Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications

by

Kristiyan Haralambiev

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Courant Institute of Mathematical Sciences

New York University

May 2011

Victor Shoup

Abstract

Non-interactive zero-knowledge (NIZK) proofs have enjoyed much interest in cryptography since they were introduced more than twenty years ago by Blum et al. [BFM88]. While quite useful when designing modular cryptographic schemes, until recently NIZK could be realized efficiently only using certain heuristics. However, such heuristic schemes have been widely criticized. In this work we focus on designing schemes which avoid them. In [GS08], Groth and Sahai presented the first efficient (and currently the only) NIZK proof system in the standard model. The construction is based on bilinear maps and is limited to languages of certain satisfiable system of equations. Given this expressibility limitation of the system of equations, we are interested in cryptographic primitives that are "compatible" with it. Equipped with such primitives and the Groth-Sahai proof system, we show how to construct cryptographic schemes efficiently in a modular fashion.

In this work, we describe properties required by any cryptographic scheme to mesh well with Groth-Sahai proofs. Towards this, we introduce the notion of "structure-preserving" cryptographic schemes. We present the first constant-size structure-preserving signature scheme for messages consisting of general bilinear group elements. This allows us (for the first time) to instantiate efficiently a modular construction of round-optimal blind signature based on the framework of Fischlin [Fis06].

Our structure-preserving homomorphic trapdoor commitment schemes yield the

first efficient leakage-resilient signatures (in the bounded leakage model) which satisfy the standard security requirements and additionally tolerate any amount of leakage.

Furthermore, we build a structure-preserving encryption scheme which satisfies the standard CCA security requirements. While resembling the notion of verifiable encryption, it provides better properties and yields the first efficient two-party protocol for joint ciphertext computation. Note that the efficient realization of such a protocol was not previously possible even using the heuristics mentioned above.

Lastly, we revisit the notion of simulation extractability and define "true-simulation extractable" NIZK proofs. Although quite similar to the notion of simulation-sound extractable NIZK proofs, there is a subtle but rather important difference which makes it weaker and easier to instantiate efficiently. As it turns out, in many scenarios this new notion is sufficient.

Contents

List of Tables

Chapter 1

Introduction

The notion of zero-knowledge proofs [63] is a fundamental and extremely powerful cryptographic tool. It allows a party to convey the correctness of a statement without revealing anything but the correctness. These seemingly contradictory property is usually realized using several rounds of communications between a prover and a verifier, where the former tries to convince the latter that a statement is correct without giving any other information. Zero-knowledge proofs are an essential building block for the modular construction of many cryptographic schemes, e.g., multi-party computation protocols in which each party has to provide evidence that they have carried their computation correctly in order to show the protocol secure against malicious parties.

Non-interactive zero-knowledge proofs [19], as their name suggests, were introduced as a means of carrying out zero-knowledge (ZK) proofs while removing the interaction between the prover and the verifier. This is done with the help of a randomly generated common reference string which both the prover, when producing a proof, and the verifier, when checking the validity of a proof, can read. Non-interactive zero-knowledge (NIZK) proofs have many cryptographic applications. A well-known

example is the Naor-Yung paradigm [85] which constructs a IND-CCA encryption scheme (see Definition 4) out of two semantically secure ones together with NIZK proofs (with certain special properties). While quite useful when designing modular cryptographic schemes, NIZK proofs could only be realized efficiently when using certain heuristics until recently. That is, to transform efficient interactive ZK proofs into NIZK proofs using the random oracle paradigm [16] or to directly use interactive assumptions [82]. Due to a series of criticisms against such heuristics starting with [41], we are interested in schemes which avoid them.

Nevertheless, we consider modular design of cryptographic protocols to be the right approach. While some cryptographic tasks find "cleverly crafted" efficient solutions dedicated to their own purposes, modular constructions make cryptographic schemes easier to build and understand, less prone to errors in the proof of correctness, and a good alternative for comparison when the buildings blocks have efficient instantiations. Moreover, such modular constructions can be instantiated under different assumptions as the designer can choose the concrete realization of the underlying cryptographic primitives.

An efficient NIZK proof system, however, has been absent until recently. In [71], Groth and Sahai presented the first efficient one (and current the only one) in the standard model. The construction is based on bilinear maps and limited to statements which can be expressed as a system of equations of certain types. We review the Groth-Sahai (GS) proof system in Section 2.4. As its expressibility is limited to only a few types of equations, the most interesting of which is the pairing-product equation (PPE), we are particularly interested in cryptographic primitives that are "compatible" with it. We call such cryptographic primitives *structure-preserving*.

Before proceeding further, the reader is encouraged to familiarize themselves with the notions and notations in Chapter 2 which we are going to use in this work.

1.1 Structure-Preserving Cryptographic Primitives

Let us start with a short motivation in the case of signatures. The combination of digital signatures and NIZK proofs of knowledge appears frequently in privacy-protecting cryptographic protocols such as blind signatures [55, 4], group signatures [15, 76, 17], anonymous credential systems [13, 12], verifiably encrypted signatures [24, 37, 94], non-interactive group encryption [44] and so on. There are efficient signature schemes, e.g., [20, 35, 13, 33], whose verification predicates are pairing-product equations, and, hence, possibly suitable counterparts to the GS proofs system. However, they cannot be used in NIZK proofs of knowledge as none of them has both signatures *and* messages consisting only of group elements. Since only group elements can be extracted from GS proofs, this entails limited applicability of those schemes or requires a stronger security notions such as *F-unforgeability* [13]. Research on signature schemes that are compatible with GS proofs was initiated in [66]. While the design goal is clear and simple, giving an efficient instantiation has proved hard for years.

The desirable properties of a cryptographic scheme which allow modular design together with GS proofs are the following:

1. the scheme satisfies the standard notion of security: unforgeable against chosen-message attacks (EUF-CMA) in the case of signatures, indistinguishability against chosen-ciphertext attacks (IND-CCA) for encryption, etc.;

2. the public keys, messages, and results of the main algorithm of the scheme (Enc for encryption, Sign for signatures, Com for commitments) are elements of a bilinear group[1];

[1] If witness indistinguishable (WI) rather thank zero-knowledge proofs suffice for our needs, we relax this to allow elements of the target group.

3. the correctness of computation for that algorithm can be verified using a conjunction of pairing product equations over the public key, the message, and the output of the algorithm[2].

Note that this proscribes the use of cryptographic hash functions, which usually play a central role in the construction of EUF-CMA signatures and IND-CCA encryption. We therefore call such a scheme *structure preserving*.

Combined with GS proofs, such structure-preserving scheme allow proving knowledge of a messages, signature/ciphertext/commitment, and public key without actually revealing them.

1.1.1 Signatures

We present the first *constant-size* structure-preserving signature scheme for messages of general bilinear group elements. A signature consists only of 7 group elements regardless of the size of the message. For a message $(\mathbf{m}_1, \ldots, \mathbf{m}_k)$, a signature $(\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ fulfills the verification equations

$$\mathbf{A} = \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r})\, \hat{e}(\mathbf{s}, \mathbf{t}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i), \text{ and}$$

$$\mathbf{B} = \hat{e}(\mathbf{h}_z, \mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u})\, \hat{e}(\mathbf{v}, \mathbf{w}) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i)$$

where the verification key is $(\mathbf{A}, \mathbf{B}, \mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^{k})$, for a fixed constant k, and $\hat{e}$ is a bilinear map. The security is based on a novel strong, "q-type", assumption which is fairly complex. However, it has an *optimal quadratic security bound* in generic bilinear groups unlike the popular strong Diffie-Hellman assumption and its variations. Then, we explore several variations.

[2]Note that if there is a public verification algorithm, e.g., in the case of signatures, essentially this requirement holds for the verification predicate.

We present a scheme that signs *unbounded-size* messages based on the observation that the constant-size signatures allow unbounded "signature chaining". We address the case of signing group elements from both base groups $\mathbb{G}_1$ and $\mathbb{G}_2$ for asymmetric pairings when the SXDH assumption holds; it is not trivial to sign a message consisting of elements from both groups as there are no efficient mappings between the two groups. Also, we consider structure-preserving signatures that provides strong unforgeability. Finally, we define the notion of *simulatable signatures* and give an efficient instantiation. It is defined in the common reference string (CRS) model and allows creating valid signatures using the trapdoor associated with the CRS. Such a property is useful in building adaptively secure protocols where a simulator has to have correct signatures without having help from a corrupted signer [4].

