ring authentication schemes by instantiating the underlying BE scheme with schemes proposed in various previous works.

- Instantiating the underlying IND-CPA secure BE scheme with the schemes proposed by Chen, Gay, and Wee [CGW15] or Gay, Kowalczyk, and Wee [GKW18], we obtain the first two-round concurrently deniable ring authentication schemes based on the k-linear assumption (over bilinear groups) in the RO model. (Especially, the 1-linear assumption is equivalent to the symmetric external Diffie-Hellman (SXDH) assumption.) Notably, by using Chen et al.'s scheme [CGW15], we obtain a scheme in the RO model such that the communication cost (or the size of the secret key) and the number of pairing operations do not depend on the number of users, based on the k-linear assumption. Moreover, by using Gay et al.'s scheme [GKW18], we obtain a scheme in the RO model such that all of the size of the secret key, the communication cost, and the number of pairing operations do not depend on the number of users based on the k-linear assumption. (Note that Gay et al.'s scheme [GKW18] is suffered from the huge public parameter size $\mathcal{O}(n^2)$ in exchange for the efficiency for another aspects.)

- Instantiating the underlying IND-CPA secure BE scheme by using an IND-CPA secure public key encryption (PKE) scheme in a parallel way, we obtain the first two-round

concurrently deniable ring authentication scheme in the RO model under the decisional Diffie-Hellman (DDH) assumption (over a pairing-free group) (that is, the first pairing-free scheme in the RO model). Moreover, we obtain the first two-round concurrently deniable ring authentication scheme in the RO model under the LWE assumption and the learning parity with noise (LPN) assumption (that is, the first post-quantum secure schemes in the RO model).

6.1 Comparison with Previous Works

In this section, we give comparisons between our construction (and its instantiations) and previous works from two perspectives: their security properties (Figure 6.1) and their efficiencies (Figure 6.2). In these comparisons, while we cite the previous work by Yamada, Attrapadung, Santoso, Schuldt, Hanaoka, and Kunihiro [YAS+12] as one of the existing deniable ring authentication schemes, they actually proposed a deniable predicate authentication scheme, which supports more general relations than a set membership (supported by a deniable ring authentication scheme).

In Figure 6.1 and 6.2, in the column "Round", we consider the setting where an (authenticated) message is shared before an authentication process starts. In other words, we ignore the round for sharing a message when counting the number of rounds.

In Figure 6.2, in the column "Public Parameter", we suppose that the maximum number of users is fixed to n and a public parameter for n users is generated by a trusted third party in every construction. For fairness, we note that some previous constructions [ZMYH17,ZCTH17] do not need such a setup procedure. In these constructions, a public/secret key of each user is generated by itself. Moreover, for all of the sizes of a public parameter and a secret key, the communication complexity, and the number of pairing operations, we ignore the $\mathsf{poly}(\lambda)$ factors. In particular, $\mathsf{poly}(\log n)$ factor is ignored.

As can be seen in Figure 6.1, before our work, there was only one two-round concurrently deniable ring authentication scheme proposed by Zeng, Chen, Tan, and He [ZCTH17]. Their scheme achieves concurrent deniability based on the Diffie-Hellman knowledge of exponent (DHK) assumption [Dam92, BP04]. Compared to their scheme, our schemes are secure in the RO model under standard (and falsifiable) assumptions.

Next, as can be seen in Figure 6.2, regarding the size of a public parameter, it is linear in n in all of the previous schemes. Regarding the communication complexity, some of the previously proposed schemes [DHIN11, YAS+12] achieved constant-size communication (that

is, it is independent of n) thanks to the compression property on a ciphertext of the underlying BE schemes, which is based on bilinear groups. Regarding the number of pairing operations, some of the previous constructions [DHIN11, YAS+12] require only a constant number of pairing operations, while the other constructions [ZMYH17, ZCTH17] require $\mathcal{O}(n)$ pairing operations.

Scheme	Round	Deniability		Soundness	Source Hiding	Assumption
Naor [Nao02]	6	Seq	Corr	w/o Corr	Comp	IND-CCA PKE
Dowsley et al. [DHIN11]	4	Seq	w/o Corr	Corr ($)	Uncond	IND-CCA VBE ([BGW05])
Yamada et al. [YAS+12]	6	Seq	Corr	Corr	Uncond	IND-CCA VPE
Zeng et al. [ZMYH17] (†)	4	Seq	Corr	Corr	Uncond	PHF and NIWI ([GS08])
Zeng et al. [ZCTH17] (†)	2	Con	w/o Corr	w/o Corr	Uncond	DBDH and DHK
Ours + [ElG84, Reg05, Ale03]	2	Con	Corr	Corr	Uncond	IND-CPA BE (ROM) [DDH, LWE, LPN]

Figure 6.1: Security comparison with previous works. In the column "Deniability", Seq and Con denote sequential deniability and concurrent deniability, respectively. Moreover, Corr (or w/o Corr) denote that an adversary can (or cannot) corrupt any prover. In the column "Soundness", Corr (or w/o Corr) denote that an adversary can (or cannot) corrupt provers *except for target ones*. In the column "Source Hiding", Comp and Uncond denote computational security and unconditional security, respectively. In the column "Assumption", VBE stands for verifiable broadcast encryption, VPE stands for verifiable predicate encryption, PHF stands for projective hash function (which is also known as hash proof system) [CS02], DBDH stands for the decisional bilinear Diffie-Hellman assumption, and DHK stands for the Diffie-Hellman knowledge assumption. ($) Their construction satisfies a weaker variant of soundness, which is only selectively secure. (See Section 3.2 for the formal definitions.) (†) Their two-round constructions do not need a setup procedure.

6.2 Technical Overview

In this section, we give a technical overview for our generic construction of deniable ring authentication. Since our starting point is the two-round authentication scheme proposed by Dolev, Dwork, and Naor [DDN91], we first review their scheme briefly. We then provide an overview of how to extend their scheme into the deniable ring authentication setting.

Recap on the (Deniable) Authentication Scheme by Dolev et al. At a high level, Dolev et al. showed that the following simple two-round protocol between a prover (Alice) and a verifier (Bob) becomes a two-round authentication scheme if the underlying PKE scheme satisfies IND-CCA security. Here, let pk be Alice's public key and sk its corresponding secret key of the underlying PKE scheme and assume that Alice and Bob are authenticating a common message m, known to both parties.

Scheme	Round	Public Parameter	Secret Key	Communication	Pairing Operation
Dowsley et al. [DHIN11]	4	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Yamada et al. [YAS+12]	6	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Zeng et al. [ZMYH17]	4	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Zeng et al. [ZCTH17]	2	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Ours + [AY20] (LWE and GBGM)	2	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Ours + [AWY20] (LWE and KOALA)	2	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Ours + [GW09] (q-type)	2	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Ours + [CGW15] (k-lin)	2	$\mathcal{O}(n)$	$\mathcal{O}(t)$	$\mathcal{O}(n/t)$	$\mathcal{O}(1)$
Ours + [GKW18] (k-lin)	2	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Figure 6.2: Efficiency comparison with existing (pairing-based) schemes. n denotes the number of users. In the row "Ours + [CGW15] (k-lin)", t denotes a tradeoff parameter between the size of a secret key and the size of a communication. (Especially, for $t = 1$, the size of a secret key is $\mathcal{O}(1)$ and the size of a communication is $\mathcal{O}(n)$. On the other hand, for $t = n$, the size of a secret key is $\mathcal{O}(n)$ and the size of a communication is $\mathcal{O}(1)$.)

1. **Alice $\Leftarrow$ Bob.** This protocol starts from Bob by picking a random value t called a *token*, encrypting it with m under pk as $c \leftarrow \mathsf{PEnc}(pk, m\|t)$, where PEnc is the encryption algorithm of the underlying PKE scheme, and sending the ciphertext c to Alice as the first message.

2. **Alice $\Rightarrow$ Bob.** Upon receiving the first message c, Alice decrypts the ciphertext c under her secret key sk as $m'\|t' \leftarrow \mathsf{PDec}(sk, c)$, where PDec is the decryption algorithm of the underlying PKE scheme. If the decryption result is of the right form (that is, the first component m' of the decrypted pair corresponds to the message m that is being authenticated), then Alice sends the (decrypted) token t' to Bob as the second message. Upon receiving the second message t', Bob locally verifies that in the communication on the message m, the communicating user is Alice by checking whether $t = t'$ holds.

Regarding its soundness, since Alice is the only one who can decrypt c and c is non-malleable (due to the IND-CCA security of the underlying PKE scheme), Bob can believe that Alice is really authenticating m. Regarding its (concurrent) deniability, for an *honest* verifier, we can easily ensure that perfect deniability holds since everyone could generate the whole transcript by itself. However, for a *malicious* verifier, it is more difficult to guarantee deniability. In order to prove deniability formally, we need to construct a simulator $\mathcal{S}$ (without sk) which can produce valid transcripts interacting with the malicious verifier. Because of the lack of sk, it is not clear how $\mathcal{S}$ simulates the encrypted token t when $\mathcal{S}$ receives a ciphertext $c = \mathsf{PEnc}(pk, m\|t)$.

At first glance, one might think that we can achieve deniability on an authentication scheme by using a non-interactive zero-knowledge (NIZK) proof of knowledge [BFM88] (for proving the validity of the ciphertext c). More precisely, it seems that instead of a ciphertext $c = \mathsf{PEnc}(pk, m\|t)$, we can set an NIZK proof π as the first message. Then, by using the trapdoor of an NIZK proof of knowledge, we construct a simulator $\mathcal{S}$ who can extract the (encrypted) message m and token t (corresponding to the witness) from the proof π and set the extracted t as the second message. However, this naive approach does not make sense. An important point is that for deniability, we must ensure that any party (without having any trapdoor information) can simulate the valid transcripts for capturing the realistic deniability. Therefore, we cannot allow a simulator $\mathcal{S}$ to have some (secret) trapdoor information for simulating the transcripts.

As observed in [DGK06], taking into account the above technical problem, it was shown that we can achieve concurrent deniability for a malicious verifier by additionally requiring *plaintext awareness* against the underlying PKE scheme (without modifying the protocol) for the above two-round protocol. In a nutshell, plaintext awareness is a property capturing the intuition that no one can generate a (valid) ciphertext c without knowing the plaintext to which it will be decrypted. This property is formalized by using a special algorithm called the *plaintext extractor* $\mathcal{X}$ against an adversary $\mathcal{A}$. Given the information that $\mathcal{A}$ has as input, the plaintext extractor $\mathcal{X}$ is in the position to retrace the computations that $\mathcal{A}$ executed for generating c, and we expect $\mathcal{X}$ to output the plaintext encrypted in c. By utilizing the plaintext awareness of the underlying PKE scheme, we can overcome the above potential problem in the deniability. The main observation is that if $\mathcal{S}$ can use the plaintext extractor $\mathcal{X}$, it can know the token t encrypted in c without sk, when receiving a ciphertext c.

How to Extend the Scheme by Dolev et al. into the Ring Setting. From the previous works on deniable ring authentication [DHIN11, YAS+12], at first glance, utilizing a BE scheme [FN94, BGW05] easily provides a solution for how to extend the deniable authentication scheme by Dolev et al. into a deniable ring authentication scheme. However, as we will see in the following, it is not so straightforward to realize this idea.

BE is a cryptographic primitive that allows a sender to encrypt plaintexts to a set $\mathsf{S} \subseteq \{1, \cdots, n\}$ of authorized users so that any user in the set S can decrypt, and any (possibly colluding) set of unauthorized users can learn nothing about the plaintext, where $n \in \mathbb{N}$ denotes the number of all users. By using a BE scheme, our deniable ring authentication scheme works as follows. Here, we assume that Alice has a secret key sk_i of BE (corresponding to her index i in a set $\mathsf{S} \subseteq \{1, \cdots, n\}$) and wants to authenticate m as a member in S.

Firstly, Bob computes a BE ciphertext $c \leftarrow \mathsf{Enc}(pp, \mathsf{S}, h\|t)$ as the first message, where pp is a (valid) public parameter of BE and h is a hash value of the concatenation of m and S. Then, upon receiving the first message c from Bob, Alice gets a key t' by computing $h'\|t' \leftarrow \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ and checks whether h' is a valid hash value of $m\|\mathsf{S}$. If h' is valid, then Alice sends the (decrypted) token t' to Bob as the second message. Finally, Bob locally verifies $t = t'$ holds.

Here, similarly to Dolev et al.'s (ordinary) deniable authentication scheme, we can see that soundness holds if the underlying BE scheme satisfies IND-CCA security. However, regarding its deniability and source hiding, we have additional technical barriers explained in the following.

Firstly, on the deniability, we have to require the underlying BE scheme satisfy plaintext awareness. However, to the best of our knowledge, we do not have any plaintext aware BE scheme (and even its formal definition) so far, and thus we need to construct a plaintext aware BE scheme from scratch. We provide the first plaintext aware BE scheme as explained in Section 6.2.

Secondly, on the source hiding, one of the natural approaches is to require an additional property called *decryption uniqueness* for the underlying BE scheme, in line with the previous works on deniable ring authentication [DHIN11, YAS+12]. Intuitively, decryption uniqueness is a property ensuring that it is impossible to generate a ciphertext which is decrypted to different plaintexts among users in an authorized set. If we have decryption uniqueness, the reason why source hiding holds in our deniable ring authentication scheme is simple. The main observation is the fact that the difference for an adversary against source hiding between an interaction with a prover using sk_i and an interaction with a prover using sk_j will only possibly occur when the prover decrypts a BE ciphertext c (which is given by the verifier as the first message), where sk_i and sk_j are secret keys corresponding to the distinct users i and j, respectively. Here, due to the decryption uniqueness, the decryption result is identical no matter whether we use sk_i or sk_j. Therefore, the above difference does not affect the view of an adversary, and thus our deniable ring authentication scheme satisfies source hiding.

From these two points, we can see that the technical challenge is reduced to constructing an efficient BE scheme satisfying plaintext awareness and decryption uniqueness simultaneously. In Section 6.2, we present our solution in the RO model for this technical challenge based solely on any IND-CPA secure BE scheme. Then, as our main technical result, we can obtain a new generic construction of two-round concurrently deniable ring authentication in the RO model based solely on an IND-CPA secure BE scheme. See Section 6.4 for the details.

Achieving Plaintext Awareness and Decryption Uniqueness. In this thesis, we present an efficient transformation from any (standard) BE scheme into a BE scheme satisfying plaintext awareness and decryption uniqueness simultaneously in the RO model. In a nutshell, our BE construction is analogous to the Fujisaki-Okamoto transformation [FO99], which converts any IND-CPA secure PKE scheme into a plaintext aware and IND-CCA secure PKE scheme in the RO model. Concretely, based on an IND-CPA secure BE scheme $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$, our BE construction $\mathsf{BE}_{\mathsf{FO}} = (\mathsf{Setup}_{\mathsf{FO}}, \mathsf{Enc}_{\mathsf{FO}}, \mathsf{Dec}_{\mathsf{FO}})$ is as follows.

The setup algorithm $\mathsf{Setup}_{\mathsf{FO}}$ outputs a public parameter and a set of n secret keys $(pp, \mathbf{sk} = (sk_1, \cdots, sk_n))$ output by the underlying setup algorithm Setup.

When encrypting a plaintext m under a user set $\mathsf{S} \subseteq \{1, \cdots, n\}$, the encryption algorithm $\mathsf{Enc}_{\mathsf{FO}}$ computes a random oracle value R (used as a randomness in Enc) of a concatenation of a randomness r (for $\mathsf{Enc}_{\mathsf{FO}}$) and m (that is, $R \leftarrow \mathsf{H}_{\mathcal{RO}}(r\|m)$), where $\mathsf{H}_{\mathcal{RO}}$ is a hash function (modeled as a random oracle), and then generates a ciphertext c of $r\|m$ under the public parameter pp and the set S by using the underlying encryption algorithm Enc of BE.

When decrypting the ciphertext c under S and sk_i $(i \in \mathsf{S})$, the decryption algorithm $\mathsf{Dec}_{\mathsf{FO}}$ firstly decrypts c under S and sk_i by using Dec and gets the result $r\|m$. Then, it re-computes the random oracle value R of $r\|m$ and re-encrypts $r\|m$ under S by using the (re-computed) randomness R. If this re-encryption result agrees with c, $\mathsf{Dec}_{\mathsf{FO}}$ outputs m.

Similarly to the Fujisaki-Okamoto transformation, $\mathsf{BE}_{\mathsf{FO}}$ satisfies plaintext awareness with the help of a random oracle and that the validity check of c by re-encryption executed in $\mathsf{Dec}_{\mathsf{FO}}$. Furthermore, due to this re-encryption check and the correctness of BE, we can see that our BE construction $\mathsf{BE}_{\mathsf{FO}}$ satisfies decryption uniqueness, too. In addition to these properties, this transformation preserves the IND-CPA security of the underlying BE scheme. Moreover, we can show that plaintext awareness and IND-CPA security in the RO model imply IND-CCA security in the RO model as in the case of PKE. See Section 6.3.2 for the details.

Finally, let us mention the efficiency of our BE construction $\mathsf{BE}_{\mathsf{FO}}$. The sizes of a public parameter and a secret key are exactly the same, and a ciphertext size is almost the same, as those of the underlying BE. Regarding its computational complexity, an additional cost for achieving plaintext awareness and decryption uniqueness in $\mathsf{BE}_{\mathsf{FO}}$ is only one execution of the encryption algorithm Enc of the underlying BE in the decryption algorithm $\mathsf{Dec}_{\mathsf{FO}}$. Thus, we can see that our approach for achieving plaintext awareness and decryption uniqueness simultaneously is very efficient.

6.3 Plaintext Awareness for Broadcast Encryption

In this section, we present a construction of BE satisfying plaintext awareness. Our BE construction is analogous to the Fujisaki-Okamoto transformation [FO99], which converts an IND-CPA secure PKE scheme (with smoothness) to a plaintext aware and IND-CCA secure PKE scheme in the RO model. Firstly, in Section 6.3.1, we introduce plaintext awareness in the RO model for BE. Secondly, in Section 6.3.2, we show that a plaintext aware and IND-CPA secure BE scheme satisfies IND-CCA security in the RO model. Finally, in Section 6.3.3, we present the description of our BE scheme and theorems regarding its security in the RO model and its decryption uniqueness.

6.3.1 Definition

In this section, we introduce plaintext awareness in the RO model for BE. Our definition is inspired by the formalization of plaintext awareness for PKE by Bellare, Desai, Pointcheval, and Rogaway [BDPR98].

Intuitively, plaintext awareness captures a property that no one can generate a (valid) ciphertext without knowing the plaintext to which it will decrypt. This is a very strong notion of security that generally implies IND-CCA security of PKE schemes (by combining with IND-CPA security) [BDPR98].

In line with the PKE setting, we define plaintext awareness in the RO model for BE by using a special algorithm called the *plaintext extractor* $\mathcal{X}$ against an adversary $\mathcal{A}$. Here, the task of $\mathcal{A}$ is to output a target user set $\mathsf{S}^* \subseteq [n]$ and a ciphertext c^*, where n is the number of users. We can say that the candidate scheme is plaintext aware if any such adversary $\mathcal{A}$ could additionally output the plaintext encrypted in c^*. This is formalized via the plaintext extractor $\mathcal{X}$, given as input the public parameter, the list of random oracle queries made by $\mathcal{A}$ together with corresponding answers, and the values output by $\mathcal{A}$. That is, $\mathcal{X}$ is in the position to retrace the computations that $\mathcal{A}$ executed for generating c^* under S^*, and we expect $\mathcal{X}$ to use this knowledge to output the plaintext encrypted in c^* under S^*.

Similarly to [BDPR98], although usually unnecessary outside the secret key setting, we provide an encryption oracle $\mathcal{EO}$ that enables creation of encrypted plaintexts under chosen sets, but without allowing $\mathcal{A}$ to see random oracle queries asked within corresponding Enc executions. Note that, by storing the queried sets and given ciphertexts into the list $\mathsf{List}_{\mathsf{EO}}$, we allow $\mathcal{X}$ to access to the information except for the encrypted plaintexts. The availability of this oracle for $\mathcal{A}$ is needed for proving the IND-CCA security of BE by combining with

IND-CPA security. See Section 6.3.2 for the details. Furthermore, this encryption oracle will play an important role for proving concurrent deniability of deniable ring authentication in Section 6.5. The formal definition of plaintext awareness in the RO model for BE is provided as follows.

Definition 21 (Plaintext Awareness). *Let* $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *be a BE scheme in the RO model. Let* $\ell_{\mathsf{RO}} := \ell_{\mathsf{RO}}(\lambda)$ *be a polynomial which denotes the output length of the random oracle. Let* $n := n(\lambda)$ *be a polynomial which denotes the number of users. Let* $\mathcal{X}$ *be a PPT plaintext extractor. Consider the following experiment for an adversary* $\mathcal{A}$*.*

$\mathsf{Exp}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{A}}(\lambda)$:
$\mathsf{List}_{\mathsf{EO}} := \emptyset$, $\mathsf{List}_{\mathsf{RO}} := \emptyset$
$(pp, \mathbf{sk} = (sk_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, n)$
$(\mathsf{S}^*, c^*) \leftarrow \mathcal{A}^{\mathcal{EO},\mathcal{CO},\mathcal{RO}}(pp)$
If $(\mathsf{S}^*, c^*) \in \mathsf{List}_{\mathsf{EO}}$ *then return* 0
For all $i \in \mathsf{S}^*$: $m^*_i \leftarrow \mathsf{Dec}(pp, \mathsf{S}^*, sk_i, c^*)$
$m' \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}^*, c^*)$
If $m^*_i = m'$ *for all* $i \in [n]$ *then return* 0
Return 1

In the experiment, the encryption oracle $\mathcal{EO}$*, the corruption oracle* $\mathcal{CO}$*, and the random oracle* $\mathcal{RO}$ *are defined as follows:*

Encryption oracle. $\mathcal{EO}$ *takes* (S, m) *as input, then computes* $c \leftarrow \mathsf{Enc}(pp, \mathsf{S}, m)$*, returns* c *to* $\mathcal{A}$*, and appends* (S, c) *to* $\mathsf{List}_{\mathsf{EO}}$*.*

Corruption oracle. $\mathcal{CO}$ *takes* i *as input, and then returns* sk_i *to* $\mathcal{A}$*.*

Random oracle. $\mathcal{RO}$ *takes* x *as input, and then checks whether* $(x, y) \in \mathsf{List}_{\mathsf{RO}}$ *holds for some* y*. If this is the case, then* $\mathcal{RO}$ *returns* y *to* $\mathcal{A}$*. Otherwise,* $\mathcal{RO}$ *samples* $y \leftarrow \{0,1\}^{\ell_{\mathsf{RO}}}$*, returns* y *to* $\mathcal{A}$*, and appends* (x, y) *to* $\mathsf{List}_{\mathsf{RO}}$*.*

We say that BE *satisfies plaintext awareness in the RO model if there exists a PPT plaintext extractor* $\mathcal{X}$ *such that for any PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$$

holds.[3]

[3] Note that as in plaintext awareness in the RO model for PKE [BDPR98], plaintext awareness in the RO model for BE is defined using a universal extractor that works for any PPT adversary $\mathcal{A}$.

6.3.2 Plaintext Awareness and IND-CPA Security Imply IND-CCA Security

In this section, we show that plaintext awareness and IND-CPA security in the RO model imply IND-CCA security in the RO model. More specifically, we show the following theorem.

Theorem 6.1. *Let* BE *be a BE scheme in the RO model. If* BE *satisfies plaintext awareness and IND-CPA security in the RO model, then* BE *satisfies IND-CCA security in the RO model.*

Proof of Theorem 6.1. Let $\ell_{\mathsf{RO}} := \ell_{\mathsf{RO}}(\lambda)$ be a polynomial which denotes the output length of the RO used in BE. Let $\mathcal{X}$ be a plaintext extractor for the plaintext awareness in the RO model of BE. Let $\mathcal{A}$ be any PPT adversary that attacks the IND-CCA security in the RO model for BE. Without loss of generality, we assume that $\mathcal{A}$ always makes exactly $Q_{dec} > 0$ decryption queries for some polynomial $Q_{dec} := Q_{dec}(\lambda)$. We proceed the proof via a sequence of games. We introduce the two games: $\mathbf{Game}_0$ and $\mathbf{Game}_1$.

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original experiment of IND-CCA security in the RO model for BE. In $\mathbf{Game}_0$, let $\mathsf{List}_{\mathsf{EO}}$ be a list, which is set as $\mathsf{List}_{\mathsf{EO}} := \emptyset$ before the challenge and as $\mathsf{List}_{\mathsf{EO}} := \{(\mathsf{S}^*, c^*)\}$ after the challenge, where S^* is the challenge set and c^* is the challenge ciphertext. Moreover, let $\mathsf{List}_{\mathsf{RO}}$ be a list for the random oracle $\mathcal{RO}$ including the queries x made by $\mathcal{A}$ and the corresponding values output by $\mathcal{RO}$. (Naturally, $\mathsf{List}_{\mathsf{RO}}$ is updated every time when $\mathcal{A}$ makes a new random oracle query x.)

$\mathbf{Game}_1$: $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except for the following change. When $\mathcal{A}$ makes a decryption query (i, S, c), the decryption oracle computes $m \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c)$ instead of computing $m \leftarrow \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$, where $\mathsf{List}_{\mathsf{EO}}$ and $\mathsf{List}_{\mathsf{RO}}$ are the lists (defined as in $\mathbf{Game}_0$) at the point $\mathcal{A}$ makes the decryption query.

For $i \in \{0, 1\}$, let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in $\mathbf{Game}_i$. By using the triangle inequality, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda) &= 2 \cdot \left| \Pr[\mathbf{Succ}_0] - \frac{1}{2} \right| \\ &\leq 2 \cdot |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| + 2 \cdot \left| \Pr[\mathbf{Succ}_1] - \frac{1}{2} \right|.\end{aligned}$$

It remains to show how $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]|$ and $2 \cdot |\Pr[\mathbf{Succ}_1] - \frac{1}{2}|$ are upper-bounded. In the following, we show that there exists a PPT adversary $\mathcal{B}^{\mathsf{pa}}$ against the plaintext awareness in the RO model of BE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq Q_{dec} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$ holds (Lemma 6.1). Then, we show that there exists a PPT adversary $\mathcal{B}^{\mathsf{cpa}}$ against the IND-CPA security in the RO model of BE such that $2 \cdot |\Pr[\mathbf{Succ}_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cpa}}}(\lambda)$ holds (Lemma 6.2).

Lemma 6.1. *There exists a PPT adversary $\mathcal{B}^{\mathsf{pa}}$ against the plaintext awareness in the RO model of BE such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq Q_{dec} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$.*

Proof of Lemma 6.1. For $\alpha \in \{0,1\}$, let $\mathbf{Bad}_\alpha$ be the event that $\mathcal{A}$ makes at least one decryption query (i, S, c) satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c) \neq \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ in $\mathbf{Game}_\alpha$. Moreover, for $j \in [Q_{dec}]$, let $\mathbf{Bad}^j_\alpha$ be the event that the j-th decryption query (i, S, c) satisfies $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c) \neq \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ in $\mathbf{Game}_\alpha$. $\mathbf{Game}_1$ proceeds identically to $\mathbf{Game}_0$ unless $\mathbf{Bad}_1$ happens. Therefore, we have the inequality

$$|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \Pr[\mathbf{Bad}_0] = \Pr[\mathbf{Bad}_1] \leq \sum_{j=1}^{Q_{dec}} \Pr[\mathbf{Bad}^j_1].$$

Then, we construct an adversary $\mathcal{B}^{\mathsf{pa}}$ that attacks the plaintext awareness of BE so that $\sum_{j=1}^{Q_{dec}} \Pr[\mathbf{Bad}^j_1] = Q_{dec} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving a public parameter pp, $\mathcal{B}^{\mathsf{pa}}$ samples $j^* \leftarrow [Q_{dec}]$, initializes $\mathsf{List}_{\mathsf{EO}} := \emptyset$ and $\mathsf{List}_{\mathsf{RO}} := \emptyset$, and gives pp to $\mathcal{A}$.

2. When $\mathcal{A}$ accesses to the decryption oracle $\mathcal{DO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{B}^{\mathsf{pa}}$ responds as follows:

   Decryption oracle. When $\mathcal{A}$ makes a decryption query (i, S, c), $\mathcal{B}^{\mathsf{pa}}$ responds as follows:

   - If (i, S, c) is the j^*-th decryption query, then $\mathcal{B}^{\mathsf{pa}}$ outputs (S, c) to the experiment and terminates.
   - Otherwise, $\mathcal{B}^{\mathsf{pa}}$ computes $m \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c)$ and gives m to $\mathcal{A}$.

   Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{B}^{\mathsf{pa}}$ also makes a corruption query i to its own oracle. Upon receiving a secret key sk_i, $\mathcal{B}^{\mathsf{pa}}$ gives sk_i to $\mathcal{A}$.

   Random oracle. When $\mathcal{A}$ makes a random oracle query x, $\mathcal{B}^{\mathsf{pa}}$ also makes a random oracle query x to its own oracle. Upon receiving a hash value y, $\mathcal{B}^{\mathsf{pa}}$ gives y to $\mathcal{A}$ and appends (x, y) to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs $(\mathsf{S}^*, m_0^*, m_1^*)$ as its challenge set and plaintexts, $\mathcal{B}^{\mathsf{pa}}$ picks a random bit $b \leftarrow \{0,1\}$ and makes an encryption query (S^*, m_b^*) to its own encryption oracle. Upon receiving a ciphertext c^* from the oracle, $\mathcal{B}^{\mathsf{pa}}$ returns c^* to $\mathcal{A}$ and appends (S^*, c^*) to $\mathsf{List}_{\mathsf{EO}}$.

4. When $\mathcal{A}$ accesses to the decryption oracle $\mathcal{DO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{B}^{\mathsf{pa}}$ responds in the same way as before.

5. When $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and terminates, $\mathcal{B}^{\mathsf{pa}}$ gives up and terminates.[4]

Recall that the success condition of $\mathcal{B}^{\mathsf{pa}}$ is to output (S, c) satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c) \neq \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ for some $i \in \mathsf{S}$ and $(\mathsf{S}, c) \notin \mathsf{List}_{\mathsf{EO}}$. Firstly, due to the condition of the decryption query by $\mathcal{A}$, $(\mathsf{S}, c) \notin \mathsf{List}_{\mathsf{EO}}$ holds. For all $j \in [Q_{dec}]$, let $\mathbf{Bad}_{\mathcal{B}}^{j}$ be the event that $\mathcal{A}$ makes a decryption query (i, S, c) satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c) \neq \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$ as the j-th query in the experiment simulated by $\mathcal{B}^{\mathsf{pa}}$. Therefore, $\mathcal{B}^{\mathsf{pa}}$ can break the plaintext awareness of BE if and only if $\mathbf{Bad}_{\mathcal{B}}^{j^*}$ occurs, namely, $\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda) = \Pr[\mathbf{Bad}_{\mathcal{B}}^{j^*}]$. Moreover, from the above construction of $\mathcal{B}^{\mathsf{pa}}$, it is easy to see that $\mathcal{B}^{\mathsf{pa}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ until it terminates. That is, $\Pr[\mathbf{Bad}_{\mathcal{B}}^{j}] = \Pr[\mathbf{Bad}_{1}^{j}]$ holds for all $j \in [Q_{dec}]$. Finally, the choice of j^* is uniformly at random and independent of $\mathcal{A}$, and thus does not affect the behavior of $\mathcal{A}$. Hence, we have

$$\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda) = \Pr[\mathbf{Bad}_{\mathcal{B}}^{j^*}] = \sum_{j=1}^{Q_{dec}} \Pr[\mathbf{Bad}_{\mathcal{B}}^{j} \wedge j = j^*] = \frac{1}{Q_{dec}} \cdot \sum_{j=1}^{Q_{dec}} \Pr[\mathbf{Bad}_{1}^{j}],$$

which in turn implies that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \sum_{j=1}^{Q_{dec}} \Pr[\mathbf{Bad}_1^j] = Q_{dec} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$ holds. □ (**Lemma 6.1**)

Lemma 6.2. *There exists a PPT adversary $\mathcal{B}^{\mathsf{cpa}}$ against the IND-CPA security in the RO model of BE such that $2 \cdot |\Pr[\mathbf{Succ}_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cpa}}}(\lambda)$.*

Proof of Lemma 6.2. Using the adversary $\mathcal{A}$, we construct an adversary $\mathcal{B}^{\mathsf{cpa}}$ as follows.