Related Work. Feasibility of structure-preserving signatures on group elements was first shown by Groth [66], who presents a construction based on the decision linear assumption (DLIN). While it is remarkable that the security can be based on a simple standard assumption, the scheme is not practical as signatures consist of hundreds of thousands of group elements. Based on the q-Hidden LRSW assumption for asymmetric bilinear groups, Green and Hohenberger presented a structure-preserving signature scheme that provides security against random message attacks [65]. Unfortunately, an extension to the chosen message security is not known. In [56], Fuchsbauer presented a scheme based on (a variant of) the Double Hidden Strong Diffie-Hellman Assumption (DHSDH) from [57]. Their scheme is pretty efficient but has limited generality since a trusted set-up is necessary and the messages must be in a special form called a Diffie-Hellman pair. In [44], Cathalo, Libert and Yung showed a scheme based on a combination of the Hidden Strong Diffie-Hellman Assumption (HSDH), Flexible Diffie-Hellman Assumption, and the DLIN assumption. Their sig-

nature consists of $9k+4$ group elements and it is left as an open problem to construct constant-size signatures.

1.1.2 Commitments

In this work, we consider (non-interactive) public-key homomorphic trapdoor commitments [72, 92]. Although we do not use the homomorphic property in our applications, most of our commitments are homomorphic. Common to all homomorphic trapdoor commitment schemes prior to the recent work of [68] is that they are homomorphic with respect to addition in a ring or field. However, in public-key cryptography, we often work over groups that are not rings, like in the case of PPEs of Groth-Sahai proofs, so it is useful to be able to commit to group elements.

Firstly, we present a trapdoor commitment scheme which satisfies the slightly relaxed structure-preserving noton. Its message and public key consist of group elements, $\vec{\mathbf{m}}$ and $(\mathbf{g}, \mathbf{h}, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k)$, respectively; but the commitment $\mathfrak{c} = (\mathbf{C}_1, \mathbf{C}_2)$ consist of elements in the target group:

$$\mathbf{C}_1 = \hat{e}(\mathbf{g}, \mathfrak{d}_1) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \quad \wedge \quad \mathbf{C}_2 = \hat{e}(\mathbf{h}, \mathfrak{d}_2) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i).$$

Here, the pair $(\mathfrak{d}_1, \mathfrak{d}_2)$ is the opening of the commitment, often referred to as a decommitment. When working in the SXDH setting, the public key and commitment sizes could be reduced in half so that the verification equation becomes: $\mathfrak{c} = \hat{e}(\mathbf{g}, \mathfrak{d}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i)$. Clearly, these commitment schemes are *length reducing*, a property than only a few trapdoor commitment schemes possess.

Next, we present a commitment scheme that satisfies the full structure-preserving notion by representing the target group elements as a product of random pairing

product equations[3]:

$$\mathbf{C}_1 = \hat{e}(\mathbf{g}, \mathbf{a}_0) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{a}_i) = \hat{e}(\mathbf{g}, \mathfrak{d}_1) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \quad \wedge$$
$$\mathbf{C}_2 = \hat{e}(\mathbf{h}, \mathbf{b}_0) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{b}_i) = \hat{e}(\mathbf{h}, \mathfrak{d}_2) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i).$$

To do so, we use novel techniques for pairing product randomization. Note, however, that the commitment $\mathfrak{c} = (\{\mathbf{a}_i, \mathbf{b}_i\}_{i=0}^{k})$ is no longer length-reducing.

Lastly, we construct a commitment scheme whose message is a scalar but can be verified using a PPE. Its verification equation is $\hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^m) = \hat{e}(\mathfrak{d}, \tilde{\mathbf{f}})$. Although it is not structure-preserving in general, for the special case when the message in given in clear together with the proof, it is fully compatible with GS proofs and provides better efficiency than the last scheme.

The security of all commitment schemes is based on novel simple assumptions which are implied by the well-known SXDH or DLIN.

Related Work. There are many examples of homomorphic commitments. Homomorphic cryptosystems such as [61, 87, 88, 25, 22] can be seen as homomorphic commitment schemes that are perfectly binding and computationally hiding. Commitments based on homomorphic encryption can be converted into computationally binding and perfectly hiding homomorphic commitments, see for instance the mixed commitments of Damgard and Nielsen [50] and the commitment schemes used by Groth et. al. [70], Boyen and Waters [28], Groth [66], and Groth and Sahai [71]. Even in the perfectly hiding versions of these schemes the size of a commitment is larger than the size of a message though. This length increase follows from the fact

[3]These equations are for the case of symmetric pairings; the asymmetric case is handled slightly differently.

that the underlying building block is a cryptosystem whose ciphertexts must be large enough to include the message.

There are also direct constructions of homomorphic trapdoor commitment schemes such as Guillou and Quisquater commitments [72] and Pedersen commitments [91]. The latter is one of the most used commitment schemes in the field of cryptography. They are perfectly hiding with a trapdoor and if the discrete-logarithm problem is hard they are computationally binding. There are many variants of the Pedersen commitment scheme. Fujisaki and Okamoto [58] and Damgard and Fujisaki [49] for instance suggest a variant where the messages can be arbitrary integers. However, none of the trapdoor commitment schemes prior to the recent work of Groth [68] has messages being (vectors of) group elements.

1.1.3 Encryption

As we mentioned earlier, Groth [66] initiated the study of structure-preserving cryptographic primitives by looking into signatures. While it is surprising, the construction he gave is based on DLIN, its instantiation is extremely inefficient. The three building blocks used in that construction are a proof system, a one-way function, and a (slightly relaxed variant of) IND-CCA encryption. While the first two components could be instantiated efficiently, there is no appropriate CCA encryption as the existing efficient schemes do not satisfy all three structure-preserving properties. On the one hand, many IND-CPA schemes would satisfy the second and the third property, but will not provide the desired IND-CCA security. On the other hand, the existing efficient IND-CCA encryption schemes come short of satisfying the second and third property as well. For example, the well-known Cramer-Shoup (CS) encryption [46], for example, which provides IND-CCA security and can be modified to

work in a bilinear group [99], does not satisfy the third property because of the way it computes its validity element, and one cannot prove that a ciphertext is correctly computed without revealing the ciphertext itself. While it is sufficient for many applications when combined with GS proofs, including our leakage-resilient constructions discussed later, it does not satisfy the full structure-preserving notion. In particular, in the construction of [66], a signature consists of a ciphertext and a proof, so it is not possible to give a NIZK proof of knowledge of a signature without revealing significant part of that signature, hence, violating the ZK property.

With this motivation in mind, we consider the problem of structure-preserving IND-CCA encryption. The construction we present has some similarities with the (Linear) CS encryption in that it also produces a ciphertext which consists of a randomness vector $\vec{\mathbf{u}} = (\mathbf{g}_1^r, \mathbf{g}_2^s, \mathbf{g}_3^{r+s})$, a one time-pad of the message $\mathbf{c} = \mathbf{m} \cdot \mathbf{h}_1^r \mathbf{h}_2^s$, and a validity element $\mathbf{V}$. As a side note, observe that if the validity element is omitted, the resulting scheme is essentially the Linear encryption by Boneh et. al. [22]. However, unlike previous work, we compute $\mathbf{V}$ while keeping the underlying algebraic structure, i.e. $\mathbf{V} = \prod_{i=0}^{3} \hat{e}(\mathbf{f}_{i,1}^r \mathbf{f}_{i,2}^s, \mathbf{u}_i) \cdot \hat{e}(\mathbf{f}_{4,1}^r \mathbf{f}_{4,2}^s, \mathbf{c}) \cdot \hat{e}(\mathbf{f}_{5,1}^r \mathbf{f}_{5,2}^s, \boldsymbol{\ell})$, where $\boldsymbol{\ell}$ is a label. Furthermore, using the novel pairing randomization techniques, we have the choice whether $\mathbf{V}$ is represented as a single group element in the target group or pairing products of random group elements. While this construction is structure-preserving, it requires a little work-around to handle its pairing product equations which are slightly more demanding than that which GS pairing-product equations provide. This is done by introducing additional variables and using the other GS equation types when proving that $\mathbf{V}$ is computed correctly. Nevertheless, up to our knowledge, this is the first structure-preserving encryption in the literature, and it provides an important step to even more efficient schemes to come. Moreover, our encryption allows solving efficiently the problem of joint ciphertext computation for the first

time.

Related Work. The notion of *verifiable encryption* has long been used in the cryptographic community. Essentially, for certain encryption schemes, it allows to encrypt a message and then prove that the plaintext of the produced ciphertext satisfy certain useful properties, or interweave the two steps. As pointed out by [8, 37], the earlier work on verifiable encryption ignored the fact that IND-CPA security is not sufficient for most applications. Moreover, work prior to [37] used expensive "cut-and-choose" proofs, the computational and communication complexity of which grows linearly in the security parameter, rather than "sigma-protocols". While the work of [37] is quite efficient, like previous efficient constructions, it uses interactive proofs, and the only way to translate it non-interactive setting is using heuristics such the random oracle.

Non-interactive commitments of bits were used in previous constructions, e.g., [70], and used as a building block for efficient applications, e.g., [100]. However, they do not provide IND-CCA security as required for our needs.

While verifiable encryption is extremely useful, and could be realized for certain non-interactive scenarios using the (Linear) CS encryption and GS proofs, e.g. [31], it comes short of the full power of structure-preserving encryption. Also, it cannot be used to solve the above-mentioned problem of joint ciphertext computation.

1.2 Applications

Privacy-Preserving Schemes. As we mentioned earlier, NIZK proofs of knowledge of signatures are the essence of many cryptographic protocols. Clearly, combining GS proofs and structure-preserving signatures allows instantiating those constructions easily. From the many possible privacy-preserving applications, we consider

blind signatures and group signatures in Chapter 5.

First, we give an instantiation Fischlin's round-optimal blind signature framework [55] which has been an open problem since Crypto'06 and considered as difficult [79]. We describe the framework next. To obtain a signature from the signer, the user commits to a message and sends the commitment to the signer. Then, the signer signs the commitment and sends back the signature. The user produces a NIZK proof of knowledge of a commitment, an opening of the commitment to that message, and a signature on the commitment. This proof constitutes a blind signature for the message. Despite its simplicity, the scheme has not been instantiated efficiently in the standard model because it requires a signature scheme which signs trapdoor commitments and whose verification equations should mesh well with the GS proof system.

We then revisit group signatures which have enjoyed much interest since they were introduced by Chaum and van Heyst [45] almost twenty years ago. Most previous constructions, [38, 10, 22, 35, 17, 28, 29, 66] among others, could be viewed as unsatisfactory in some aspect: relying on the random-oracle heuristic, satisfying weaker security definitions, or not being efficient. The scheme by Groth [67] is both practical and satisfies the strengthened security definitions of [17]. However, it does not support concurrent join of new users. Using our signature schemes in combination with the GS proof system and an appropriate encryption scheme [80, 99], we overcome this shortcoming and construct a group signature scheme under the strongest security definitions which supports concurrent join while achieving comparable efficiency. Our construction follows a common approach used, e.g., in [38, 67]. The dynamic join protocol between a group member and the issuer simply consists in the issuer signing the member's verification key. To sign a message m, the member signs the message using her secret key and produces a NIWI proof of knowledge of a verification key, a

signature on that key by the issuer, and a signature on the message under that key.

Joint Ciphertext Computation We consider the two-party protocol for joint CCA ciphertext computation with respect to a third-party public key, in which the two parties provide verifiable inputs; also, we require that only one of the parties learn the output where the other party gets no output. While the desired properties of this protocol might not be immediately clear, those are exactly the right requirements of a building block when constructing *oblivious revocation authority* [32] to handle disputes between users and service providers. While existing solutions seemingly solve this problem for such revocation authority, they require each party to provide too much identifying information to the authority.

While quite simply formulated, the two-party protocol cannot be solved efficiently using known encryption schemes and proof systems (interactive or not) as they do not work well together in this case. In particular, the practical verifiable encryption of [37] comes short because the CCA encryption relies on cryptographic hash function which prevents achieving the two competing requirements, i.e. the first party outputs a ciphertext of the joint inputs, but is unable to produce any related ciphertext, and the second party learns no information about the ciphertext.

Leakage-Resilient Cryptography Traditionally, the security of cryptographic schemes has been analyzed in an idealized setting, where an adversary only sees the specified "input/output behavior" of a scheme, but has no other access to its internal secret state. Unfortunately, in the real world, an adversary may often learn some partial information about secret state via various *key leakage* attacks. That is why the cryptographic community has recently initiated the investigation of increasingly general (formally modeled) classes of leakage attacks, with the aim of constructing

leakage-resilient cryptographic schemes that remain provably secure even in the presence of such attacks.

We consider the bounded leakage model recently introduced recently by Akavia et al. [5], which has already attracted a lot of attention [83, 7, 75, 6]. In this model, an adversary is allowed to learn the output of any efficiently computable function of the secret key while being constrained that the information learned is bounded by ℓ bits. Unfortunately, we observe that the existing solutions of "leakage-resilient" encryption and signature schemes fail to satisfy at least one of the following desirable properties: **efficiency** — while we are interested in a modular cryptographic construction, it should have some efficient instantiation based on standard cryptographic assumptions; **strong security** — the construction should satisfy the standard security definitions, i.e., EUF-CMA for signatures and IND-CCA for encryption; **leakage flexibility** — it should be possible to set the parameters of the scheme so that the relative leakage $\ell/|\mathtt{sk}|$ is arbitrarily close to 1.

Looking at the existing schemes, we notice that they can be largely divided into two categories: efficient schemes with some *inherent* limitation to achieve relative leakage approaching 1 and more theoretical schemes [83, 75] which achieve good relative leakage but rely on the notion of *simulation-sound* NIZK (ss-NIZK) proofs [95]. Without getting into the definition of ss-NIZK proof here, we point out that the existing cryptographic machinery does not allow us to instantiate non-trivial ss-NIZK proofs efficiently. The work of [66] construct ss-NIZK proofs for practical languages and uses them them to construct group signatures, but the resulting schemes has signature size of "thousands or perhaps even millions of group elements" [67]. On the other hand, the recent breakthrough of Groth and Sahai [71] allows us to construct efficient NIZK proofs for a non-trivial class of languages. While the techniques of [66] could be applied to GS proofs to achieve ss-NIZK proofs, the resulting proofs are *sig-*

nificantly less efficient. Therefore, we generalize the existing construction sufficiently, so that they rely only on regular NIZKs, in the hope that we can then instantiate them efficiently using the powerful Groth-Sahai techniques.

In the end, this is indeed what we realize. The generic leakage-resilient signature scheme presented in Section 4.3.3, which generalizes the construction of [75], can be easily instantiated with GS proofs and a *structure-preserving second-preimage resistant relations* derived from our structure-preserving commitments. Furthermore, in the line of that work, we abstract the notion of *true-simulation extractable NIZK* proofs which we discuss next. Equipped with this powerful and efficiently realizable notion, we are also able to construct efficient leakage-resilient encryption scheme that satisfies the desired properties described above.

1.3 Simulation Extractability Revisited

In the process of generalizing the existing theoretical leakage resilient schemes [83, 75], we abstract away an elegant notion of independent interest: *true-simulation extractable* (tSE) NIZK proofs. While quite similar to the notion of simulation-sound extractable NIZK proofs [66], it involves a subtle but rather important difference: whether the adversary has oracle access to simulated proofs for arbitrary (even false) statements or only true ones. Intuitively, both the Naor-Segev's leakage-resilient CCA encryption [83] and Katz-Vaikuntanathan's leakage-resilient signature scheme [75] used the technique of encrypting a witness x for some relation $\mathtt{R}$, and then providing a ss-NIZK proof π that the ciphertext $\mathfrak{c}$ indeed contains an encryption of a valid witness x. The main reason for using this technique is to allow the reduction to extract a valid witness from any "new" valid pair $(\mathfrak{c}^\star, \pi^\star)$ produced by the attacker $\mathcal{A}$ (who saw many such valid pairs earlier). We abstract this property into the tSE notion

mentioned above (of which the above mentioned technique is a specific example, where the pair $(\mathfrak{c}, \pi)$ together makes up a single tSE-NIZK proof). We believe that true-simulation extractability, as we abstract it, is *precisely* the right notion for generalizing and proving the security of the previous constructions. Moreover, using Groth-Sahai proofs, we are able to instantiate these new generic constructions efficiently.

Chapter 2

Preliminaries

We denote the security parameter as λ, which customarily is given to setup algorithms in the form of 1^λ. We say that an algorithm is *efficient* if it runs in probabilistic polynomial time in the security parameter. Throughout this work, all algorithms are considered to be efficient unless explicitly stated otherwise. With $\mathsf{negl}(\lambda)$, or simply a *negligible* function for an implicit security parameter, we denote a function f such that $|f(\lambda)| < o(1/\lambda^c)$ for every positive integer c.

2.1 Basic Definitions

2.1.1 Digital Signatures

Definition 1 (Digital Signature Scheme)**.** *A digital signature scheme* SIG *is a tuple of algorithms* $(\mathsf{SIG.Key}, \mathsf{SIG.Sign}, \mathsf{SIG.Vrf})$ *defined as follows:*

$\mathsf{SIG.Key}(1^\lambda)$: *A randomized key generation algorithm that takes a security parameter* 1^λ *and generates a verification key* `vk` *and a signing key* `sk`*. The verification key also determines a message space* $\mathcal{M}$*.*

$\mathsf{SIG.Sign}(\mathtt{sk}, m)$: *A randomized signature generation algorithm that computes a signature σ for the input message m using the signing key* $\mathtt{sk}$.