1. Upon receiving a public parameter pp, $\mathcal{B}^{\mathsf{cpa}}$ initializes $\mathsf{List}_{\mathsf{EO}} := \emptyset$ and $\mathsf{List}_{\mathsf{RO}} := \emptyset$, and gives pp to $\mathcal{A}$.

2. When $\mathcal{A}$ accesses to the decryption oracle $\mathcal{DO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{B}^{\mathsf{cpa}}$ responds as follows:

[4] Actually, this step is never reached since we are assuming that $\mathcal{A}$ always makes exactly Q_{dec} decryption queries, and thus $\mathcal{B}^{\mathsf{pa}}$ will terminate when $\mathcal{A}$ makes the j^*-th decryption query with $j^* \in [Q_{dec}]$.

Decryption oracle. When $\mathcal{A}$ makes a decryption query (i, S, c), $\mathcal{B}^{\mathsf{cpa}}$ computes $m \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}, c)$ and gives m to $\mathcal{A}$

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{B}^{\mathsf{cpa}}$ also makes a corruption query i to its own oracle. Upon receiving a secret key sk_i, $\mathcal{B}^{\mathsf{cpa}}$ gives sk_i to $\mathcal{A}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query x, $\mathcal{B}^{\mathsf{cpa}}$ also makes a random oracle query x to its own oracle. Upon receiving a hash value y, $\mathcal{B}^{\mathsf{cpa}}$ gives y to $\mathcal{A}$ and appends (x, y) to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs $(\mathsf{S}^*, m_0^*, m_1^*)$ as its challenge set and plaintexts, $\mathcal{B}^{\mathsf{cpa}}$ also outputs $(\mathsf{S}^*, m_0^*, m_1^*)$ to its experiment. Upon receiving the challenge ciphertext c^* from the experiment, $\mathcal{B}^{\mathsf{cpa}}$ gives c^* to $\mathcal{A}$ and appends (S^*, c^*) to $\mathsf{List}_{\mathsf{EO}}$.

4. When $\mathcal{A}$ accesses to the decryption oracle $\mathcal{DO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{B}^{\mathsf{cpa}}$ responds in the same way as before.

5. When $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and terminates, $\mathcal{B}^{\mathsf{cpa}}$ outputs b' to its experiment and terminates.

In the following, let b be the challenge bit for $\mathcal{B}^{\mathsf{cpa}}$. We can see that $\mathcal{B}^{\mathsf{cpa}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ so that the challenge bit for $\mathcal{A}$ is the same as the challenge bit b for $\mathcal{B}^{\mathsf{cpa}}$. Therefore, if $\mathcal{A}$ breaks the IND-CCA security in the RO model of BE, $\mathcal{B}^{\mathsf{cpa}}$ can break the IND-CPA security in the RO model of BE since it just outputs b' which is the output by $\mathcal{A}$. Hence, $2 \cdot |\Pr[\mathbf{Succ}_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cpa}}}(\lambda)$ holds. □ (**Lemma 6.2**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq 2Q_{dec} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cpa}}}(\lambda).$$

Since BE satisfies plaintext awareness and IND-CPA security in the RO model and Q_{dec} is some polynomial in λ, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, BE satisfies IND-CCA security in the RO model. □ (**Theorem 6.1**)

6.3.3 Constructing Plaintext Aware Broadcast Encryption

In this section, we present the description of our BE scheme and theorems regarding its plaintext awareness and IND-CPA security in the RO model, and its decryption uniqueness.

$\mathsf{Setup}_{\mathsf{FO}}(1^\lambda, n)$:	$\mathsf{Enc}_{\mathsf{FO}}(pp, \mathsf{S}, m; r)$	$\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}, sk_i, c)$:
$(pp, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$	$R \leftarrow \mathsf{H}_{\mathcal{RO}}(r\|\|m)$	$x \leftarrow \mathsf{Dec}(pp, \mathsf{S}, sk_i, c)$
Return $(pp, \mathbf{sk})$	$c \leftarrow \mathsf{Enc}(pp, \mathsf{S}, r\|\|m; R)$	If $x = \perp$ then return $\perp$
	Return c	Parse $x := r\|\|m$
		$R \leftarrow \mathsf{H}_{\mathcal{RO}}(r\|\|m)$
		If $\mathsf{Enc}(pp, \mathsf{S}, r\|\|m; R) \neq c$ then return $\perp$
		Return m

Figure 6.3: Our construction of broadcast encryption $\mathsf{BE}_{\mathsf{FO}}$.

Let $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a BE scheme with the plaintext space $\{0,1\}^*$ and the randomness space $\mathcal{R}$ for Enc. Let $\mathsf{H}_{\mathcal{RO}} : \{0,1\}^* \to \mathcal{R}$ be a hash function, which is modeled as a random oracle in the security proofs. Using BE and $\mathsf{H}_{\mathcal{RO}}$, we construct the BE scheme $\mathsf{BE}_{\mathsf{FO}} = (\mathsf{Setup}_{\mathsf{FO}}, \mathsf{Enc}_{\mathsf{FO}}, \mathsf{Dec}_{\mathsf{FO}})$ with the plaintext space $\{0,1\}^*$ and the randomness space $\{0,1\}^\lambda$ as described in Figure 6.3.

The correctness of $\mathsf{BE}_{\mathsf{FO}}$ is straightforward due to the correctness of BE. The IND-CPA security, plaintext awareness, and decryption uniqueness of $\mathsf{BE}_{\mathsf{FO}}$ are guaranteed by the following three theorems.

Theorem 6.2. *If* BE *satisfies IND-CPA security, then* $\mathsf{BE}_{\mathsf{FO}}$ *satisfies IND-CPA security in the RO model.*

Proof of Theorem 6.2. Let $n := n(\lambda)$ be an arbitrary polynomial that denotes the number of users. Let $\mathcal{A}$ be any PPT adversary that attacks the IND-CPA security in the RO model of $\mathsf{BE}_{\mathsf{FO}}$. Let $Q_{\mathcal{RO}}$ be the number of random oracle queries made by $\mathcal{A}$. We proceed the proof via a sequence of games. We introduce the following games: $\mathbf{Game}_0$ and $\mathbf{Game}_1$.

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original experiment of IND-CPA security in the RO model for $\mathsf{BE}_{\mathsf{FO}}$. The detailed description is as follows:

1. The setup phase proceeds as follows:
   (a) Generate $(pp, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$.
   (b) Prepare lists $\mathsf{S}_{\mathsf{Corr}} := \emptyset$ and $\mathsf{List}_{\mathsf{RO}} := \emptyset$.
   (c) The public parameter pp is given to $\mathcal{A}$.
2. $\mathcal{A}$ may start making queries to the corruption oracle $\mathcal{CO}$ and the random oracle $\mathcal{RO}$, which are responded as follows:

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{CO}$ gives sk_i to $\mathcal{A}$ and appends i to $\mathsf{S}_{\mathsf{Corr}}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query $r\|m$, $\mathcal{RO}$ checks whether $(r\|m, R) \in \mathsf{List}_{\mathsf{RO}}$ holds for some R. If this is the case, then $\mathcal{RO}$ gives R to $\mathcal{A}$. Otherwise, $\mathcal{RO}$ samples $R \leftarrow \mathcal{R}$, gives R to $\mathcal{A}$, and appends $(r\|m, R)$ to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs a challenge tuple $(\mathsf{S}^*, m_0^*, m_1^*)$ such that $|m_0^*| = |m_1^*|$ and $\mathsf{S}^* \subseteq [n]\backslash\mathsf{S}_{\mathsf{Corr}}$, the experiment proceeds as follows:
   (a) Sample the challenge bit $b \leftarrow \{0,1\}$.
   (b) Sample a randomness $r^* \leftarrow \{0,1\}^\lambda$.
   (c) Compute $R^* \leftarrow \mathsf{H}_{\mathcal{RO}}(r^*\|m_b^*)$.
   (d) Compute $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{S}^*, r^*\|m_b^*; R^*)$.
   (e) The challenge ciphertext c^* is given to $\mathcal{A}$.
4. When $\mathcal{A}$ accesses to $\mathcal{CO}$ and $\mathcal{RO}$, the oracles respond in the same way as before.
5. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

$\mathbf{Game}_1$: $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except for the following two changes.

1. The experiment samples the randomness r^* (that is used to generate the challenge ciphertext) in the setup phase.
2. When $\mathcal{A}$ makes a random oracle query $r\|m$ such that $r = r^*$ before the challenge, $\mathcal{RO}$ gives $\perp$ to $\mathcal{A}$.

Here, for $i \in \{0,1\}$, let $\mathbf{Succ}_i$ be the event that $\mathcal{A}$ succeeds in guessing its challenge bit b in $\mathbf{Game}_i$. We have $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \frac{Q_{\mathcal{RO}}}{2^\lambda}$ because of the following reasons.

- The change 1 is conceptual.
- Regarding the change 2, $\mathbf{Game}_0$ is identical to $\mathbf{Game}_1$ until $\mathcal{A}$ makes a random oracle query including r^* before the challenge. Moreover, r^* is chosen uniformly at random and information-theoretically hidden from $\mathcal{A}$ before the challenge. Since $\mathcal{A}$ makes random oracle queries at most $Q_{\mathcal{RO}}$ times, the probability that $\mathcal{A}$ makes such a query is bounded by $\frac{Q_{\mathcal{RO}}}{2^\lambda}$.

For $i \in \{0,1\}$, let $\mathbf{Bad}_i$ be the event that $\mathcal{A}$ makes a random oracle query containing r^* after the challenge in $\mathbf{Game}_i$. We call such a random oracle query a *bad* query. By using the triangle inequality, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{A}}(\lambda) &= 2 \cdot \left|\Pr[\mathbf{Succ}_0] - \frac{1}{2}\right| \\ &\leq 2 \cdot |\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| + 2 \cdot \left|\Pr[\mathbf{Succ}_1] - \frac{1}{2}\right|\end{aligned}$$

and

$$\begin{aligned}2 \cdot \left|\Pr[\mathbf{Succ}_1] - \frac{1}{2}\right| &= 2 \cdot \left|\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \Pr[\mathbf{Succ}_1|\mathbf{Bad}_1] \cdot \Pr[\mathbf{Bad}_1] - \frac{1}{2}\right| \\ &= 2 \cdot \left|\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2} \cdot \Pr[\mathbf{Bad}_1] - \frac{1}{2} \cdot \Pr[\mathbf{Bad}_1]\right. \\ &\quad \left. + \Pr[\mathbf{Succ}_1|\mathbf{Bad}_1] \cdot \Pr[\mathbf{Bad}_1] - \frac{1}{2}\right| \\ &\leq 2 \cdot \left|\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2} \cdot \Pr[\mathbf{Bad}_1] - \frac{1}{2}\right| \\ &\quad + 2 \cdot \left|\Pr[\mathbf{Succ}_1|\mathbf{Bad}_1] - \frac{1}{2}\right| \cdot \Pr[\mathbf{Bad}_1] \\ &\leq 2 \cdot \left|\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2} \cdot \Pr[\mathbf{Bad}_1] - \frac{1}{2}\right| + \Pr[\mathbf{Bad}_1].\end{aligned}$$

It remains to show how $2 \cdot |\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2}\Pr[\mathbf{Bad}_1] - \frac{1}{2}|$ and $\Pr[\mathbf{Bad}_1]$ are upper-bounded. In the following, we show that there exists a PPT adversary $\mathcal{B}_1$ against the IND-CPA security of BE such that $2 \cdot |\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2}\Pr[\mathbf{Bad}_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_1}(\lambda)$ (Lemma 6.3). Then, we show that there exists a PPT adversary $\mathcal{B}_2$ against the IND-CPA security of BE such that $\Pr[\mathbf{Bad}_1] \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_2}(\lambda) + \frac{Q_{\mathcal{RO}}}{2^\lambda}$ (Lemma 6.4).

Lemma 6.3. *There exists a PPT adversary $\mathcal{B}_1$ against the IND-CPA security of BE such that $2 \cdot |\Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2}\Pr[\mathbf{Bad}_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_1}(\lambda)$.*

Proof of Lemma 6.3. Using the adversary $\mathcal{A}$, we construct an adversary $\mathcal{B}_1$ as follows.

1. Upon receiving a public parameter pp, $\mathcal{B}_1$ samples a randomness $r^* \leftarrow \{0,1\}^\lambda$ and gives pp to $\mathcal{A}$.

2. When $\mathcal{A}$ accesses to the corruption oracle and the random oracle, $\mathcal{B}_1$ responds as follows:

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{B}_1$ also makes a corruption query i to its own oracle. Upon receiving the corresponding secret key sk_i, $\mathcal{B}_1$ gives sk_i to $\mathcal{A}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query $r\|m$, $\mathcal{B}_1$ responds as follows:

(a) If $r = r^*$ holds, then $\mathcal{B}_1$ gives $\perp$ to $\mathcal{A}$.

(b) If $(r\|m, R) \in \mathsf{List}_{\mathsf{RO}}$ holds for some R, then $\mathcal{B}_1$ gives R to $\mathcal{A}$.

(c) $\mathcal{B}_1$ samples $R \leftarrow \mathcal{R}$, gives R to $\mathcal{A}$, and appends $(r\|m, R)$ to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs a challenge tuple $(\mathsf{S}^*, m_0^*, m_1^*)$, $\mathcal{B}_1$ outputs $(\mathsf{S}^*, r^*\|m_0^*, r^*\|m_1^*)$ as the challenge tuple. Upon receiving the challenge ciphertext c^*, $\mathcal{B}_1$ gives c^* to $\mathcal{A}$.

4. When $\mathcal{A}$ accesses to the corruption oracle, $\mathcal{B}_1$ responds in the same way as before. When $\mathcal{A}$ accesses to the random oracle, $\mathcal{B}_1$ responds as follows:

   (a) If $r = r^*$ holds, then $\mathcal{B}_1$ picks a random bit $b' \leftarrow \{0,1\}$, outputs b' to the experiment, and terminates.

   (b) Otherwise, $\mathcal{B}_1$ responds in the same way as before.

5. When $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$ and terminates, $\mathcal{B}_1$ sets $b' := \beta'$, outputs b' to the experiment, and terminates.

In the following, let b be the challenge bit of $\mathcal{B}_1$. Moreover, let $\mathbf{Succ}_{\mathcal{B}}$ be the event that $b = b'$ occurs and $\mathbf{Bad}_{\mathcal{B}}$ the event that the adversary $\mathcal{A}$ (simulated by $\mathcal{B}_1$) makes a random oracle query including r^* after the challenge. Note that until $\mathbf{Bad}_{\mathcal{B}}$ occurs, $\mathcal{B}$ simulates $\mathbf{Game}_1$ perfectly for $\mathcal{A}$ as if $\mathcal{A}$'s challenge bit is that of $\mathcal{B}$ and the randomness R^* used to generate c^* is $\mathsf{H}_{\mathcal{RO}}(r^*\|m_b^*)$. Therefore, we have $\Pr[\mathbf{Succ}_{\mathcal{B}} \wedge \overline{\mathbf{Bad}_{\mathcal{B}}}] = \Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}]$ and $\Pr[\mathbf{Bad}_{\mathcal{B}}] = \Pr[\mathbf{Bad}_1]$. Moreover, if the event $\mathbf{Bad}_{\mathcal{B}}$ occurs, then $\mathcal{B}_1$ outputs a random bit. Hence, $\Pr[\mathbf{Succ}_{\mathcal{B}}|\mathbf{Bad}_{\mathcal{B}}] = \frac{1}{2}$ holds. Therefore, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_1}(\lambda) &= 2 \cdot \left| \Pr[\mathbf{Succ}_{\mathcal{B}}] - \frac{1}{2} \right| \\
&= 2 \cdot \left| \Pr[\mathbf{Succ}_{\mathcal{B}} \wedge \overline{\mathbf{Bad}_{\mathcal{B}}}] + \Pr[\mathbf{Succ}_{\mathcal{B}}|\mathbf{Bad}_{\mathcal{B}}] \cdot \Pr[\mathbf{Bad}_{\mathcal{B}}] - \frac{1}{2} \right| \\
&= 2 \cdot \left| \Pr[\mathbf{Succ}_1 \wedge \overline{\mathbf{Bad}_1}] + \frac{1}{2} \Pr[\mathbf{Bad}_1] - \frac{1}{2} \right| .
\end{aligned}$$

□ (**Lemma 6.3**)

Lemma 6.4. *There exists a PPT adversary $\mathcal{B}_2$ against the IND-CPA security of* BE *such that* $\Pr[\mathbf{Bad}_1] \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_2}(\lambda) + \frac{Q_{\mathcal{RO}}}{2^\lambda}$.

Proof of Lemma 6.4. Using the adversary $\mathcal{A}$, we construct an adversary $\mathcal{B}_2$ as follows.

1. Upon receiving a public parameter pp, $\mathcal{B}_2$ samples a randomness $r^* \leftarrow \{0,1\}^\lambda$ and gives pp to $\mathcal{A}$.
2. When $\mathcal{A}$ accesses to the corruption oracle and the random oracle, $\mathcal{B}_2$ responds in the same way as $\mathcal{B}_1$ in the proof of Lemma 6.3 does.
3. When $\mathcal{A}$ outputs a challenge tuple $(\mathsf{S}^*, m_0^*, m_1^*)$, $\mathcal{B}_2$ samples a bit $\beta \leftarrow \{0,1\}$, sets $M_0^* := r^* \| m_\beta^*$ and $M_1^* := 0^{|r^* \| m_\beta^*|}$, and outputs $(\mathsf{S}^*, M_0^*, M_1^*)$ to its experiment. Upon receiving the challenge ciphertext c^*, $\mathcal{B}_2$ gives c^* to $\mathcal{A}$.
4. When $\mathcal{A}$ accesses to the corruption oracle, $\mathcal{B}_2$ responds in the same way as before. When $\mathcal{A}$ accesses to the random oracle, $\mathcal{B}_2$ responds as follows:
   (a) If $r = r^*$ holds, then $\mathcal{B}_2$ outputs $b' := 0$ to the experiment and terminates.
   (b) Otherwise, $\mathcal{B}_2$ proceeds in the same way as before.
5. When $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$ and terminates, $\mathcal{B}_2$ outputs $b' := 1$ to the experiment, and terminates.

In the following, let b be the challenge bit of $\mathcal{B}_2$. Moreover, let $\mathbf{Bad}_\mathcal{B}$ be the event that $\mathcal{A}$ makes a random oracle query $r\|m$ such that $r = r^*$ after the challenge in the experiment simulated by $\mathcal{B}$. We can see that $\mathcal{B}_2$ outputs 0 to the experiment (that is, $b' = 0$ holds) if and only if $\mathbf{Bad}_\mathcal{B}$ occurs. That is, we have $\Pr[b' = 0] = \Pr[\mathbf{Bad}_\mathcal{B}]$, which in turn implies $|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| = |\Pr[\mathbf{Bad}_\mathcal{B}|b = 0] - \Pr[\mathbf{Bad}_\mathcal{B}|b = 1]|$. Moreover, in the case that $b = 0$ holds, $\mathcal{B}_2$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ until $\mathbf{Bad}_\mathcal{B}$ occurs. That is, we have $\Pr[\mathbf{Bad}_\mathcal{B}|b = 0] = \Pr[\mathbf{Bad}_1]$. Furthermore, in the case that $b = 1$ holds, r^* is information-theoretically hidden in the view of $\mathcal{A}$, and thus the probability that $\mathcal{A}$ makes a bad query (after the challenge) is bounded by $\frac{Q_{\mathcal{RO}}}{2^\lambda}$. That is, we have $\Pr[\mathbf{Bad}_\mathcal{B}|b = 1] \leq \frac{Q_{\mathcal{RO}}}{2^\lambda}$.

From the above arguments, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_2}(\lambda) &= |\Pr[b'=0|b=0] - \Pr[b'=0|b=1]| \\ &= |\Pr[\mathbf{Bad}_{\mathcal{B}}|b=0] - \Pr[\mathbf{Bad}_{\mathcal{B}}|b=1]| \\ &= |\Pr[\mathbf{Bad}_1] - \Pr[\mathbf{Bad}_{\mathcal{B}}|b=1]| \\ &\geq \Pr[\mathbf{Bad}_1] - \Pr[\mathbf{Bad}_{\mathcal{B}}|b=1] \\ &\geq \Pr[\mathbf{Bad}_1] - \frac{Q_{\mathcal{RO}}}{2^\lambda}.\end{aligned}$$

That is, $\Pr[\mathbf{Bad}_1] \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_2}(\lambda) + \frac{Q_{\mathcal{RO}}}{2^\lambda}$ holds. □ (**Lemma 6.4**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE},\mathcal{B}_2}(\lambda) + \frac{3Q_{\mathcal{RO}}}{2^\lambda}.$$

Since BE satisfies IND-CPA security and $Q_{\mathcal{RO}}$ is some polynomial in λ, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, $\mathsf{BE}_{\mathsf{FO}}$ satisfies IND-CPA security in the RO model. □ (**Theorem 6.2**)

Theorem 6.3. *If* BE *satisfies smoothness, then* $\mathsf{BE}_{\mathsf{FO}}$ *satisfies plaintext awareness in the RO model.*

We note that in the proof of Theorem 6.3, we only need a non-programmable random oracle. We also note that as mentioned in Section 6.3.1, these two theorems imply that $\mathsf{BE}_{\mathsf{FO}}$ satisfies IND-CCA security in the RO model under the same assumptions.

Proof of Theorem 6.3. Firstly, we construct a plaintext extractor $\mathcal{X}$ for plaintext awareness in the RO model as follows.

$\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}^*, c^*)$**:**

1. If $(\mathsf{S}^*, c^*) \in \mathsf{List}_{\mathsf{EO}}$ holds, then return $\perp$.
2. For each $(r\|m, R) \in \mathsf{List}_{\mathsf{RO}}$:
   - Compute $c \leftarrow \mathsf{Enc}(pp, \mathsf{S}^*, r\|m; R)$.
   - If $c^* = c$ then return m.
3. Return $\perp$.

Now, suppose $\mathsf{Exp}^{\mathsf{pa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{X},\mathcal{A}}(\lambda)$ is executed, and $\mathcal{A}$ finally outputs (S^*, c^*). Fix any $i \in \mathsf{S}^*$. Let r^* be the randomness included in the result of $\mathsf{Dec}(pp, \mathsf{S}^*, sk_i, c^*)$. Let R^* be the hash value which is computed in $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*)$. Here, due to the construction of $\mathsf{Dec}_{\mathsf{FO}}$, we can see that $\mathsf{Dec}_{\mathsf{FO}}$ outputs $m^* \neq \perp$ if and only if

$$c^* = \mathsf{Enc}(pp, \mathsf{S}^*, r^* \| m^*; R^*) \tag{6.1}$$

holds.

We should consider the following three cases regarding the consistency between the output by $\mathcal{X}$ and the output by $\mathsf{Dec}_{\mathsf{FO}}$ for the pair (S^*, c^*) output by $\mathcal{A}$.

1. Firstly, we consider the case that $r^* \| m^*$ has been (directly) queried to the random oracle by $\mathcal{A}$ (that is, $(r^* \| m^*, R^*) \in \mathsf{List}_{\mathsf{RO}}$). In this case, since $\mathcal{X}$ has access to $\mathsf{List}_{\mathsf{RO}}$ and checks whether $\mathsf{Dec}_{\mathsf{FO}}$ outputs a plaintext m^* or $\perp$ by the equation (6.1), we can see that $\mathcal{X}$ outputs the same m^* which is output by $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*)$.

2. Secondly, we consider the case that $r^* \| m^*$ has been queried to the random oracle by the encryption oracle. In this case, we can see that $(\mathsf{S}^*, c^*) \in \mathsf{List}_{\mathsf{EO}}$ holds, where $c^* = \mathsf{Enc}_{\mathsf{FO}}(pp, \mathsf{S}^*, m^*; r^*)$. In this case, the experiment $\mathsf{Exp}^{\mathsf{pa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{X},\mathcal{A}}(\lambda)$ outputs 0.

3. Thirdly, we consider the case that $r^* \| m^*$ has not been queried to the random oracle. In this case, R^* is sampled from $\mathcal{R}$ uniformly at random when computing $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*)$. Here, due to the smoothness of BE, we can ensure that $c^* = \mathsf{Enc}(pp, \mathsf{S}^*, r^* \| m^*; R^*)$ holds with a negligible probability. Therefore, by the construction of $\mathsf{Dec}_{\mathsf{FO}}$, $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*) = \perp$ holds with an overwhelming probability. Since $\mathcal{X}$ always outputs $\perp$ when $r^* \| m^*$ has not been queried to the random oracle, we can say that the output by $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*)$ agrees with the output by $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}^*, c^*)$ with an overwhelming probability.

From the above arguments, for a pair of a set and a ciphertext (S^*, c^*) output by $\mathcal{A}$, we can see that either of the following conditions holds in any case.

- $(\mathsf{S}^*, c^*) \in \mathsf{List}_{\mathsf{EO}}$
- The output by $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}^*, sk_i, c^*)$ agrees with the output by $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{S}^*, c^*)$ except for a negligible probability.

Therefore, the experiment $\mathsf{Exp}^{\mathsf{pa}}_{\mathsf{BE}_{\mathsf{FO}},\mathcal{X},\mathcal{A}}(\lambda)$ returns 0 with an overwhelming probability, which in turn implies that $\mathsf{BE}_{\mathsf{FO}}$ satisfies plaintext awareness in the RO model.

□ **(Theorem 6.3)**

Theorem 6.4. $\mathsf{BE}_{\mathsf{FO}}$ *satisfies decryption uniqueness.*

Proof of Theorem 6.4. Let $(pp, \mathbf{sk} = (sk_1, \cdots, sk_n)) \leftarrow \mathsf{Setup}_{\mathsf{FO}}(1^\lambda, n)$. Fix $\mathsf{S} \subseteq [n]$, $i \in \mathsf{S}$, and c. In $\mathsf{Dec}_{\mathsf{FO}}(pp, \mathsf{S}, sk_i, c)$, we check whether c is a valid ciphertext by executing the re-encryption procedure. Thus, $\mathsf{BE}_{\mathsf{FO}}$ satisfies decryption uniqueness based on the correctness of the underlying BE scheme BE. □ (**Theorem 6.4**)

6.4 Our Deniable Ring Authentication Scheme

In this section, we formally describe our deniable ring authentication scheme in the RO model based on a BE scheme in the RO model and a collision-resistant hash function.[5] Let $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a BE scheme in the RO model, and suppose it uses an RO with the output length ℓ_{out}.[6] Let $\mathsf{CRHF} = (\mathsf{HKG}, \mathsf{Hash})$ be a collision-resistant hash function. Using BE and CRHF, we construct the deniable ring authentication scheme in the RO model $\mathsf{DRA} = (\mathsf{DRA.Setup}, \langle \mathsf{DRA.Prove}, \mathsf{DRA.Verify} \rangle)$ as described in Figure 6.4, where DRA uses an RO with the output length ℓ_{out}. The correctness of DRA is straightforward due to the correctness of BE.

6.5 Security Proof

Here, in Theorems 6.5, 6.6, and 6.7, we show that DRA satisfies concurrent soundness, source hiding, and concurrent deniability, respectively.

Theorem 6.5. *If* BE *satisfies IND-CCA security and* CRHF *satisfies collision-resistance, then* DRA *satisfies concurrent soundness.*

In the following proof for Theorem 6.5, since we need not use an RO explicitly, the RO is not considered in the security game for better readability. Note that if we include the RO in the arguments, the proof works without any problem. Moreover, if the underlying BE scheme BE is secure in the standard model (without a random oracle), DRA satisfies concurrent soundness in the standard model.

[5] Although we present the deniable ring authentication scheme in the RO model, we explicitly introduce a collision-resistant hash function for simplifying our arguments.

[6] Looking ahead, since we would like to rely on the plaintext awareness of BE in the RO model for proving concurrent deniability, we consider BE in the RO model here. If we only consider concurrent soundness and source hiding for our construction, an RO is not needed.

$\mathsf{DRA.Setup}(1^\lambda, n)$:	$\langle \mathsf{DRA.Prove}(sk_i), \mathsf{DRA.Verify} \rangle (pp', \mathsf{R}, m)$:
$(pp, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$	$\mathbf{Step}_1(\mathsf{DRA.Prove} \Leftarrow \mathsf{DRA.Verify})$:
$hk \leftarrow \mathsf{HKG}(1^\lambda)$	Parse $pp' := (pp, hk)$
$pp' := (pp, hk)$	$t \leftarrow \{0,1\}^\lambda$
Return $(pp', \mathbf{sk})$	$h \leftarrow \mathsf{Hash}(hk, m \Vert \mathsf{R})$
	$c \leftarrow \mathsf{Enc}(pp, \mathsf{R}, h \Vert t)$
	Send $\mathsf{msg}_1 := c$ to $\mathsf{DRA.Prove}$
	Store t as a state
	$\mathbf{Step}_2(\mathsf{DRA.Prove} \Rightarrow \mathsf{DRA.Verify})$:
	Receive msg_1 and parse $\mathsf{msg}_1 := c$
	$h' \Vert t' \leftarrow \mathsf{Dec}(pp, \mathsf{R}, sk_i, c)$
	If $h' \neq \mathsf{Hash}(hk, m \Vert \mathsf{R})$ then $t' := \perp$
	Send $\mathsf{msg}_2 := t'$ to $\mathsf{DRA.Verify}$
	$\mathbf{Step}_3(\mathsf{DRA.Verify})$:
	Receive msg_2 and parse $\mathsf{msg}_2 := t'$
	If $t = t'$ then $v := 1$ else $v := 0$
	Return v

Figure 6.4: Our construction of deniable ring authentication DRA.

Proof of Theorem 6.5. Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of users. Let $\mathcal{A}$ be any PPT adversary that attacks the concurrent soundness of DRA. We proceed the proof via a sequence of games. We introduce the games: $\mathbf{Game}_i$ for $i \in \{0, 1, 2\}$. For simplicity and without loss of generality, in the following games, $\mathcal{A}$ does not corrupt the users in the challenge ring R^* and also does not make a prover query $(\cdot, \mathsf{R}^*, m^*)$ to the prover oracle $\mathcal{PO}$ (since such queries make $\mathcal{A}$ lose).

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original experiment of the concurrent soundness for DRA. The detailed description is as follows:

1. The experiment proceeds as follows:
   (a) Generate $(pp, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$ and $hk \leftarrow \mathsf{HKG}(1^\lambda)$, and set $pp' := (pp, hk)$.
   (b) Prepare a counter $\mathsf{cnt} := 1$ and a list $\mathsf{List}_{\mathsf{PO}} := \emptyset$.
   (c) Give the public parameter pp' to $\mathcal{A}$.
2. $\mathcal{A}$ may start making queries to the prover oracle $\mathcal{PO}$, the execution oracle $\mathcal{EO}$, and the corruption oracle $\mathcal{CO}$, which are responded as follows:

Prover oracle. When $\mathcal{A}$ makes a prover query of the form (i, R, m), $\mathcal{PO}$ initiates $\mathcal{P}_{\mathsf{cnt}}$ as a stateful instance of $\mathsf{DRA.Prove}(sk_i, pp', \mathsf{R}, m)$, appends $(\mathsf{cnt}, i, \mathsf{R}, m)$ to $\mathsf{List}_{\mathsf{PO}}$, and sets $\mathsf{cnt} := \mathsf{cnt} + 1$.

Execution oracle. When $\mathcal{A}$ makes an execution query of the form (j, msg_1) such that $j \in [\mathsf{cnt}]$, $\mathcal{EO}$ responds as follows:

(a) $\mathcal{EO}$ parses $\mathsf{msg}_1 := c$ and computes $h \| t \leftarrow \mathsf{Dec}(pp, \mathsf{R}, sk_i, c)$.

(b) $\mathcal{EO}$ checks $h \neq \mathsf{Hash}(hk, m \| \mathsf{R})$. If this is the case, $\mathcal{EO}$ gives $\mathsf{msg}_2 := \perp$ to $\mathcal{A}$.