$\mathsf{SIG.Vrf}(\mathtt{vk}, m, \sigma)$: *A verification algorithm that outputs 1 for acceptance or 0 for rejection depending on whether (m, σ) is a valid pair with respect to* $\mathtt{vk}$.

A signature scheme must provide correctness in the sense that for properly generated keys, a signature on a message generated with $\mathsf{SIG.Sign}$ is always accepted. The security parameter 1^λ allows $\mathsf{SIG.Key}$ implicitly to select an appropriate algebraic setting. $\mathsf{SIG.Key}$ may also take some other parameters if necessary.

We use standard notion of *existential unforgeability against adaptive chosen message attacks* [64] (EUF-CMA in short) formally defined as follows.

Definition 2 (Existential Unforgeability against Adaptive Chosen Message Attacks)**.** *A signature scheme is existentially unforgeable against adaptive chosen message attacks if, for any polynomial-time adversary $\mathcal{A}$, the following experiment returns* 1 *with negligible probability.*

$\mathrm{Exp}_{EUF\text{-}CMA}$:

$(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathsf{SIG.Key}(1^\lambda)$

$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}(\cdot)}(\mathtt{vk})$

Return 1 if $m^\star \notin Q_m$ and $\mathsf{SIG.Vrf}(\mathtt{vk}^\star, m^\star, \sigma^\star) = 1$. Return 0 otherwise.

$\mathcal{O}_{\mathsf{sign}}(\cdot)$ is a signing oracle that takes message m and returns $\sigma \leftarrow \mathsf{SIG.Sign}(\mathtt{sk}, m)$. Q_m denotes the set of messages submitted to $\mathcal{O}_{\mathsf{sign}}(\cdot)$. By requiring $(m^\star, \sigma^\star) \notin Q_{m,\sigma}$ where $Q_{m,\sigma}$ is the set of pairs of messages and their corresponding singatures observed by $\mathcal{O}_{\mathsf{sign}}(\cdot)$, we have the notion of Strong EUF-CMA (denoted by sEUF-CMA for short).

2.1.2 Public-Key Encryption

Definition 3 (Public-Key Encryption)**.** *A public-key encryption scheme* ENC *is a tuple of algorithms* $(\mathsf{ENC.Key}, \mathsf{ENC.Enc}, \mathsf{ENC.Dec})$ *such that:*

$\mathsf{ENC.Key}(1^\lambda)$: *A randomized key generation algorithm that takes security parameter* 1^λ *and generates a public key* $\mathtt{pk}$ *and a secret key* $\mathtt{sk}$*. A message space* $\mathcal{M}$ *is implicitly associated with* $\mathtt{pk}$*.*

$\mathsf{ENC.Enc}(\mathtt{pk}, m)$: *A randomized encryption algorithm that computes a ciphertext* $\mathfrak{c}$ *for the input message* m *with respect to the public key* $\mathtt{pk}$*.*

$\mathsf{ENC.Dec}(\mathtt{sk}, \mathfrak{c})$: *A decryption algorithm that recovers the plaintext* m *from the ciphertext* $\mathfrak{c}$ *using the secret key* $\mathtt{sk}$*. If the ciphertext is invalid, it returns* $\perp$*.*

An encryption scheme must provide correctness, i.e., for any properly generated key pair any message encrypted with $\mathsf{ENC.Enc}$ should be recovered using $\mathsf{ENC.Dec}$ from the ciphertext with the corresponding secret key. In this work, the encryption schemes work over groups of prime order. The security parameter 1^λ allows $\mathsf{ENC.Key}$ to select such a group of an appropriate size. $\mathsf{ENC.Key}$ may also take additional parameters if necessary.

We use the standard notion of *indistinguishability against chosen-ciphertext attack* (IND-CCA) [84, 53] formally defined as follows.

Definition 4 (Indistinguishability against Chosen-Ciphertext Attack)**.** *An encryption scheme is indistinguishable against chosen-ciphertext attack if for any polynomial-time adversary* $\mathcal{A}$ *the following experiment returns* 1 *with probability* $\frac{1}{2} + \mathsf{negl}(\lambda)$*.*

$\mathrm{Exp}_{\mathit{IND\text{-}CCA}}$:

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{ENC.Key}(1^\lambda)$

$(m_0, m_1, \mathrm{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(\mathtt{pk})$

$b \leftarrow \{0, 1\}$

$\mathfrak{c}^\star \leftarrow \mathsf{ENC.Enc}(\mathtt{pk}, m_b)$

$b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\cdot)}(\mathrm{state}, \mathfrak{c}^\star)$

Return 1 if $b = b^\star$; 0, otherwise.

$\mathcal{O}_{\mathsf{dec}}(\cdot)$ is a decryption oracle that takes a ciphertext $\mathfrak{c}$ and returns a message $m \leftarrow \mathsf{ENC.Dec}(\mathtt{sk}, \mathfrak{c})$ before b is chosen, and returns $m \leftarrow \mathsf{ENC.Dec}(\mathtt{sk}, \mathfrak{c})$ only if $\mathfrak{c} \neq \mathfrak{c}^\star$ after that. If $\mathcal{A}$ tries to submit $\mathfrak{c}^\star$ to the oracle, the query gets rejected (or otherwise the adversary can trivially recover b).

We can also define a labeled encryption ENC in which a message is encrypted and decrypted according to a public label L. We write $\mathsf{ENC.Enc}^L(m)$ to denote the encryption of a message m under a label L. Similarly, we use $\mathsf{ENC.Dec}^L(\mathfrak{c})$ to denote the decryption of ciphertext $\mathfrak{c}$ under the label L. In this case, we extend the correctness to require $\mathsf{ENC.Dec}^L(\mathsf{ENC.Enc}^L(m)) = m$. Also, the experiment defined in Definition 4 is modified as follows. A query to the decryption oracle now consists of a ciphertext $\mathfrak{c}$ and a label L, to which the oracle responds with $m = \mathsf{ENC.Dec}^L(\mathtt{sk}, \mathfrak{c})$. In the challenge generation stage, $\mathcal{A}$ outputs a label $L^\star$ as well as messages m_0, m_1 and the challenger computes $\mathfrak{c}^\star \leftarrow \mathsf{ENC.Enc}^{L^\star}(\mathtt{pk}, m_b)$. Finally, in the second stage of decryption queries the adversary is allowed to ask for decryptions of any ciphertext $\mathfrak{c}$ under a label L for which $(L, \mathfrak{c}) \neq (L^\star, \mathfrak{c}^\star)$.

2.1.3 Trapdoor Commitments

Definition 5 (Trapdoor Commitment Scheme)**.** *A trapdoor commitment scheme* TC *is a tuple of algorithms* $(\mathsf{TC.Key}, \mathsf{TC.Com}, \mathsf{TC.Vrf}, \mathsf{TC.Sim}, \mathsf{TC.Open})$ *such that:*

$\mathsf{TC.Key}(1^\lambda)$: *A randomized key generation algorithm that takes security parameter* 1^λ *and generates a commitment key* $\mathtt{ck}$ *and a secret trapdoor key* $\mathtt{tk}$*. The commitment key implicitly defines a message space* $\mathcal{M}$*.*

$\mathsf{TC.Com}(\mathtt{ck}, m)$: *A randomized algorithm that outputs* $(\mathfrak{c}, \mathfrak{d})$*, where* $\mathfrak{c}$ *is a commitment of the input message* m *and* $\mathfrak{d}$ *is the corresponding decommitment value.*

$\mathsf{TC.Vrf}(\mathtt{ck}, \mathfrak{c}, m, \mathfrak{d})$: *A verification algorithm that verifies whether the commitment* $\mathfrak{c}$ *corresponds to the message* m *with an opening* $\mathfrak{d}$ *with respect to* $\mathtt{ck}$*. If so, returns* 1*; otherwise,* 0*.*

$\mathsf{TC.Sim}(\mathtt{ck}, \mathtt{tk})$: *A randomized algorithm takes the trapdoor key and computes a equivocal commitment* $\mathfrak{c}$ *and a corresponding equivocation key* $\mathtt{ek}$*.*

$\mathsf{TC.Open}(\mathtt{ck}, \mathfrak{c}, m, \mathtt{ek})$: *An equivocation algorithm algorithm takes an equivocal commitment* $\mathfrak{c}$*, a message* m*, and an equivocation key* $\mathtt{ek}$*, and outputs a decommitment* $\mathfrak{d}$ *for which* $\mathsf{TC.Vrf}(\mathtt{ck}, \mathfrak{c}, m, \mathfrak{d}) = 1$*.*

Such trapdoor commitment scheme has to be correct: any commitment/decommitment pair produced by $\mathsf{TC.Com}$ for a message m should satisfy the verification algorithm $\mathsf{TC.Vrf}$. We say that a trapdoor commitment scheme TC is secure if it satisfies the following definitions of *perfect hiding*, *computational binding*, and *perfect trapdoor* [72, 91, 68]:

Definition 6 (Perfect Hiding)**.** *A commitment scheme is said to be perfectly hiding if for every adversary* $\mathcal{A}$*, the following experiment returns 1 with probability* $\frac{1}{2}$*.*

$\mathrm{Exp}_{\mathit{Perfect\ Hiding}}$:

$(\mathtt{ck}, \mathtt{tk}) \leftarrow \mathsf{TC.Key}(1^\lambda)$

$(m_0, m_1, \mathrm{state}) \leftarrow \mathcal{A}(\mathtt{ck})$

$b \leftarrow \{0, 1\}$

$\mathfrak{c} \leftarrow \mathsf{TC.Com}(\mathtt{ck}, m_b)$

$b^\star \leftarrow \mathcal{A}(\mathrm{state}, \mathfrak{c})$

Return 1 if $b = b^\star$; otherwise 0.