(c) $\mathcal{EO}$ sets $\mathsf{msg}_2 := t$ and gives msg_2 to $\mathcal{A}$.

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{CO}$ returns sk_i to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a challenge tuple (R^*, m^*), the experiment executes $\langle \mathcal{A}^{\mathcal{PO}, \mathcal{EO}, \mathcal{CO}}, \mathsf{DRA.Verify} \rangle (pp', \mathsf{R}^*, m^*)$ as follows:

(a) The experiment samples $t^* \leftarrow \{0,1\}^\lambda$, computes $h^* \leftarrow \mathsf{Hash}(hk, m^* \| \mathsf{R}^*)$ and $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, h^* \| t^*)$, and gives $\mathsf{msg}_1^* := c^*$ to $\mathcal{A}$.

(b) When $\mathcal{A}$ outputs $\mathsf{msg}_2^* := t'$ to the experiment and terminates, the experiment checks whether $t^* = t'$ holds.

$\mathbf{Game}_1$: $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except for the following change. Let $\mathsf{msg}_1^* = c^*$ be the first message in the challenge execution. After the challenge execution, when $\mathcal{A}$ makes an execution query $(\cdot, \mathsf{msg}_1^*)$, $\mathcal{EO}$ gives $\perp$ to $\mathcal{A}$.

$\mathbf{Game}_2$: $\mathbf{Game}_2$ is identical to $\mathbf{Game}_1$ except for the following change. In the challenge execution $\langle \mathcal{A}^{\mathcal{PO}, \mathcal{EO}, \mathcal{CO}}, \mathsf{DRA.Verify} \rangle (pp', \mathsf{R}^*, m^*)$, the experiment computes $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, 0^{|h^* \| t^*|})$ as the first message msg_1^*, instead of computing $h^* \leftarrow \mathsf{Hash}(hk, m^* \| \mathsf{R}^*)$ and $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, h^* \| t^*)$.

For $i \in \{0,1,2\}$, let $\mathbf{Succ}_i$ be the event that $t^* = t'$ holds in $\mathbf{Game}_i$. By using the triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{cs}}_{\mathsf{DRA}, \mathcal{A}}(\lambda) = \Pr[\mathbf{Succ}_0] \leq \sum_{i=0}^{1} \Big| \Pr[\mathbf{Succ}_i] - \Pr[\mathbf{Succ}_{i+1}] \Big| + \Pr[\mathbf{Succ}_2].$$

It remains to show how $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]|$, $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]|$, and $\Pr[\mathbf{Succ}_2]$ are upper-bounded. We show that there exists a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ against the collision-resistance of CRHF such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF}, \mathcal{B}^{\mathsf{cr}}}(\lambda)$ (Lemma 6.5). Then,

we show that there exists a PPT adversary $\mathcal{B}^{\mathsf{cca}}$ against the IND-CCA security of BE such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cca}}}(\lambda)$ (Lemma 6.6). Finally, we show that $\Pr[\mathbf{Succ}_2] \leq \frac{1}{2^\lambda}$ holds (Lemma 6.7).

Lemma 6.5. *There exists a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ against the collision-resistance of CRHF such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$.*

Proof of Lemma 6.5. For $\alpha \in \{0,1\}$, let $\mathbf{Coll}_\alpha$ be the event that $\mathcal{A}$ makes at least one prover query (i, R, m) satisfying $\mathsf{Hash}(hk, m^*\|\mathsf{R}^*) = \mathsf{Hash}(hk, m\|\mathsf{R})$ and $m^*\|\mathsf{R}^* \neq m\|\mathsf{R}$ in $\mathbf{Game}_\alpha$. Here, due to the condition of the prover oracle $\mathcal{PO}$ for $\mathcal{A}$, we have $m^*\|\mathsf{R}^* \neq m\|\mathsf{R}$. Moreover, in the case that $\mathsf{Hash}(hk, m^*\|\mathsf{R}^*) \neq \mathsf{Hash}(hk, m\|\mathsf{R})$ holds, when $\mathcal{A}$ makes an execution query $(\cdot, \mathsf{msg}_1^*)$, $\mathcal{A}$ is always given $\perp$ due to the construction of DRA. Thus, $\mathbf{Game}_0$ proceeds identically to $\mathbf{Game}_1$ unless $\mathbf{Coll}_0$ occurs, which in turn implies $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \Pr[\mathbf{Coll}_0] = \Pr[\mathbf{Coll}_1]$. Then, we can construct a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ that attacks the collision-resistance of CRHF so that $\Pr[\mathbf{Coll}_0] = \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$, using the adversary $\mathcal{A}$. Since the construction of $\mathcal{B}^{\mathsf{cr}}$ is straightforward, we omit the details here. Consequently, we have $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$. □ (**Lemma 6.5**)

Lemma 6.6. *There exists a PPT adversary $\mathcal{B}^{\mathsf{cca}}$ against the IND-CCA security of BE such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| = \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cca}}}(\lambda)$.*

Proof of Lemma 6.6. Using the adversary $\mathcal{A}$, we construct an adversary $\mathcal{B}^{\mathsf{cca}}$ as follows.

1. Upon receiving a public parameter pp from the experiment, $\mathcal{B}^{\mathsf{cca}}$ proceeds as follows:
   (a) $\mathcal{B}^{\mathsf{cca}}$ generates $hk \leftarrow \mathsf{HKG}(1^\lambda)$.
   (b) $\mathcal{B}^{\mathsf{cca}}$ initializes a counter $\mathsf{cnt} := 1$ and lists $\mathsf{List}_{\mathsf{PO}} := \emptyset$ and $\mathsf{S}_{\mathsf{Corr}} := \emptyset$.
   (c) $\mathcal{B}^{\mathsf{cca}}$ sets $pp' := (pp, hk)$ and gives pp' to $\mathcal{A}$.
2. When $\mathcal{A}$ accesses to the prover oracle $\mathcal{PO}$, the execution oracle $\mathcal{EO}$, and the corruption oracle $\mathcal{CO}$, $\mathcal{B}^{\mathsf{cca}}$ responds as follows:

   Prover oracle. When $\mathcal{A}$ makes a prover query of the form (i, R, m), $\mathcal{B}^{\mathsf{cca}}$ initiates a prover $\mathcal{P}_{\mathsf{cnt}}$ without sk_i, appends $(\mathsf{cnt}, i, \mathsf{R}, m)$ to $\mathsf{List}_{\mathsf{PO}}$, and sets $\mathsf{cnt} := \mathsf{cnt} + 1$.

   Execution oracle. When $\mathcal{A}$ makes an execution query of the form (j, msg_1) such that $j \in [\mathsf{cnt}]$, $\mathcal{B}^{\mathsf{cca}}$ responds as follows:

(a) $\mathcal{B}^{\mathsf{cca}}$ parses $\mathsf{msg}_1 := c$ and makes a decryption query (i, R, c) to its own decryption oracle.

(b) Upon receiving the decryption result $h\|t$, $\mathcal{B}^{\mathsf{cca}}$ checks $h \neq \mathsf{Hash}(hk, m\|\mathsf{R})$. If this is the case, then $\mathcal{B}^{\mathsf{cca}}$ sends $\mathsf{msg}_2 := \perp$ to $\mathcal{A}$.

(c) $\mathcal{B}^{\mathsf{cca}}$ sets $\mathsf{msg}_2 := t$ and gives msg_2 to $\mathcal{A}$.

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{B}^{\mathsf{cca}}$ also makes a corruption query i to its own oracle. Upon receiving sk_i, $\mathcal{B}^{\mathsf{cca}}$ gives sk_i to $\mathcal{A}$ and appends i to $\mathsf{S}_{\mathsf{Corr}}$.

3. When $\mathcal{A}$ outputs a challenge tuple (R^*, m^*), $\mathcal{B}^{\mathsf{cca}}$ executes $\langle \mathcal{A}^{\mathcal{PO},\mathcal{EO},\mathcal{CO}}, \mathsf{DRA.Verify} \rangle (pp', \mathsf{R}^*, m^*)$ as follows:

(a) $\mathcal{B}^{\mathsf{cca}}$ samples $t^* \leftarrow \{0,1\}^\lambda$.

(b) $\mathcal{B}^{\mathsf{cca}}$ computes $h^* \leftarrow \mathsf{Hash}(hk, m^*\|\mathsf{R}^*)$.

(c) $\mathcal{B}^{\mathsf{cca}}$ sets $M_0^* := h^*\|t^*$ and $M_1^* := 0^{|h^*\|t^*|}$, and outputs $(\mathsf{R}^*, M_0^*, M_1^*)$ as its challenge to the experiment.

(d) Upon receiving the challenge ciphertext c^*, $\mathcal{B}^{\mathsf{cca}}$ sets $\mathsf{msg}_1^* := c^*$ and gives msg_1^* to $\mathcal{A}$.

(e) When $\mathcal{A}$ makes prover queries, execution queries, and corruption queries, $\mathcal{B}^{\mathsf{cca}}$ responds in the same way as before, except that when $\mathcal{A}$ makes an execution query (j, msg_1) to $\mathcal{EO}$, $\mathcal{B}^{\mathsf{cca}}$ additionally checks whether $\mathsf{msg}_1 = \mathsf{msg}_1^*$ holds. If this is the case, then $\mathcal{B}^{\mathsf{cca}}$ returns $\mathsf{msg}_2 := \perp$ to $\mathcal{A}$.

(f) When $\mathcal{A}$ outputs msg_2^* and terminates, $\mathcal{B}^{\mathsf{cca}}$ proceeds as follows:

i. $\mathcal{B}^{\mathsf{cca}}$ parses $\mathsf{msg}_2^* := t'$ and checks whether $t^* = t'$ holds.

ii. If this is (resp., is not) the case, then $\mathcal{B}^{\mathsf{cca}}$ outputs 1 (resp., 0) to the experiment and terminates.

In the following, let b be the challenge bit for $\mathcal{B}^{\mathsf{cca}}$. Note that $\mathcal{B}^{\mathsf{cca}}$ does not make a decryption query $(\cdot, \mathsf{R}^*, c^*)$ to its own oracle since it can answer $\perp$ to $\mathcal{A}$ and we assume that $\mathcal{A}$ does not make a prover query $(\cdot, \mathsf{R}^*, m^*)$. We can see that $\mathcal{B}^{\mathsf{cca}}$ perfectly simulates $\mathbf{Game}_1$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, h^*\|t^*)$ from the experiment. This ensures that the probability that $\mathcal{B}^{\mathsf{cca}}$ outputs 1 given the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, h^*\|t^*)$ is exactly the same as the probability that $\mathbf{Succ}_1$ happens in $\mathbf{Game}_1$. That

is, $\Pr[b' = 1|b = 0] = \Pr[\mathbf{Succ}_1]$ holds. On the other hand, $\mathcal{B}^{\mathsf{cca}}$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ if it receives the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, 0^{|h^*\|t^*|})$ from the experiment. This ensures that the probability that $\mathcal{B}^{\mathsf{cca}}$ outputs 1 given the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, 0^{|h^*\|t^*|})$ is exactly the same as the probability that $\mathbf{Succ}_2$ happens in $\mathbf{Game}_2$. That is, $\Pr[b' = 1|b = 1] = \Pr[\mathbf{Succ}_2]$ holds. Therefore, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cca}}}(\lambda) &= |\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \\ &= |\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]|.\end{aligned}$$

□ (**Lemma 6.6**)

Lemma 6.7. $\Pr[\mathbf{Succ}_2] \leq \frac{1}{2^\lambda}$ *holds.*

Proof of Lemma 6.7. In $\mathbf{Game}_2$, since the experiment computes $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, 0^{|h^*\|t^*|})$ as the first message msg_1^* instead of computing $h^* \leftarrow \mathsf{Hash}(hk, m^*\|\mathsf{R}^*)$ and $c^* \leftarrow \mathsf{Enc}(pp, \mathsf{R}^*, h^*\|t^*)$, t^* is information-theoretically hidden in the view of $\mathcal{A}$. Hence, we can see that the probability that $\mathcal{A}$ outputs the second message $\mathsf{msg}_2^* := t'$ in the challenge execution satisfying $t' = t^*$ is at most $\frac{1}{2^\lambda}$. That is, we have $\Pr[\mathbf{Succ}_2] \leq \frac{1}{2^\lambda}$. □ (**Lemma 6.7**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{cs}}_{\mathsf{DRA},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{BE},\mathcal{B}^{\mathsf{cca}}}(\lambda) + \frac{1}{2^\lambda}.$$

Since BE satisfies IND-CCA security and CRHF satisfies collision-resistance, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{cs}}_{\mathsf{DRA},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Thus, DRA satisfies concurrent soundness.

□ **(Theorem 6.5)**

Theorem 6.6. *If* BE *satisfies decryption uniqueness, then* DRA *satisfies source hiding.*

We note that if we assume that the underlying BE scheme satisfies decryption uniqueness in the RO model, then DRA satisfies source hiding in the RO model.

Proof of Theorem 6.6. Let $\mathcal{A}$ be any adversary against the source hiding for DRA. Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of users and $(pp' = (pp, hk), \mathbf{sk})$ a pair of an (honestly generated) public parameter and secret keys. In the security experiment of the source hiding for DRA, notice that the difference for $\mathcal{A}$ between an interaction with a prover using sk_{i_0} and that using sk_{i_1} will only possibly occur at $\mathbf{Step}_2$ in $\langle \mathsf{DRA.Prove}(sk_{i_b}), \mathcal{A}\rangle(pp', \mathsf{R}^*, m^*)$, where b is the challenge bit for $\mathcal{A}$, and R^* and m^* are the challenge ring and message chosen by $\mathcal{A}$. Concretely, in $\mathbf{Step}_2$, we decrypt the ciphertext

c^* (which is given by $\mathcal{A}$ as the first message) by using the secret key sk_{i_b}. In this situation, thanks to the decryption uniqueness of BE, we can ensure that the result of the decryption is always the same, both in the case of sk_{i_0} and sk_{i_1}, as long as $i_0, i_1 \in \mathsf{R}^* \subseteq [n]$ holds (even if c^* is maliciously generated). Therefore, DRA satisfies source hiding. □ **(Theorem 6.6)**

Theorem 6.7. *If* BE *satisfies plaintext awareness in the RO model and* CRHF *satisfies collision-resistance, then* DRA *satisfies concurrent deniability in the RO model.*

We note that in the following proof of Theorem 6.7, we need only a non-programmable random oracle.

Proof of Theorem 6.7. Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of users. Let $\mathcal{A}$ be any PPT adversary that attacks the concurrent deniability in the RO model for DRA. Let $\mathcal{H}_{\mathcal{RO}}$ be the set of all functions with the output length ℓ_{out}. Let $Q_{\mathcal{TO}}$ and $Q_{\mathcal{EO}}$ be the number of transcript queries made by $\mathcal{A}$ and the number of execution queries made by $\mathcal{A}$, respectively.

First, we construct the following PPT simulator $\mathcal{S}$ that runs in $\mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$.

1. Upon receiving a public parameter $pp' := (pp, hk)$ from the experiment, $\mathcal{S}$ runs $\mathcal{A}(pp)$ and prepares a counter $\mathsf{cnt} := 1$ and lists $\mathsf{List}_{\mathsf{TO}} := \emptyset$ and $\mathsf{List}_{\mathsf{RO}} := \emptyset$.

2. When $\mathcal{A}$ accesses to the prover oracle $\mathcal{PO}$, the execution oracle $\mathcal{EO}$, the transcript oracle $\mathcal{TO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{S}$ responds as follows:

   Prover oracle. When $\mathcal{A}$ makes a prover query of the form (i^*, R^*, m^*), $\mathcal{S}$ initiates a prover $\mathcal{P}_{\mathsf{cnt}}$ without sk_{i^*}, appends $(\mathsf{cnt}, i^*, \mathsf{R}^*, m^*)$ to $\mathsf{List}_{\mathsf{PO}}$, and sets $\mathsf{cnt} := \mathsf{cnt} + 1$.

   Execution oracle. When $\mathcal{A}$ makes an execution query of the form (j, msg_1^*) such that $j \in [\mathsf{cnt}]$, $\mathcal{S}$ checks whether $((i, \mathsf{R}, m), (\mathsf{msg}_1^*, t)) \in \mathsf{List}_{\mathsf{TO}}$ holds for some i, R, m, and t.

   - If this is the case, then $\mathcal{S}$ checks whether $(m^*, \mathsf{R}^*) = (m, \mathsf{R})$ holds.
     - If this is the case, then $\mathcal{S}$ gives $\mathsf{msg}_2^* := t$ to $\mathcal{A}$.
     - Otherwise, $\mathcal{S}$ gives $\mathsf{msg}_2^* := \perp$ to $\mathcal{A}$.
   - Otherwise, $\mathcal{S}$ responds as follows:
     (a) $\mathcal{S}$ retrieves all tuples $\{(i, \mathsf{R}, m), (\mathsf{msg}_1, \mathsf{msg}_2)\}$ from $\mathsf{List}_{\mathsf{TO}}$ and sets $\mathsf{List}_{\mathsf{EO}} := \{(\mathsf{R}, \mathsf{msg}_1)\}$.

(b) $\mathcal{S}$ parses $\mathsf{msg}_1^* := c^*$ and runs $h^* \| t^* \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*)$, where $\mathcal{X}$ is the plaintext extractor due to the plaintext awareness of BE.

(c) $\mathcal{S}$ checks whether $h^* \neq \mathsf{Hash}(hk, m^* \| \mathsf{R}^*)$ holds. If this is the case, then $\mathcal{S}$ gives $\mathsf{msg}_2^* := \perp$ to $\mathcal{A}$.

(d) $\mathcal{S}$ gives $\mathsf{msg}_2^* := t^*$ to $\mathcal{A}$.

Transcript oracle. When $\mathcal{A}$ makes a transcript query of the form (i, R, m), $\mathcal{S}$ also makes a transcript query (i, R, m) to its own transcript oracle. Upon receiving a transcript $\mathsf{tr} = (c, t)$, $\mathcal{S}$ gives (c, t) to $\mathcal{A}$, and appends $((i, \mathsf{R}, m), (c, t))$ to $\mathsf{List}_{\mathsf{TO}}$.

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{S}$ also makes a corruption query i to its own oracle. Upon receiving sk_i, $\mathcal{S}$ gives sk_i to $\mathcal{A}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query x, $\mathcal{S}$ also makes a random oracle query x to its own oracle. Upon receiving a hash value y, $\mathcal{S}$ gives y to $\mathcal{A}$ and appends (x, y) to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs out and terminates, $\mathcal{S}$ also outputs out to the experiment and terminates.

Then, we proceed the proof via a sequence of games. We introduce the following three games: $\{\mathbf{Game}_i\}_{i \in \{0,1,2\}}$.

$\mathbf{Game}_0$: $\mathbf{Game}_0$ is the original real experiment $\mathsf{Exp}^{\mathsf{cd\text{-}real}}_{\mathsf{DRA},\mathcal{A}}(\lambda)$ of the concurrent deniability for DRA. The detailed description is as follows:

1. The setup phase proceeds as follows:
   (a) Generate $(pp, \mathbf{sk} = (sk_1, \cdots, sk_n)) \leftarrow \mathsf{Setup}(1^\lambda)$ and $hk \leftarrow \mathsf{HKG}(1^\lambda)$, and set $pp' := (pp, hk)$.
   (b) Set a counter $\mathsf{cnt} := 1$ and lists $\mathsf{List}_{\mathsf{PO}} := \emptyset$, $\mathsf{List}_{\mathsf{TO}} := \emptyset$, $\mathsf{S}_{\mathsf{Corr}} := \emptyset$, and $\mathsf{List}_{\mathsf{RO}} := \emptyset$.
   (c) Choose a random oracle $\mathcal{RO} \leftarrow \mathcal{H}_{\mathcal{RO}}$.
   (d) The public parameter pp' is given to $\mathcal{A}$.
2. $\mathcal{A}$ may start making queries to the prover oracle $\mathcal{PO}$, the execution oracle $\mathcal{EO}$, the transcript oracle $\mathcal{TO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, which are responded as follows:

Prover oracle. When $\mathcal{A}$ makes a prover query of the form (i^*, R^*, m^*), $\mathcal{PO}$ initiates $\mathcal{P}_{\mathsf{cnt}}$ as a stateful instance of $\mathsf{DRA.Prove}(sk_{i^*}, pp', \mathsf{R}^*, m^*)$, appends $(\mathsf{cnt}, i^*, \mathsf{R}^*, m^*)$ to $\mathsf{List}_{\mathsf{PO}}$, and sets $\mathsf{cnt} := \mathsf{cnt} + 1$.

Execution oracle. When $\mathcal{A}$ makes an execution query of the form (j, msg_1^*) such that $j \in [\mathsf{cnt}]$, $\mathcal{EO}$ responds as follows:

(a) $\mathcal{EO}$ parses $\mathsf{msg}_1^* := c^*$ and computes $h^* \| t^* \leftarrow \mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}^*, c^*)$.

(b) $\mathcal{EO}$ checks $h^* \neq \mathsf{Hash}(hk, m^* \| \mathsf{R}^*)$. If this is the case, $\mathcal{EO}$ gives $\mathsf{msg}_2^* := \bot$ to $\mathcal{A}$.

(c) $\mathcal{EO}$ sets $\mathsf{msg}_2^* := t^*$ and gives msg_2^* to $\mathcal{A}$.

Transcript oracle. When $\mathcal{A}$ makes a transcript query of the form (i, R, m), $\mathcal{TO}$ responds as follows:

(a) $\mathcal{TO}$ samples $t \leftarrow \{0,1\}^\lambda$.

(b) $\mathcal{TO}$ computes $h \leftarrow \mathsf{Hash}(hk, m \| \mathsf{R})$ and $c \leftarrow \mathsf{Enc}(pp, \mathsf{R}, h \| t)$.

(c) $\mathcal{TO}$ gives a transcript (c, t) to $\mathcal{A}$ and appends $((i, \mathsf{R}, m), (c, t))$ to $\mathsf{List}_{\mathsf{TO}}$.[7]

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{CO}$ gives sk_i to $\mathcal{A}$ and appends sk_i to $\mathsf{S}_{\mathsf{Corr}}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query x, $\mathcal{RO}$ responds as follows:

(a) If $(x, y) \in \mathsf{List}_{\mathsf{RO}}$ for some y, then $\mathcal{RO}$ gives y to $\mathcal{A}$.

(b) $\mathcal{RO}$ samples $y \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, gives y to $\mathcal{A}$, and appends (x, y) to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs out and terminates, view is set as $(pp, \mathsf{List}_{\mathsf{TO}}, \mathsf{S}_{\mathsf{Corr}}, \mathsf{List}_{\mathsf{RO}}, \mathsf{out})$ and $(\mathsf{view}, \mathcal{RO})$ is output.

$\mathbf{Game}_1$: $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except for the following change. When $\mathcal{A}$ makes an execution query (j, msg_1^*) such that $j \in [\mathsf{cnt}]$ and $\mathsf{msg}_1^* = \mathsf{msg}_1$, where $((i, \mathsf{R}, m), (\mathsf{msg}_1, \mathsf{msg}_2)) \in \mathsf{List}_{\mathsf{TO}}$ for some i, R, m, and msg_2, the execution oracle $\mathcal{EO}$ responds as follows. If $(m^*, \mathsf{R}^*) \neq (m, \mathsf{R})$ holds, then $\mathcal{EO}$ gives $\bot$ to $\mathcal{A}$. Otherwise, $\mathcal{EO}$ gives msg_2 to $\mathcal{A}$.

$\mathbf{Game}_2$: $\mathbf{Game}_2$ is identical to $\mathbf{Game}_1$ except for the following change. When $\mathcal{A}$ makes an execution query $(j, \mathsf{msg}_1^*(= c^*))$ such that $j \in [\mathsf{cnt}]$ and $((\cdot, \cdot), (\mathsf{msg}_1^*, \cdot)) \notin$

[7] Note that while $\mathcal{TO}$ just gives $\mathsf{msg}_2 := t$ to $\mathcal{A}$ (without decrypting c) here, this does not affect the view of $\mathcal{A}$ due to the correctness of BE.

$\mathsf{List}_{\mathsf{TO}}$, the execution oracle $\mathcal{EO}$ computes $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*)$ instead of computing $\mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}, c^*)$, where $\mathsf{List}_{\mathsf{EO}}$ is a list of all pairs of rings and the first messages stored in $\mathsf{List}_{\mathsf{TO}}$ (that is, $\mathsf{List}_{\mathsf{EO}} = \{(\mathsf{R}, \mathsf{msg}_1)\}$, where $\mathsf{List}_{\mathsf{TO}} = \{((\cdot, \mathsf{R}, \cdot), (\mathsf{msg}_1, \cdot))\}$) at the point that the execution query (j, msg_1^*) is queried.

Here, due to the following reasons, the distribution of the output in $\mathbf{Game}_2$ is exactly the same as one in $\mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$ where $\mathcal{S}$ runs.

From $\mathbf{Game}_0$ to $\mathbf{Game}_2$, we change only how to respond to the execution queries made by $\mathcal{A}$, and thus we have to consider only this change. When $\mathcal{A}$ makes an execution query (j, msg_1^*) in $\mathbf{Game}_2$, we have the following three cases regarding how to respond to it.

- If $((\cdot, \mathsf{R}, m), (\mathsf{msg}_1^*, t)) \in \mathsf{List}_{\mathsf{TO}}$ and $(m^*, \mathsf{R}^*) = (m, \mathsf{R})$ hold for some t, then $\mathcal{EO}$ gives $t' := t$ to $\mathcal{A}$ (due to the change in $\mathbf{Game}_1$).
- If $((\cdot, \mathsf{R}, m), (\mathsf{msg}_1^*, t)) \in \mathsf{List}_{\mathsf{TO}}$ and $(m^*, \mathsf{R}^*) \neq (m, \mathsf{R})$ hold for some t, then $\mathcal{EO}$ gives $t' := \bot$ to $\mathcal{A}$ (due to the change in $\mathbf{Game}_1$).
- If $((\cdot, \cdot, \cdot), (\mathsf{msg}_1^*, \cdot)) \notin \mathsf{List}_{\mathsf{TO}}$ holds, then $\mathcal{EO}$ gives $t' \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*)$ to $\mathcal{A}$ (due to the change in $\mathbf{Game}_2$).

We can see that the above procedures are exactly the same as what $\mathcal{S}$ does in $\mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$.

Let $\mathcal{D}$ be any PPT distinguisher for the above adversary $\mathcal{A}$ and simulator $\mathcal{S}$. Let p_{real} be the probability that $\mathcal{D}$ outputs 1 given $\mathsf{view}_{\mathsf{real}}$, where $(\mathsf{view}_{\mathsf{real}}, \cdot) \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}real}}_{\mathsf{DRA},\mathcal{A}}(\lambda)$. Also, let p_{ideal} be the probability that $\mathcal{D}$ outputs 1 given $\mathsf{view}_{\mathsf{ideal}}$, where $(\mathsf{view}_{\mathsf{ideal}}, \cdot) \leftarrow \mathsf{Exp}^{\mathsf{cd\text{-}ideal}}_{\mathsf{DRA},\mathcal{S}}(\lambda)$. For all $i \in \{0, 1, 2\}$, let $(\mathsf{view}'_i, \cdot)$ be the output in $\mathbf{Game}_i$, and $\mathbf{True}_i$ the event that $\mathcal{D}$ outputs 1 given view'_i in $\mathbf{Game}_i$. Here, by definition, $p_{\mathsf{real}} = \Pr[\mathbf{True}_0]$ and $p_{\mathsf{ideal}} = \Pr[\mathbf{True}_2]$ hold. Therefore, we can estimate $\mathsf{Adv}^{\mathsf{cd}}_{\mathsf{DRA},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda)$ as

$$\mathsf{Adv}^{\mathsf{cd}}_{\mathsf{DRA},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) = |\Pr[\mathbf{True}_0] - \Pr[\mathbf{True}_2]| \leq \sum_{i=0}^{1} |\Pr[\mathbf{True}_i] - \Pr[\mathbf{True}_{i+1}]|.$$

It remains to show how each $|\Pr[\mathbf{True}_i] - \Pr[\mathbf{True}_{i+1}]|$ is upper-bounded. We show that there exists a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ against the collision-resistance of CRHF such that $|\Pr[\mathbf{True}_0] - \Pr[\mathbf{True}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$ (Lemma 6.8). Then, we show that there exists a PPT adversary $\mathcal{B}^{\mathsf{pa}}$ against the plaintext awareness in the RO model of BE such that $|\Pr[\mathbf{True}_1] - \Pr[\mathbf{True}_2]| \leq Q_{\mathcal{EO}} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$ (Lemma 6.9).

Lemma 6.8. *There exists a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ against the collision-resistance of CRHF such that $|\Pr[\mathbf{True}_0] - \Pr[\mathbf{True}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$.*

Proof of Lemma 6.8. For $\alpha \in \{0,1\}$, let $\mathbf{Coll}_\alpha$ be the event that when $\mathcal{A}$ makes an execution query (j, msg_1^*) such that $j \in [\mathsf{cnt}]$ and $\mathsf{msg}_1^* = \mathsf{msg}_1$, where $((\mathsf{R}, m), (\mathsf{msg}_1, \mathsf{msg}_2)) \in \mathsf{List}_{\mathsf{TO}}$ for some R, m, and msg_2, $\mathsf{Hash}(hk, m\|\mathsf{R}) = \mathsf{Hash}(hk, m^*\|\mathsf{R}^*)$ and $m\|\mathsf{R} \neq m^*\|\mathsf{R}^*$ hold in $\mathbf{Game}_\alpha$. Here, in the case that $\mathsf{Hash}(hk, m^*\|\mathsf{R}^*) \neq \mathsf{Hash}(hk, m\|\mathsf{R})$ holds, $\mathcal{A}$ is always given $\perp$ due to the construction of DRA. Moreover, we can see that the change, in the case that $(m^*, \mathsf{R}^*) = (m, \mathsf{R})$ holds in $\mathbf{Game}_1$, does not affect the view of $\mathcal{A}$ since $\mathsf{msg}_2 = \mathsf{msg}_2^*$ holds and BE satisfies correctness. Thus, $\mathbf{Game}_0$ proceeds identically to $\mathbf{Game}_1$ unless $\mathbf{Coll}_0$ happens. Therefore, we have the inequality $|\Pr[\mathbf{True}_0] - \Pr[\mathbf{True}_1]| \leq \Pr[\mathbf{Coll}_0] = \Pr[\mathbf{Coll}_1]$. Then, we can construct a PPT adversary $\mathcal{B}^{\mathsf{cr}}$ that attacks the collision-resistance of CRHF so that $\Pr[\mathbf{Coll}_0] = \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$, using the adversary $\mathcal{A}$. Since the construction of $\mathcal{B}^{\mathsf{cr}}$ is straightforward, we omit the details here. Consequently, we have $|\Pr[\mathbf{True}_0] - \Pr[\mathbf{True}_1]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda)$. □ (**Lemma 6.8**)

Lemma 6.9. *There exists a PPT adversary $\mathcal{B}^{\mathsf{pa}}$ against the plaintext awareness in the RO model of BE such that $|\Pr[\mathbf{True}_1] - \Pr[\mathbf{True}_2]| \leq Q_{\mathcal{EO}} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$.*

Proof of Lemma 6.9. For $\alpha \in \{1,2\}$, let $\mathbf{Bad}_\alpha$ be the event that $\mathcal{A}$ makes at least one execution query $(j, \mathsf{msg}_1^*(= c^*))$ satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*) \neq \mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}, c^*)$ for some $i^* \in \mathsf{R}^*$ in $\mathbf{Game}_\alpha$, where $((\cdot,\cdot),(c^*,\cdot)) \notin \mathsf{List}_{\mathsf{TO}}$. Moreover, for all $\ell \in [Q_{\mathcal{EO}}]$, let $\mathbf{Bad}_\alpha^\ell$ be the event that the ℓ-th execution query $(j, \mathsf{msg}_1^*(= c^*))$ satisfies $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*) \neq \mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}, c^*)$ for some $i^* \in \mathsf{R}^*$ in $\mathbf{Game}_\alpha$, where $((\cdot,\cdot),(c^*,\cdot)) \notin \mathsf{List}_{\mathsf{TO}}$. $\mathbf{Game}_2$ proceeds identically to $\mathbf{Game}_1$ unless $\mathbf{Bad}_2$ happens. Therefore, we have the inequality $|\Pr[\mathbf{True}_1] - \Pr[\mathbf{True}_2]| \leq \Pr[\mathbf{Bad}_2] \leq \sum_{\ell=1}^{Q_{\mathcal{EO}}} \Pr[\mathbf{Bad}_2^\ell]$. Then, we construct a PPT adversary $\mathcal{B}^{\mathsf{pa}}$ that attacks the plaintext awareness in the RO model of BE so that $\sum_{\ell=1}^{Q_{\mathcal{EO}}} \Pr[\mathbf{Bad}_2^\ell] = Q_{\mathcal{EO}} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving a public parameter pp, $\mathcal{B}^{\mathsf{pa}}$ proceeds as follows:
   (a) $\mathcal{B}^{\mathsf{pa}}$ samples $\ell^* \leftarrow [Q_{\mathcal{EO}}]$.
   (b) $\mathcal{B}^{\mathsf{pa}}$ generates $hk \leftarrow \mathsf{HKG}(1^\lambda)$.
   (c) $\mathcal{B}^{\mathsf{pa}}$ initializes a counter $\mathsf{cnt} := 1$ and lists $\mathsf{List}_{\mathsf{TO}} := \emptyset$ and $\mathsf{List}_{\mathsf{RO}} := \emptyset$.
   (d) $\mathcal{B}^{\mathsf{pa}}$ sets $pp' := (pp, hk)$ and gives pp' to $\mathcal{A}$.
2. When $\mathcal{A}$ accesses to the prover oracle $\mathcal{PO}$, the execution oracle $\mathcal{EO}$, the transcript oracle $\mathcal{TO}$, the corruption oracle $\mathcal{CO}$, and the random oracle $\mathcal{RO}$, $\mathcal{B}^{\mathsf{pa}}$ responds as follows:

Prover oracle. When $\mathcal{A}$ makes a prover query of the form (i^*, R^*, m^*), $\mathcal{B}^{\mathsf{pa}}$ initiates a prover $\mathcal{P}_{\mathsf{cnt}}$ without sk_{i^*}, appends $(\mathsf{cnt}, i^*, \mathsf{R}^*, m^*)$ to $\mathsf{List}_{\mathsf{PO}}$, and sets $\mathsf{cnt} := \mathsf{cnt} + 1$.