Definition 7 (Computational Binding)**.** *A commitment scheme is computationally binding if for every polynomial time adversary $\mathcal{A}$, the following experimetn returns 1 with negligible probability.*

$\mathrm{Exp}_{\mathit{Comp.Binding}}$:

$(\mathtt{ck}, \mathtt{tk}) \leftarrow \mathsf{TC.Key}(1^\lambda)$

$(\mathfrak{c}, m_0, m_1, \mathfrak{d}_0, \mathfrak{d}_1) \leftarrow \mathcal{A}(\mathtt{ck})$

return 1 if $m_0 \neq m_1$ and $1 = \mathsf{TC.Vrf}(\mathtt{ck}, \mathfrak{c}, m_0, \mathfrak{d}_0) = \mathsf{TC.Vrf}(\mathtt{ck}, \mathfrak{c}, m_1, \mathfrak{d}_1)$;

otherwise, return 0.

Definition 8 (Perfect Trapdoor)**.** *A trapdoor commitment scheme is a perfect trapdoor one if for every polynomial time adversary $\mathcal{A}$, the following experiment returns 1 with probability $\frac{1}{2}$.*

$\mathrm{Exp}_{\textit{Perfect Trapdoor}}$:

$(\mathtt{ck}, \mathtt{tk}) \leftarrow \mathsf{TC.Key}(1^\lambda)$

$(m, \mathrm{state}) \leftarrow \mathcal{A}(\mathtt{ck})$

$b \leftarrow \{0, 1\}$

if $b = 0$

$(\mathfrak{c}, \mathtt{ek}) \leftarrow \mathsf{TC.Sim}(\mathtt{ck}, \mathtt{tk})$

$\mathfrak{d} \leftarrow \mathsf{TC.Open}(\mathtt{ck}, \mathfrak{c}, m, \mathtt{ek})$

else

$(\mathfrak{c}, \mathfrak{d}) \leftarrow \mathsf{TC.Com}(\mathtt{ck}, m)$

$b^\star \leftarrow \mathcal{A}(\mathrm{state}, \mathfrak{c}, \mathfrak{d})$

return 1 if $b = b^\star$; otherwise return 0.

2.1.4 Non-Interactive Zero-Knowledge Proofs

We start by recalling the notion of *non-interactive zero-knowledge (NIZK)* [19, 54]. In fact, we use the stronger notion of *composable* NIZK [66].

Definition 9 (Non-Interactive Zero-Knowledge)**.** *Let* $\mathtt{R}$ *be an NP relation on pairs* (x, y) *with a corresponding language* $\mathfrak{L}_\mathtt{R} = \{y \mid \exists\, x \text{ s.t. } (x, y) \in \mathtt{R}\}$*. A NIZK proof system* Π *for a relation* $\mathtt{R}$ *is a tuple of algorithms* $(\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ *defined as follows:*

$\Pi.\mathsf{Crs}(1^\lambda)$: *A randomized algorithm generating a common reference string* $\mathtt{crs}$*.*

$\Pi.\mathsf{Prove}(\mathtt{crs}, y, x)$: *A randomized algorithm that outputs a proof* π *that* $\mathtt{R}(x, y) = 1$*.*

$\Pi.\mathsf{Vrf}(\mathtt{crs}, y, \pi)$: *A verification algorithm that verifies whether proof* π *that* $y \in \mathfrak{L}_{\mathtt{R}}$ *is correct. If it is, the algorithm outputs* 1*; otherwise,* 0.

A NIZK proof system Π is required to satisfy *perfect correctness*, *perfect soundness*, and *(composable) zero knowledge* defined as follows:

Definition 10 (Perfect Correctness)**.** Π *is said to be perfectly correct if for every adversary* $\mathcal{A}$ *the following experiment never returns 1.*

$\mathrm{Exp}_{\textit{Perfect Correctness}}$:

$\mathtt{crs} \leftarrow \mathsf{TC.Key}(1^\lambda)$

$(y, x) \leftarrow \mathcal{A}(\mathtt{crs})$

$\pi \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, y, x)$

return 1 if $(x, y) \in \mathtt{R}$ and $\Pi.\mathsf{Vrf}(\mathtt{crs}, y, \pi) = 0$

ohterwise, return 0.

Definition 11 (Perfect Soundness)**.** Π *is perfectly sound if for every adversary* $\mathcal{A}$ *the following experiment never returns 1.*

$\mathrm{Exp}_{\textit{Perfect Soundness}}$:

$\mathtt{crs} \leftarrow \mathsf{TC.Key}(1^\lambda)$

$(y, \pi) \leftarrow \mathcal{A}(\mathtt{crs})$

return 1 if $\Pi.\mathsf{Vrf}(\mathtt{crs}, y, \pi) = 1$ and $y \notin \mathfrak{L}_{\mathtt{R}}$;

otherwise, return 0.

To define zero-knowledge, we need to specify two randomized algorithms. The first one, $\Pi.\mathsf{SimCrs}(1^\lambda)$, takes a security parameter and generates a simulated common reference string with a corresponding trapdoor key $\mathtt{tk}$. The second, a simulator,

$\Pi.\mathsf{Sim}(\mathtt{crs}, y, \mathtt{tk})$ takes as input a statement and the trapdoor key, but not any witnesses, and outputs a proof π for which $\Pi.\mathsf{Vrf}(\mathtt{crs}, y, \pi)$ accepts. Note the simulation algorithm can be used to produce proofs for any statement since it does not check whether $y \in \mathfrak{L}_{\mathtt{R}}$ or not.

Definition 12 ((Composable) Zero-Knowledge)**.** Π *is said to be zero-knowledge if for any polynomial time adversaries* $\mathcal{A}$ *and* $\mathcal{B}$*, both experiments defined next return* 1 *with probabilities* $\frac{1}{2} + \mathsf{negl}(\lambda)$*. If we let* $\mathcal{B}$ *be computationally unbounded and require* $\mathrm{Exp}_{\text{SIM-IND}}$ *to return* 1 *with probability* $\frac{1}{2}$*, we say that* Π *has a perfect zero-knowledge simulation.*

$\mathrm{Exp}_{\textit{CRS-IND}}$:

$b \leftarrow \{0,1\}$

if $b = 0$, $\mathtt{crs} \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$;

else, $(\mathtt{crs}, \mathtt{tk}) \leftarrow \Pi.\mathsf{SimCrs}(1^\lambda)$

$b^\star \leftarrow \mathcal{A}(\mathtt{crs})$

return 1 if $b = b^\star$ and 0 otherwise.

$\text{Exp}_{\textit{SIM-IND}}$:

$(\mathtt{crs}, \mathtt{tk}) \leftarrow \Pi.\mathsf{SimCrs}(1^\lambda)$

$(y, x, \text{state}) \leftarrow \mathcal{B}(\mathtt{crs})$

$\pi_0 \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, y, x)$

$\pi_1 \leftarrow \Pi.\mathsf{Sim}(\mathtt{crs}, y, \mathtt{tk})$

$b \leftarrow \{0, 1\}$

$b^\star \leftarrow \mathcal{B}(\text{state}, \pi_b, \mathtt{tk})$

return 1 if $b = b^\star$ and $(x, y) \in \mathtt{R}$;

otherwise, return 0.