Execution oracle. When $\mathcal{A}$ makes an execution query of the form (j, msg_1^*) such that $j \in [\mathsf{cnt}]$, $\mathcal{B}^{\mathsf{pa}}$ parses $\mathsf{msg}_1^* := c^*$ and responds as follows:

- If (j, msg_1^*) is the ℓ^*-th execution query, then $\mathcal{B}^{\mathsf{pa}}$ outputs (R^*, c^*) to its experiment and terminates, where $(j, \cdot, m^*, \mathsf{R}^*) \in \mathsf{List}_{\mathsf{PO}}$ for some m^* and R^*.
- Otherwise, $\mathcal{B}^{\mathsf{pa}}$ checks whether $((\mathsf{R}, m), (c^*, t)) \in \mathsf{List}_{\mathsf{TO}}$ holds for some R, m, and t.
  - If this is the case, then $\mathcal{B}^{\mathsf{pa}}$ checks whether $m^* \| \mathsf{R}^* = m \| \mathsf{R}$ holds.
    - If this is the case, then $\mathcal{B}^{\mathsf{pa}}$ gives $\mathsf{msg}_2^* := t$ to $\mathcal{A}$.
    - Otherwise, $\mathcal{B}^{\mathsf{pa}}$ gives $\mathsf{msg}_2^* := \bot$ to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}^{\mathsf{pa}}$ responds as follows:
    (a) $\mathcal{B}^{\mathsf{pa}}$ retrieves all tuples $\{((\mathsf{R}, m), (\mathsf{msg}_1, \mathsf{msg}_2))\}$ from $\mathsf{List}_{\mathsf{TO}}$ and sets $\mathsf{List}_{\mathsf{EO}} := \{(\mathsf{R}, \mathsf{msg}_1)\}$.
    (b) $\mathcal{B}^{\mathsf{pa}}$ runs $h^* \| t^* \leftarrow \mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*)$.
    (c) $\mathcal{B}^{\mathsf{pa}}$ checks whether $h^* \neq \mathsf{Hash}(hk, m^* \| \mathsf{R}^*)$ holds. If this is the case, then $\mathcal{B}^{\mathsf{pa}}$ gives $\mathsf{msg}_2^* := \bot$ to $\mathcal{A}$.
    (d) $\mathcal{B}^{\mathsf{pa}}$ gives $\mathsf{msg}_2^* := t^*$ to $\mathcal{A}$.

Transcript oracle. When $\mathcal{A}$ makes a transcript query of the form (i, R, m), $\mathcal{B}^{\mathsf{pa}}$ responds as follows:

(a) $\mathcal{B}^{\mathsf{pa}}$ samples $t \leftarrow \{0,1\}^\lambda$, computes $h \leftarrow \mathsf{Hash}(hk, m \| \mathsf{R})$, and makes an encryption query $(\mathsf{R}, h \| t)$ to its own oracle.
(b) Upon receiving a ciphertext c, $\mathcal{B}^{\mathsf{pa}}$ sets $\mathsf{msg}_1 := c$.
(c) $\mathcal{B}^{\mathsf{pa}}$ sets $\mathsf{msg}_2 := t$.
(d) $\mathcal{B}^{\mathsf{pa}}$ gives $(\mathsf{msg}_1, \mathsf{msg}_2)$ to $\mathcal{A}$ and appends $((i, \mathsf{R}, m), (\mathsf{msg}_1, \mathsf{msg}_2))$ to $\mathsf{List}_{\mathsf{TO}}$.

Corruption oracle. When $\mathcal{A}$ makes a corruption query i, $\mathcal{B}^{\mathsf{pa}}$ also makes a corruption query i to its own oracle. Upon receiving sk_i, $\mathcal{B}^{\mathsf{pa}}$ gives sk_i to $\mathcal{A}$.

Random oracle. When $\mathcal{A}$ makes a random oracle query x, $\mathcal{B}^{\mathsf{pa}}$ also makes a random oracle query x to its own oracle. Upon receiving y, $\mathcal{B}^{\mathsf{pa}}$ gives y to $\mathcal{A}$ and appends (x, y) to $\mathsf{List}_{\mathsf{RO}}$.

3. When $\mathcal{A}$ outputs out and terminates, $\mathcal{B}^{\mathsf{pa}}$ gives up and terminates.

Recall that the success condition of $\mathcal{B}^{\mathsf{pa}}$ is to output (R^*, c^*) satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*) \neq \mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}, c^*)$ for some $i^* \in \mathsf{R}^*$ and $(\mathsf{R}^*, c^*) \notin \mathsf{List}_{\mathsf{EO}}$. Here, let $\mathbf{Bad}^{\ell}_{\mathcal{B}}$ be the event that $\mathcal{A}$ makes an execution query (j, msg^*_1) satisfying $\mathcal{X}(\mathsf{List}_{\mathsf{EO}}, \mathsf{List}_{\mathsf{RO}}, pp, \mathsf{R}^*, c^*) \neq \mathsf{Dec}(pp, \mathsf{R}^*, sk_{i^*}, c^*)$ for some $i^* \in \mathsf{R}^*$, where $((\cdot,\cdot),(c^*,\cdot)) \notin \mathsf{List}_{\mathsf{TO}}$, as the ℓ-th execution query in the experiment simulated by $\mathcal{B}$. Therefore, $\mathcal{B}^{\mathsf{pa}}$ can break the plaintext awareness of BE if and only if $\mathbf{Bad}^{j^*}_{\mathcal{B}}$ occurs, namely, $\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda) = \Pr[\mathbf{Bad}^{j^*}_{\mathcal{B}}]$. Moreover, from the above construction of $\mathcal{B}^{\mathsf{pa}}$, we can see that $\mathcal{B}^{\mathsf{pa}}$ perfectly simulates $\mathbf{Game}_2$ for $\mathcal{A}$ until it terminates. Thus, $\Pr[\mathbf{Bad}^{\ell}_{\mathcal{B}}] = \Pr[\mathbf{Bad}^{\ell}_2]$ holds for all $\ell \in [Q_{\mathcal{EO}}]$. Furthermore, if we assume that $\mathbf{Bad}^{\ell}_2$ happens, we can ensure that $(\mathsf{R}^*, c^*) \notin \mathsf{List}_{\mathsf{EO}}$ holds since $((\cdot,\cdot),(c^*,\cdot)) \notin \mathsf{List}_{\mathsf{TO}}$ holds now. Finally, the choice of ℓ^* is uniformly at random and independent of $\mathcal{A}$, and thus does not affect the behavior of $\mathcal{A}$. Hence, we have

$$\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda) = \Pr[\mathbf{Bad}^{\ell^*}_{\mathcal{B}}] = \sum_{\ell=1}^{Q_{\mathcal{EO}}} \Pr[\mathbf{Bad}^{\ell}_{\mathcal{B}} \wedge \ell = \ell^*] = \frac{1}{Q_{\mathcal{EO}}} \cdot \sum_{\ell=1}^{Q_{\mathcal{EO}}} \Pr[\mathbf{Bad}^{\ell}_2],$$

which in turn implies that $|\Pr[\mathbf{True}_1] - \Pr[\mathbf{True}_2]| \leq \sum_{\ell=1}^{Q_{\mathcal{EO}}} \Pr[\mathbf{Bad}^{\ell}_2] = Q_{\mathcal{EO}} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda)$ holds. □ (**Lemma 6.9**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{cd}}_{\mathsf{DRA},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{B}^{\mathsf{cr}}}(\lambda) + Q_{\mathcal{EO}} \cdot \mathsf{Adv}^{\mathsf{pa}}_{\mathsf{BE},\mathcal{X},\mathcal{B}^{\mathsf{pa}}}(\lambda).$$

Since BE satisfies plaintext awareness in the RO model, CRHF satisfies collision-resistance, and $Q_{\mathcal{EO}}$ is some polynomial in λ, for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for any PPT distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathsf{cd}}_{\mathsf{DRA},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, DRA satisfies concurrent deniability in the RO model. □ (**Theorem 6.7**)

6.6 An Instantiation of Our Deniable Ring Authentication Scheme

In this section, we give a simple and efficient instantiation of our deniable ring authentication scheme based on an existing BE scheme. Concretely, we present an instantiation based on Gay et al.'s BE scheme [GKW18] under the k-linear assumption. Before describing our instantiation, we introduce some notations for a bilinear group and the k-linear assumption.

Preliminaries for Bilinear Groups. Let $\mathcal{G}$ be a PPT algorithm that, given a security parameter 1^λ as input, outputs an asymmetric bilinear group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, \mathsf{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order $p = \Omega(2^\lambda)$, P_i are generators of $\mathbb{G}_i$ for $i \in \{1, 2\}$, and e is a non-degenerate bilinear map. Here, we require that the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as well as the bilinear map e be computable in deterministic polynomial time, and define a generator in $\mathbb{G}_T$ as $P_T := \mathsf{e}(P_1, P_2)$. We use the implicit representation of group elements as in [EHK+13]. Specifically, for $i \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$, we define $[a]_i := aP_i \in \mathbb{G}_i$ as the implicit representation of a in $\mathbb{G}_i$. Given $[a]_1$ and $[b]_2$, we can efficiently compute $[ab]_T$ using the bilinear map e. Similarly, for a matrix

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{pmatrix} \in \mathbb{Z}_p^{n \times m},$$

we define

$$[\mathbf{A}]_i := \begin{pmatrix} a_{1,1}P_i & \dots & a_{1,m}P_i \\ \vdots & \ddots & \vdots \\ a_{n,1}P_i & \dots & a_{n,m}P_i \end{pmatrix} \in \mathbb{G}_i^{n \times m}$$

as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_i$. For two matrices $\mathbf{A} \in \mathbb{Z}_p^{\ell \times m}$ and $\mathbf{B} \in \mathbb{Z}_p^{m \times n}$, define $\mathsf{e}([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T \in \mathbb{G}_T^{\ell \times m}$.

The k-Linear Assumption. Let $\mathcal{D}_k := \{\mathbf{A}\}$ be a matrix distribution defined as

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & \dots & 0 & 0 \\ 0 & a_2 & \dots & 0 & 0 \\ 0 & 0 & & \ddots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & a_k \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix} \in \mathbb{Z}_p^{(k+1) \times k},$$

where $a_j \leftarrow \mathbb{Z}_p^*$ for all $j \in [k]$.

Definition 22 (The k-Linear Assumption)**.** *We say that the k-linear assumption holds relative to $\mathcal{G}$ in $\mathbb{G}_i$ for $i \in \{1, 2, T\}$ if for any PPT adversary $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(G, [\mathbf{A}]_i, [\mathbf{Aw}]_i) = 1] - \Pr[\mathcal{A}(G, [\mathbf{A}]_i, [\mathbf{u}]_i) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $G := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, \mathsf{e}) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{u} \leftarrow \mathbb{Z}_p^{k+1}$.

We note that the 1-linear assumption corresponds to the SXDH assumption.

An Instantiation of Our Broadcast Encryption Scheme. Before providing an instantiation our deniable ring authentication scheme, we give an instantiation of our plaintext aware and IND-CCA secure BE scheme with Gay et al's IND-CPA secure BE scheme.

Let $\mathsf{H}_{\mathcal{RO}} : \{0,1\}^* \to \mathbb{Z}_p^k$ be a hash function which is modeled as a random oracle. Let $\mathsf{KDF} : \mathbb{G}_T \to \{0,1\}^{\lambda+\ell_m}$ be a key derivation function, where ℓ_m is some polynomial in λ. Then, an instantiation of our BE scheme $\mathsf{BE}_{\mathsf{FO}}^{\mathsf{GKW}} = (\mathsf{Setup}_{\mathsf{FO}}^{\mathsf{GKW}}, \mathsf{Enc}_{\mathsf{FO}}^{\mathsf{GKW}}, \mathsf{Dec}_{\mathsf{FO}}^{\mathsf{GKW}})$ with the plaintext space $\mathcal{M} = \{0,1\}^{\ell_m}$ and the randomness space $\mathcal{R} = \{0,1\}^{\lambda}$ for $\mathsf{Enc}_{\mathsf{FO}}^{\mathsf{GKW}}$ is described in Figure 6.5.

An Instantiation of Our Deniable Ring Authentication Scheme. Now, we present the description of an instantiation of our deniable ring authentication scheme by using the above instantiation of our BE scheme based on Gay et al.'s BE scheme. Let $\mathsf{CRHF} = (\mathsf{HKG}, \mathsf{Hash})$ be a collision-resistant hash function, where Hash has an input space $\{0,1\}^*$ and the output space $\{0,1\}^{\ell_h}$, and ℓ_h is some polynomial in λ. Let $\mathsf{H}_{\mathcal{RO}} : \{0,1\}^* \to \mathbb{Z}_p^k$ be a hash function which is modeled as a random oracle. Let $\mathsf{KDF} : \mathbb{G}_T \to \{0,1\}^{\ell_h+2\lambda}$ be a key derivation function. Then, the description of an instantiation with the message space $\{0,1\}^{\ell_m}$ using our BE scheme (based on Gay et al.'s scheme) is given in Figure 6.6, where ℓ_m is some polynomial in λ.

As shown in Figure 6.6, we can see that this instantiation is efficient due to the underlying Gay et al.'s BE scheme [GKW18] and our transformation. More precisely, a user secret key consists of $k+1$ elements of $\mathbb{G}_2$ and the communication cost consists of $2k+1$ elements of $\mathbb{G}_1$ and a bit string of length $\ell_h + 2\lambda$, where k is the parameter of the underlying k-linear assumption.