Sometimes, we will resort to a weaker form of proofs, i.e., non-interactive witness-indistinguishable (NIWI) proof system. The difference from the above definition of NIZK is that simulation indistinguishability is replaced by witness indistinguishability:

Definition 13 ((Composable) Witness-Indistinguishability)**.** Π *is said to be witness-indistinguishable if for any polynomial time adversaries* $\mathcal{A}, \mathcal{B}$ *the experiment* $\text{Exp}_{\text{CRS-IND}}$*, as defined above, and the following experiment* Exp_{WI} *return* 1 *with probabilities* $\frac{1}{2} + \mathsf{negl}(\lambda)$*. If we let* $\mathcal{B}$ *be computationally unbounded and require* Exp_{WI} *to return* 1 *with probability* $\frac{1}{2}$*, we say that* Π *is perfect witness distinguishable for the simulated common reference string.*

Exp_{WI}:

$\mathtt{crs} \leftarrow \Pi.\mathsf{SimCrs}(1^\lambda)$

$(y, x_0, x_1, \mathrm{state}) \leftarrow \mathcal{B}(\mathtt{crs})$

$b \leftarrow \{0,1\}$

$\pi \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, y, x_b)$

$b^\star \leftarrow \mathcal{B}(\mathrm{state}, \pi, \mathtt{tk})$

return 1 if $b = b^\star$ and $(x_0, y), (x_1, y) \in \mathtt{R}$;

otherwise, return 0.

For our purposes later, it will be slightly more convenient to use the notion of *(same-string) NIZK argument* from [96]. That is, the common reference string is generated using $\Pi.\mathsf{SimCrs}$, rather than using $\Pi.\mathsf{Crs}$, along with the corresponding trapdoor key; and the originally defined $\Pi.\mathsf{Crs}$ is used only in the proof of soundness. In that case, the soundness holds only for polynomial time adversaries. Note, however, that the definitions and constructions given in Section 3 can be easily extended to the case of NIZK proofs.

2.1.5 Leakage-Resilient Cryptographic Primitives

Following previous works [5, 83], we model leakage attacks by giving the adversary access to a *leakage oracle*, which he can adaptively access to learn leakage on the secret key. A leakage oracle $\mathcal{O}^{\lambda,\ell}(\cdot)$ is parameterized by a leakage parameter ℓ, a security parameter λ, and (implicitly) a secret key $\mathtt{sk}$. A query to the oracle consists of a function $\hbar_i : \{0,1\}^* \to \{0,1\}^{\alpha_i}$, to which the oracle answers with $y_i = \hbar_i(\mathtt{sk})$. We only require that the functions $\hbar_i$ be efficiently computable, and the total number

of bits leaked is $\sum_i \alpha_i \leq \ell$.

Definition 14 (Leakage Resilient EUF-CMA Signatures(ℓ-LR-EUF)). *A signature scheme* SIG *is* ℓ-leakage resilient *if it is existentially unforgeable against adaptive chosen message attacks in the presence of key leakage. That is if no polynomial time adversary wins with non-negligible probability the experiment* $\mathrm{Exp}_{\text{EUF-CMA}}$ *form Definition 2 modified so that* $\mathcal{A}$ *has an oracle access to* $\mathcal{O}^{\lambda,\ell}(\cdot)$ *in addition to* $\mathcal{O}_{\mathsf{sign}}(\cdot)$.

Definition 15 (Leakage Resilient IND-CCA Encryption(ℓ-LR-CCA)). *We say that an encryption scheme* ENC *is* ℓ-leakage resilient *IND-CCA if any polynomial time adversary* $\mathcal{A}$ *wins with probability* $\frac{1}{2} + \mathsf{negl}(\lambda)$ *the experiment* $\mathrm{Exp}_{\text{IND-CCA}}$ *from Definition 4 modified so that* $\mathcal{A}$ *has an oracle access to* $\mathcal{O}^{\lambda,\ell}(\cdot)$ *in addition to* $\mathcal{O}_{\mathsf{dec}}(\cdot)$ *in the first phase, i.e., before the challenge ciphertext is computed.*

Note that a 0-LR-CCA encryption is simply an IND-CCA encryption.

Definition 16 (Leakage Resilient CPA-Secure Encryption (ℓ-LR-CPA)). *We say that an encryption scheme* ENC *is* ℓ-leakage resilient IND-CPA *if any polynomial time adversary* $\mathcal{A}$ *wins with probability* $\frac{1}{2} + \mathsf{negl}(\lambda)$ *the experiment* $\mathrm{Exp}_{\text{IND-CCA}}$ *from Definition 4 modified so that* $\mathcal{A}$ *has an oracle access to* $\mathcal{O}^{\lambda,\ell}(\cdot)$ *in the first phase but no oracle access to* $\mathcal{O}_{\mathsf{dec}}(\cdot)$ *at all.*

Note that an encryption that is 0-LR-CPA is in fact IND-CPA secure, i.e., the standard notion of semantically secure encryption; we do not use it in this work.

Finally, we define the notion of *leakage resilient hard relations* which generalizes the notion of (leakage-resilient) one-way functions.

Definition 17 (Leakage Resilient Hard Relation(ℓ-LR-HR)). *A relation* $\mathtt{R}$ *with a randomized sampling algorithm* Sample *is an* ℓ-leakage resilient hard relation *if:*

For any $(y, x) \leftarrow \mathsf{Sample}(1^\lambda)$ *it holds that* $(x, y) \in \mathtt{R}$.

There is a polynomial-time algorithm that decides whether $(x, y) \in \mathtt{R}$.

For any polynomial time adversary $\mathcal{A}$ *with an access to the leakage oracle* $\mathcal{O}^{\lambda,\ell}(\cdot)$, *the following experiment returns* 1 *with negligible probability.*

$\mathrm{Exp}_{\text{LR-}\mathtt{R}}$*:*

$(y, x) \leftarrow \mathsf{Sample}(1^\lambda)$

$x^\star \leftarrow \mathcal{A}^{\mathcal{O}^{\lambda,\ell}(\cdot)}(y)$

output 1 *if* $\mathtt{R}(x^\star, y) = 1$*; and* 0 *otherwise.*

Notice that without loss of generality, we can assume that $\mathcal{A}$ *queries* $\mathcal{O}^{\lambda,\ell}(\cdot)$ *only once with a function* $\hbar$ *whose output is* ℓ *bits.*

2.2 Notation and Common Setup

Let $\mathbb{G}$ be a group of prime order p. For clarity, we are write groups elements, such as $\mathbf{g}, \mathbf{h} \in \mathbb{G}$, in boldface to distinguish them easily from exponents such as $x, y, \alpha, \beta \in \mathbb{Z}_p$. The identity element of the group is denoted with $\mathbf{1}_\mathbb{G}$ or simply $\mathbf{1}$. Also, when working in a group equipped with a bilinear map, a.k.a. pairing, we write in capital elements in the target group, e.g., $\mathbf{A}, \mathbf{B} \in \mathbb{G}_T$. Occasionally, we deviate from these rules when denoting keys associated with a scheme, e.g., $\mathtt{pk}$, $\mathtt{sk}$, or the main result of our schemes, i.e., a signature σ, a ciphertext $\mathfrak{c}$, a NIZK proof π, etc., without emphasizing their particular structure.

When using vectors, our presentation benefits from the following notation. For two vectors $\vec{\mathbf{a}}, \vec{\mathbf{b}} \in \mathbb{G}^n$, when we say that we "multiple" them, we mean component-wise multiplication of each of the n components, i.e. $\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} = (\mathbf{a}_1\mathbf{b}_1, \ldots, \mathbf{a}_n\mathbf{b}_n)$. This extends to a product of k vectors $\prod_{i=1}^k \vec{\mathbf{a}}_i = (\prod_{i=1}^k \mathbf{a}_{i,1}, \ldots, \prod_{i=1}^k \mathbf{a}_{i,n})$. Lastly, for

a vector of group elements $\vec{\mathbf{a}} \in \mathbb{G}^n$ and a vector of scalars $\vec{\chi} \in \mathbb{Z}_p^n$, we define the "inner-product" $\langle \vec{\mathbf{a}}, \vec{\chi} \rangle = \prod_{i=1}^{n} \mathbf{a}_i^{\chi_i} \in \mathbb{G}$.

Following previous works, e.g. [23], we denote with $\hat{e}$ an efficiently computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which is:

- *Bilinear:* for all $\mathbf{a} \in \mathbb{G}_1, \mathbf{b} \in \mathbb{G}_2$, and $\alpha, \beta \in \mathbb{Z}$, $\hat{e}(\mathbf{a}^\alpha, \mathbf{b}^\beta) = \hat{e}(\mathbf{a}, \mathbf{b})^{\alpha\beta}$;
- *Non-degenerate:* the map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbf{1}_{\mathbb{G}_T}$. In particular, if $\mathbb{G}_1$ and $\mathbb{G}_2$ are prime order groups generated, respectively, by $\mathbf{g}$ and $\tilde{\mathbf{g}}$, then $\hat{e}(\mathbf{g}, \tilde{\mathbf{g}})$ is a generator of $\mathbb{G}_T$.

Moreover, let $\Lambda := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, \mathbf{g}, \tilde{\mathbf{g}})$ be a description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p equipped with such efficient bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It also includes a random generator $\mathbf{g}$ of $\mathbb{G}_1$ and $\tilde{\mathbf{g}}$ of $\mathbb{G}_2$. By $\mathbb{G}_1^*$ we denote $\mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$, and similarly for $\mathbb{G}_2^*$ and $\mathbb{G}_T^*$. By Λ_{sym} we denote a special case of Λ where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Similarly, Λ_{xdh} denotes a case where the Decision Diffie-Hellman (DDH) assumption holds for $\mathbb{G}_1$ ($\mathrm{DDH}_{\mathbb{G}_1}$ in short). This setting implies that there is no efficiently computable homomorphism $\mathbb{G}_1 \to \mathbb{G}_2$. And Λ_{sxdh} denotes a case where the DDH assumption holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$. This means that no efficient mapping is available for either direction. The Λ_{xdh} and Λ_{sxdh} settings are usually referred to as the (Symmetric) External Diffie-Hellman Assumption (SXDH) [98, 22, 60, 101]. For differences of these settings in practice, we refer to [59]. A scheme (or an assumption or a proof) designed and proven in one setting may not necessarily go through in a different setting. In particular, if the scheme is for Λ_{sym} and uses the homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, it does not work, or not known to be secure when used with Λ_{xdh} or Λ_{sxdh}. We treat Λ as a common parameter implicitly given to all algorithms of interest. However, we present our constructions with care so that it is clear in which setting they work and are secure.

2.3 Assumptions

First, we review the well-known DDH assumption:

Assumption 1 (Decisional Diffie-Hellman Assumption (DDH))**.** *Let* $\mathbb{G}$ *be a group of prime order* p*. Let* $\mathbf{g}_1, \mathbf{g}_2 \leftarrow \mathbb{G}$ *and* $r, r_1, r_2 \leftarrow \mathbb{Z}_p$*. The decisional Diffie-Hellman (*DDH*) assumption states that the following two distributions are computationally indistinguishable:* $(\mathbb{G}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2})$ *and* $(\mathbb{G}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_1^{r}, \mathbf{g}_2^{r})$.

When working in the Λ_{sym} setting, we often use a generalization of DDH:

Assumption 2 (Decisional Linear Assumption (DLIN)[22])**.** *Let* $\mathbb{G}$ *be a group of prime order* p *and let* $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \leftarrow \mathbb{G}$*. The* DLIN *assumption holds if for* $r, s, t \leftarrow \mathbb{Z}_p$ *no probabilistic polynomial time adversary could distinguish between*

$$(\mathbb{G}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_1^{r}, \mathbf{g}_2^{s}, \mathbf{g}_3^{r+s}) \qquad \textit{and} \qquad (\mathbb{G}, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_1^{r}, \mathbf{g}_2^{s}, \mathbf{g}_3^{t}),$$

with a non-negligible advantage.

Both DDH and DLIN can be viewed as instances of an even more general assumption, the K-linear assumption, for $K = 1$ and $K = 2$ respectively. Although we do not use K-linear for $K > 2$, it is useful when describing generic schemes which depend on K. So, for completeness, we define it here.

Assumption 3 (K-Linear Assumption (K-linear)[74, 99])**.** *Let* $\mathbb{G}$ *be a group of prime order* p*. Also, let* $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K \leftarrow \mathbb{G}$ *and let* $r_0, r_1, \ldots, r_K \leftarrow \mathbb{Z}_p$*. The* K-linear *assumption holds if no probabilistic polynomial time adversary could distinguish between*

$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K, \mathbf{g}_1^{r_1}, \ldots, \mathbf{g}_K^{r_K}, \mathbf{g}_0^{\sum_{i=1}^{K} r_i}) \;\; \textit{and} \;\; (\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K, \mathbf{g}_1^{r_1}, \ldots, \mathbf{g}_K^{r_K}, \mathbf{g}_0^{r_0}),$$

with a non-negligible advantage.

Next, we introduce a simple assumption, called Double Pairing Assumption, that holds in asymmetric bilinear setting when (S)XDH holds, i.e., $\Lambda \in \{\Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$.

Assumption 4 (Double Pairing Assumption (DBP))**.** *Given* $\Lambda \in \{\Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$ *and* $(\mathbf{g}_z, \mathbf{g}_r) \leftarrow \mathbb{G}_1^{*2}$*, it is hard to find* $(\mathbf{z}, \mathbf{r}) \in \mathbb{G}_2^* \times \mathbb{G}_2^*$ *such that*

$$1 = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r}). \tag{2.1}$$

It is obvious that DBP does not hold for $\Lambda = \Lambda_{\mathsf{sym}}$ since $(\mathbf{z}, \mathbf{r}) = (\mathbf{g}_r^{-1}, \mathbf{g}_z) \neq (1, 1)$ fulfills the relation. On the other hand, we can show that DBP holds for $\Lambda \in \{\Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$ where DDH is assumed hard in $\mathbb{G}_1$.

Theorem 1. *If* $\mathrm{DDH}_{\mathbb{G}_1}$ *holds for* Λ*, then* DBP *holds for* Λ*.*

Proof. Assume that the DBP assumption does not hold and there is an adversary $\mathcal{A}$ that produces a pair $(\mathbf{z}, \mathbf{r}) \neq (1, 1)$ satisfying equation (2.1) for randomly chosen $\mathbf{g}_z, \mathbf{g}_r$ with non-negligible probability. We construct $\mathcal{B}$ which breaks $\mathrm{DDH}_{\mathbb{G}_1}$.

Let $\mathcal{B}$ be given a challenge tuple $(\mathbf{g}, \mathbf{g}_\alpha, \mathbf{g}_\beta, \mathbf{g}_\gamma)$, where $\mathbf{g}_\alpha = \mathbf{g}^\alpha$, $\mathbf{g}_\beta = \mathbf{g}^\beta$, and $\mathbf{g}_\gamma = \mathbf{g}^\gamma$. Then, $\mathcal{B}$ chooses $\delta \leftarrow \mathbb{Z}_p^*$ and runs $\mathcal{A}$ on the input $(\mathbf{g}^\delta, \mathbf{g}_\alpha^\delta)$ along with the appropriate public parameters. If $\mathcal{A}$ outputs $(\mathbf{z}, \mathbf{r}) \neq (1, 1)$ satisfying $\hat{e}(\mathbf{g}^\delta, \mathbf{z})\ \hat{e}(\mathbf{g}_\alpha^\delta, \mathbf{r}) = 1$, it is true that $\mathbf{z} = \mathbf{r}^{-\alpha}$. Then, it holds that $\hat{e}(\mathbf{g}_\beta, \mathbf{z})\ \hat{e}(\mathbf{g}_\gamma, \mathbf{r}) = \hat{e}(\mathbf{g}^\beta, \mathbf{r}^{-\alpha})\ \hat{e}(\mathbf{g}^\gamma, \mathbf{r}) = \hat{e}(\mathbf{g}, \mathbf{r})^{\gamma - \alpha\beta}$; that equation is equal to $\mathbf{1}$ if and only if $\alpha\beta = \gamma \mod p$.

Therefore, $\mathcal{B}$ has the same success probability of breaking $\mathrm{DDH}_{\mathbb{G}_1}$ as $\mathcal{A}$ of breaking the DBP assumption. □

We note that the DBP assumption could be viewed as a simpler version of the Simultaneous Triple Pairing Assumption (STP) [68]. The DBP assumption was introduced in an earlier version of this work, and independently in [69] (personal communication) by Groth, who also showed explicitly that DBP implies STP.

Next is an extension of DBP, called Simultaneous Double Pairing Assumption (SDP), which is a weaker assumption and can be justified in any setting $\Lambda \in \{\Lambda_{\mathsf{sym}}, \Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$ by a standard argument in the generic bilinear group model.

Assumption 5 (Simultaneous Double Pairing Assumption (SDP))**.** *Given* Λ *and* $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u) \leftarrow \mathbb{G}_1^{*4}$*, it is hard to find* $(\mathbf{z}, \mathbf{r}, \mathbf{u}) \in \mathbb{G}_2^{*3}$ *such that*

$$1 = \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}) \qquad \textit{and} \qquad 1 = \hat{e}(\mathbf{h}_z, \mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u}). \tag{2.2}$$

As shown in [44], SDP is implied by DLIN.