$\mathsf{Setup}^{\mathsf{GKW}}_{\mathsf{FO}}(1^\lambda, n)$:

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, \mathsf{e}) \leftarrow \mathcal{G}(1^\lambda)$

$\mathbf{A} \leftarrow \mathcal{D}_k$

$\mathbf{k} \leftarrow \mathbb{Z}_p^{k+1}$

$\mathbf{W}_0, \cdots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{(k+1)\times k}$

$\mathbf{u}_1, \cdots, \mathbf{u}_n \leftarrow \mathbb{Z}_p^k$

$pp := ([\mathbf{A}]_1, [\mathbf{A}^\top \mathbf{W}_0]_1, ([\mathbf{A}^\top \mathbf{W}_i]_1, [\mathbf{u}_i]_1)_{i\in[n]},$
$[\mathbf{A}^\top \mathbf{k}]_T, ([\mathbf{W}_j \mathbf{u}_i]_2)_{i,j\in[n],i\neq j})$

$\forall i \in [n] : sk_i := [\mathbf{k} + \mathbf{W}_0 \mathbf{u}_i]_2 \in \mathbb{G}_2^{k+1}$

$\mathbf{sk} := (sk_i)_{i\in[n]}$

Return $(pp, \mathbf{sk})$

$\mathsf{Enc}^{\mathsf{GKW}}_{\mathsf{FO}}(pp, \mathsf{S}, m; r)$

$\mathbf{s} \leftarrow \mathsf{H}_{\mathcal{RO}}(r \| m)$

$M := r \| m$

$C_0 := [\mathbf{s}^\top \mathbf{A}^\top]_1$

$C_1 := [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{W}_0 + \sum_{j\notin \mathsf{R}} \mathbf{W}_j)]_1$

$R \leftarrow \mathsf{KDF}([\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T)$

$C_2 := R \oplus M$

Return $C := (C_0, C_1, C_2)$

$\mathsf{Dec}^{\mathsf{GKW}}_{\mathsf{FO}}(pp, \mathsf{S}, sk_i, C)$:

Parse $C := (C_0, C_1, C_2)$

$D_1 := \frac{\mathsf{e}(C_0, sk_i)}{\mathsf{e}(C_1, [\mathbf{r}_i]_1)}$

$D_2 := D_1 \cdot \mathsf{e}([\mathbf{s}^\top \mathbf{A}^\top]_1, [\sum_{j\notin \mathsf{R}} \mathbf{W}_j \mathbf{r}_i]_2)$

$M' := C_2 \oplus D_2$

Parse $M' := r' \| m'$

$\mathbf{s}' \leftarrow \mathsf{H}_{\mathcal{RO}}(r' \| m')$

$C_0' := [\mathbf{s}'^\top \mathbf{A}^\top]_1$

$C_1' := [\mathbf{s}'^\top \mathbf{A}^\top (\mathbf{W}_0 + \sum_{j\notin \mathsf{R}} \mathbf{W}_j)]_1$

$R' \leftarrow \mathsf{KDF}([\mathbf{s}'^\top \mathbf{A}^\top \mathbf{k}]_T)$

$C_2' := R' \oplus M'$

If $(C_0, C_1, C_2) \neq (C_0', C_1', C_2')$
then return $\perp$

Return m'

Figure 6.5: An instantiation of plaintext aware and IND-CCA secure broadcast encryption $\mathsf{BE}^{\mathsf{GKW}}_{\mathsf{FO}}$ based on Gay et al.'s scheme [GKW18].

$\mathsf{DRA.Setup}(1^\lambda, n)$:

$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, \mathsf{e}) \leftarrow \mathcal{G}(1^\lambda)$

$\mathbf{A} \leftarrow \mathcal{D}_k$

$\mathbf{k} \leftarrow \mathbb{Z}_p^{k+1}$

$\mathbf{W}_0, \cdots, \mathbf{W}_n \leftarrow \mathbb{Z}_p^{(k+1)\times k}$

$\mathbf{u}_1, \cdots, \mathbf{u}_n \leftarrow \mathbb{Z}_p^k$

$pp := ([\mathbf{A}]_1, [\mathbf{A}^\top \mathbf{W}_0]_1, ([\mathbf{A}^\top \mathbf{W}_i]_1, [\mathbf{u}_i]_1)_{i\in[n]},$
$[\mathbf{A}^\top \mathbf{k}]_T, ([\mathbf{W}_j \mathbf{u}_i]_2)_{i,j\in[n],i\neq j})$

$\forall i \in [n] : sk_i := [\mathbf{k} + \mathbf{W}_0 \mathbf{u}_i]_2 \in \mathbb{G}_2^{k+1}$

$\mathbf{sk} := (sk_i)_{i\in[n]}$

$hk \leftarrow \mathsf{HKG}(1^\lambda)$

$pp' := (pp, hk)$

Return $(pp', \mathbf{sk})$

$\langle \mathsf{DRA.Prove}(sk_i), \mathsf{DRA.Verify}() \rangle (pp', \mathsf{R}, m)$:

$\mathbf{Step}_1(\mathsf{DRA.Prove} \Leftarrow \mathsf{DRA.Verify})$:

$t, r \leftarrow \{0,1\}^\lambda$

$h \leftarrow \mathsf{Hash}(hk, m \| \mathsf{R})$

$\mathbf{s} \leftarrow \mathsf{H}_{\mathcal{RO}}(h \| t \| r)$

$M := h \| t \| r$

$C_0 := [\mathbf{s}^\top \mathbf{A}^\top]_1$

$C_1 := [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{W}_0 + \sum_{j\notin \mathsf{R}} \mathbf{W}_j)]_1$

$R \leftarrow \mathsf{KDF}([\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T)$

$C_2 := R \oplus M$

$C := (C_0, C_1, C_2)$

Send $\mathsf{msg}_1 := C$ to $\mathsf{DRA.Prove}$

Store t as a state

$\mathbf{Step}_2(\mathsf{DRA.Prove} \Rightarrow \mathsf{DRA.Verify})$:

Parse $C := (C_0, C_1, C_2)$

$D_1 := \frac{\mathsf{e}(C_0, sk_i)}{\mathsf{e}(C_1, [\mathbf{r}_i]_1)}$

$D_2 := D_1 \cdot \mathsf{e}([\mathbf{s}^\top \mathbf{A}^\top]_1, [\sum_{j\notin \mathsf{R}} \mathbf{W}_j \mathbf{r}_i]_2)$

$M' := C_2 \oplus D_2$

Parse $M' := h' \| t' \| r'$

$\mathbf{s}' \leftarrow \mathsf{H}_{\mathcal{RO}}(h' \| t' \| r')$

$C_0' := [\mathbf{s}'^\top \mathbf{A}^\top]_1$

$C_1' := [\mathbf{s}'^\top \mathbf{A}^\top (\mathbf{W}_0 + \sum_{j\notin \mathsf{R}} \mathbf{W}_j)]_1$

$R' \leftarrow \mathsf{KDF}([\mathbf{s}'^\top \mathbf{A}^\top \mathbf{k}]_T)$

$C_2' := R' \oplus M'$

If $(C_0, C_1, C_2) \neq (C_0', C_1', C_2')$ then $t' := \perp$

If $h' \neq \mathsf{Hash}(hk, m \| \mathsf{R})$ then $t' := \perp$

Send $\mathsf{msg}_2 := t'$ to $\mathsf{DRA.Verify}$

$\mathbf{Step}_3(\mathsf{DRA.Verify})$:

Receive msg_2 and parse $\mathsf{msg}_2 := t'$

If $t = t'$ then $v := 1$ else $v := 0$

Return v

Figure 6.6: A simple and efficient instantiation of our deniable ring authentication scheme.

Chapter 7

Conclusion and Future Work

This thesis have focused on two major cryptographic authentication primitives over ad-hoc groups, ring signature and (deniable) ring authentication.

In Chapter 4, we propose the first generic construction of ring signature with unconditional anonymity in the plain model based on the standard assumption. Our construction is based on a statistical ZAP argument, a lossy encryption scheme, and a MU-EUF-CMA$^{\mathrm{Corr}}$ secure signature scheme. From the previous works [BFJ+20, BHJ+15, BHY09, GJJM20], all of these building blocks can be instantiated under the quasi-polynomial LWE assumption, and thus our ring signature scheme with unconditional anonymity in the plain model can be instantiated under the quasi-polynomial LWE assumption. As one of the drawback of this ring signature scheme, it has $\mathcal{O}(n)$ signature size, where n is the number of users in a ring. Thus, we leave to explore more efficient ring signature schemes with unconditional anonymity in the plain model based on standard assumptions as an interesting open problem.

In Chapter 5, we propose a new generic construction of tightly secure ring signature in the plain model. The merit of our construction is that its signature size is $\mathcal{O}(\log n)$, which is the same as one of the tightly secure ring signature scheme proposed by Libert et al. [LPQ18] asymptotically. We leave to explore tightly secure ring signature schemes with practical efficiency in the plain model as an interesting open problem.

In Chapter 6, we propose a new generic construction of two-round concurrently deniable ring authentication in the random oracle model. Our generic construction is based on any IND-CPA secure broadcast encryption (BE) scheme. Instantiating the underlying IND-CPA secure BE scheme with the schemes proposed by Agrawal et al. [AY20, AWY20], we obtain the first two-round concurrently deniable ring authentication scheme with optimal efficiency in an asymptotic sense. Here, by optimal efficiency, we mean that all of the sizes of a public

parameter and secret keys, the communication costs, and the number of pairing operations are independent of n, where n is the number of users in a ring. In addition to these main instantiations, through our generic construction, we further obtain various two-round concurrently deniable ring authentication schemes. As an interesting open problem, we leave to explore how to extend our deniable ring authentication scheme into deniable predicate authentication scheme.

Bibliography

[ADK+13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 312–331. Springer, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36362-7_20.

[AHN+17] Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 548–580. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_19.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_26.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In *TCC 2020, Part I*, LNCS, pages 149–178. Springer, Heidelberg, March 2020. doi:10.1007/978-3-030-64375-1_6.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 13–43. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_2.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003. doi:10.1109/SFCS.2003.1238204.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004. doi:10.1145/1030083.1030103.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_3.

[BDH+19] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Ring signatures: Logarithmic-size, no setup - from standard assumptions. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2019, Part III*, LNCS, pages 281–311. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_10.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998. doi:10.1007/BFb0055718.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 642–667. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_22.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. doi:10.1145/62212.62222.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EURO-*

CRYPT 2003, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_26.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005. doi:10.1007/11535218_16.

[BHJ+15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_26.

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. doi:10.1007/s00145-013-9167-4.

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 405–434. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_14.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009. doi:10.1007/978-3-642-01001-9_1.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, Heidelberg, March 2006. doi:10.1007/11681878_4.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_23.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, Heidelberg, December 2004. doi:10.1007/978-3-540-30539-2_4.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

[BW19] Ward Beullens and Hoeteck Wee. Obfuscating simple functionalities from knowledge assumptions. In *PKC 2019, Part II*, LNCS, pages 254–283. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-17259-6_9.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741.

[CGH+21] Rohit Chatterjee, Sanjam Garg, Mohammad Hajiabadi, Dakshita Khurana, Xiao Liang, Giulio Malavolta, Omkant Pandey, and Sina Shiehian. Compact ring signatures from learning with errors. LNCS, pages 282–312. Springer, Heidelberg, 2021. doi:10.1007/978-3-030-84242-0_11.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 423–434. Springer, Heidelberg, July 2007. doi:10.1007/978-3-540-73420-8_38.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_20.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_5.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_4.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991. doi:10.1007/3-540-46416-6_22.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_25.

[CWLY06] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS 06*, pages 297–302. ACM Press, March 2006.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_36.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. doi:10.1145/103418.103474.

[DG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005*, pages 112–121. ACM Press, November 2005. doi:10.1145/1102120.1102137.

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, October / November 2006. doi:10.1145/1180405.1180454.

[DHIN11] Rafael Dowsley, Goichiro Hanaoka, Hideki Imai, and Anderson C. A. Nascimento. Round-optimal deniable ring authentication in the presence of big brother. In Yongwha Chung and Moti Yung, editors, *WISA 10*, volume 6513 of *LNCS*, pages 307–321. Springer, Heidelberg, August 2011.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_36.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892117.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, May 1998. doi:10.1145/276698.276853.

[DRS18] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 419–440. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-79063-3_20.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_8.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48329-2_40.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors,

PKC'99, volume 1560 of *LNCS*, pages 53–68. Springer, Heidelberg, March 1999. doi:10.1007/3-540-49162-7_5.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_8.

[GJJM20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 668–699. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_23.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 253–280. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_9.

[GKW18] Romain Gay, Lucas Kowalczyk, and Hoeteck Wee. Tight adaptively secure broadcast encryption with short ciphertexts and keys. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 123–139. Springer, Heidelberg, September 2018. doi:10.1007/978-3-319-98113-0_7.

[Gon19] Alonso González. Shorter ring signatures from standard assumptions. In *PKC 2019, Part I*, LNCS, pages 99–126. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-17253-4_4.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006. doi:10.1007/11818175_6.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_24.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, Heidelberg, April 2009. doi:10.1007/978-3-642-01001-9_10.

[HJ12] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_35.

[HK08] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 308–325. Springer, Heidelberg, December 2008. doi:10.1007/978-3-540-89255-7_19.

[HLL+15] Jingnan He, Bao Li, Xianhui Lu, Dingding Jia, Haiyang Xue, and Xiaochao Sun. Identity-based lossy encryption from learning with errors. In Keisuke Tanaka and Yuji Suga, editors, *IWSEC 15*, volume 9241 of *LNCS*, pages 3–20. Springer, Heidelberg, August 2015. doi:10.1007/978-3-319-22425-1_1.

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 489–518. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56617-7_17.

[HS03] Javier Herranz and Germán Sáez. Forking lemmas for ring signature schemes. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 266–279. Springer, Heidelberg, December 2003.

[IKE] D. Harkins and D. Carrel, eds. The Internet Key Exchange (IKE). RFC 2409, November 1998.

[JOR18] Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure structure-preserving signatures. In Michel Abdalla and Ricardo Dahab,

editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 123–152. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76581-5_5.

[Kat03] Jonathan Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 211–228. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_13.

[LJYP14] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_1.

[LPJY15] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_28.

[LPQ18] Benoît Libert, Thomas Peters, and Chen Qian. Logarithmic-size ring signatures with tight security from the DDH assumption. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *ESORICS 2018, Part II*, volume 11099 of *LNCS*, pages 288–308. Springer, Heidelberg, September 2018. doi:10.1007/978-3-319-98989-1_15.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer, Heidelberg, February 2011. doi:10.1007/978-3-642-19074-2_24.

[MS17] Giulio Malavolta and Dominique Schröder. Efficient ring signatures in the standard model. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 128–157. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70697-9_5.

[Nao02] Moni Naor. Deniable ring authentication. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 481–498. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_31.

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 121–145. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_6.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_19.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996. doi:10.1007/3-540-68339-9_33.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001. doi:10.1007/3-540-45682-1_32.

[Sch11] Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 189–206. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20465-4_12.

[Signal] Open Whisper Systems. Signal. http://signal.org/, 2016.

[SM04a] Willy Susilo and Yi Mu. Deniable ring authentication revisited. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04*, volume 3089 of *LNCS*, pages 149–163. Springer, Heidelberg, June 2004. doi:10.1007/978-3-540-24852-1_11.

[SM04b] Willy Susilo and Yi Mu. Non-interactive deniable ring authentication. In Jong In Lim and Dong Hoon Lee, editors, *ICISC 03*, volume 2971 of *LNCS*, pages 386–401. Springer, Heidelberg, November 2004.

[SS10] Sven Schäge and Jörg Schwenk. A CDH-based ring signature scheme with short signatures and public keys. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 129–142. Springer, Heidelberg, January 2010.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 166–180. Springer, Heidelberg, April 2007. doi:10.1007/978-3-540-71677-8_12.

[YAS+12] Shota Yamada, Nuttapong Attrapadung, Bagus Santoso, Jacob C. N. Schuldt, Goichiro Hanaoka, and Noboru Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 243–261. Springer, Heidelberg, May 2012. doi:10.1007/978-3-642-30057-8_15.

[ZCTH17] S. Zeng, Y. Chen, S. Tan, and M. He. Concurrently deniable ring authentication and its applications to LBS in VANETs. *Peer-to-Peer Netw. Appl.*, 10(4) : 844–856, 2017.

[ZMYH17] Shengke Zeng, Yi Mu, Guomin Yang, and Mingxing He. Deniable ring authentication based on projective hash functions. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *ProvSec 2017*, volume 10592 of *LNCS*, pages 127–143. Springer, Heidelberg, October 2017.