Finally, we introduce a novel assumption by extending SDP so that it should be hard to find another answer given several answers. Observe that, given an answer to an instance of SDP, one can easily yield more answers by exploiting the linearity of the relation to be satisfied. We eliminate such a linearity by multiplying random pairings to both sides of the equations in (2.2). The intuition is that, it should be hard to merge two random pairings $\hat{e}(\mathbf{s}, \mathbf{t})\, \hat{e}(\mathbf{s}', \mathbf{t}')$ into one equivalent pairing $\hat{e}(\mathbf{s}'', \mathbf{t}'')$. We call such a random part *flexible* as random pairings can be easily randomized or combined when their relation with respect to the same bases is known.

Assumption 6 (Simultaneous Flexible Pairing Assumption (SFP))**.** *Let* Λ *be a common parameter and let* $\mathbf{g}_z$*,* $\mathbf{h}_z$*,* $\mathbf{g}_r$*, and* $\mathbf{h}_u$ *be random generators of* $\mathbb{G}_1$*. Let* $(\mathbf{a}, \tilde{\mathbf{a}}), (\mathbf{b}, \tilde{\mathbf{b}})$ *be random pairs in* $\mathbb{G}_1 \times \mathbb{G}_2$*. For* $j = 1, \ldots, q$*, let* $R_j = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ *that satisfies*

$$\hat{e}(\mathbf{a}, \tilde{\mathbf{a}}) = \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r})\, \hat{e}(\mathbf{s}, \mathbf{t}) \quad \textit{and} \quad \hat{e}(\mathbf{b}, \tilde{\mathbf{b}}) = \hat{e}(\mathbf{h}_z, \mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u})\, \hat{e}(\mathbf{v}, \mathbf{w}). \tag{2.3}$$

Given $(\Lambda, \mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \mathbf{a}, \tilde{\mathbf{a}}, \mathbf{b}, \tilde{\mathbf{b}})$ *and uniformly chosen* $R_1, \ldots, R_q$*, it is hard to find* $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{s}^\star, \mathbf{t}^\star, \mathbf{u}^\star, \mathbf{v}^\star, \mathbf{w}^\star)$ *that fulfill relations in (2.3) under the restriction that* $\mathbf{z}^\star \neq 1$ *and* $\mathbf{z}^\star \neq \mathbf{z} \in \mathtt{R}_j$ *for every* R_j*.*

Theorem 2. *[3] For any generic algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ breaks* SFP *with ℓ group operations and pairings is bound by $\mathcal{O}(q^2 + \ell^2)/p$.*

Note that the following relation holds with respect to SFP and SDP.

Theorem 3. SFP $\Rightarrow$ SDP.

The proof of the theorem follows immediately by the following observation: given an answer $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star)$ to SDP, if one lets $(\mathbf{s}^\star, \mathbf{t}^\star, \mathbf{v}^\star, \mathbf{w}^\star) = (\mathbf{a}, \tilde{\mathbf{a}}, \mathbf{b}, \tilde{\mathbf{b}})$, then the resulting tuple $R^\star = (\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star, \mathbf{s}^\star, \mathbf{t}^\star, \mathbf{v}^\star, \mathbf{w}^\star)$ satisfies equations (2.3); and by the uniform choice $R_1, \ldots, R_q$, it breaks the SFP assumption with overwhelming probability.

Lastly, for completeness, we recall a simple assumption which plays a minor role.

Assumption 7 (Computational Co-Diffie-Hellman Assumption (co-CDH) [27])**.** *Given Λ and $(\mathbf{g}, \tilde{\mathbf{g}}, \tilde{\mathbf{g}}^\alpha) \in \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{G}_2^*$ for random $\alpha \in \mathbb{Z}_p^*$, it is hard to compute $\mathbf{g}^\alpha \in \mathbb{G}_1$.*

2.4 The Groth-Sahai Proof System

In this section, we review the NIZK proof system of Groth and Sahai [71] for proving that a system of equations is satisfiable. We give details for the type of equations used in this work, i.e., pairing-product (one-sided in the DLIN case) and one-sided multi-exponentiation. For full details and more general form of these types refer to [71]. In fact, we use the system mainly as a NIZK argument system, achieving only computational soundness. This can be done by running all the algorithms with a simulated CRS. Note that in the GS proof system, there are two types of common reference string (CRS) and those are computationally indistinguishable: the real one gives perfectly sounds proofs whereas the simulated one yields perfect witness indis-

tinguishable proofs, which could in many cases be transformed into zero-knowledge proofs.

When working under the K-linear assumption ($K = 1$ for SXDH and $K = 2$ for DLIN), the common reference strings for the proof system Π can be defined as $\mathtt{crs} = (\vec{\mathfrak{u}}_0, \vec{\mathfrak{u}}_1, \ldots, \vec{\mathfrak{u}}_K, \vec{\mathfrak{u}})$. Regardless of the $\mathtt{crs}$ type, $\vec{\mathfrak{u}}_i = (\mathfrak{u}_0, \mathbf{1}, \ldots, \mathbf{1}, \mathfrak{u}_i, \mathbf{1}, \ldots, \mathbf{1})$, for $i = 1, \ldots, K$, where $\mathfrak{u}_0, \ldots, \mathfrak{u}_K$ are randomly chosen group elements in $\mathbb{G}_1^*$. The secret key $\mathtt{sk}$ which enables extraction for a real CRS consists of the discrete logarithms of $\mathfrak{u}_1, \ldots, \mathfrak{u}_K$ with respect to $\mathfrak{u}_0$. Let's denote with $\mathfrak{U}$ the linear span $\mathsf{span}(\vec{\mathfrak{u}}_1, \ldots, \vec{\mathfrak{u}}_K)$, and note that $(\mathbf{g}, \mathbf{1}, \ldots, \mathbf{1}) \notin \mathfrak{U}$.

For a real CRS, which yields perfectly sound proofs, $\vec{\mathfrak{u}}_0 \leftarrow \mathfrak{U}$ and $\vec{\mathfrak{u}} \leftarrow \mathbb{G}_1^{K+1} \backslash \mathfrak{U}$. When the CRS is simulated, $\vec{\mathfrak{u}}_0 \leftarrow \mathbb{G}_1^{K+1} \backslash \mathfrak{U}$ and $\vec{\mathfrak{u}} \leftarrow \mathfrak{U}$. In the case of asymmetric pairings, i.e. in the Λ_{sxdh} setting, another set of vectors $\vec{\mathfrak{v}}_0, \vec{\mathfrak{v}}_1, \ldots, \vec{\mathfrak{v}}_K, \vec{\mathfrak{v}} \in \mathbb{G}_2^{K+1}$ is defined analogously for randomly chosen $\mathfrak{v}_0, \ldots, \mathfrak{v}_K \in \mathbb{G}_2^*$. Although in the symmetric setting Λ_{sym} we use only one-sided equations and a second set of vectors is not needed, we set $\vec{\mathfrak{v}} = \vec{\mathfrak{u}}$ and $\vec{\mathfrak{v}}_i = \vec{\mathfrak{u}}_i$, $i = 0, \ldots, K$, and use the two sets of vectors interchangeably for consistent notation (in the two settings).

The GS proof system gives a proof for a set of equations being satisfiable by committing to each witness component separately and computing corresponding proof elements for each of the equations. First we describe how those commitments are computed, and, then, how those proofs elements are computed and verified. Some of the notation if borrowed from [31].

Witness Commitments

Each witness is composed of several group elements, each in $\mathbb{G}_1$ or $\mathbb{G}_2$, and exponents from $\mathbb{Z}_p$. To commit to a group element $\mathbf{x} \in \mathbb{G}_1$, the prover chooses randomness $\vec{s} \leftarrow \mathbb{Z}_p^{K+1}$ and computes a commitment $\vec{\mathfrak{a}}_{\mathbf{x}} \leftarrow \mathsf{GSCom}(\{\vec{\mathfrak{u}}_i\}_{i=0}^K, \mathbf{x}; \vec{s}) \quad =$

$(\mathbf{x}, \mathbf{1}, \ldots, \mathbf{1}) \cdot \prod_{j=0}^{K} \vec{\mathfrak{u}}_j^{s_j}$. Group elements $\tilde{\mathbf{x}} \in \mathbb{G}_2$ are committed analogously as $\vec{\mathfrak{b}}_{\tilde{\mathbf{x}}} \leftarrow \mathsf{GSCom}(\{\vec{\mathfrak{v}}_i\}_{i=0}^{K}, \tilde{\mathbf{x}}; \vec{s})$ using the other set of vectors. Note that when `crs` is real, $\vec{\mathfrak{u}}_0 \in \mathfrak{U}$ and $\vec{\mathfrak{v}}_0 \in \mathfrak{V}$, so the commitments are perfectly binding. And using `sk` which consists of the discrete logarithms of $\mathfrak{u}_i$ with respect to $\mathfrak{u}_0$ and $\mathfrak{v}_i$ with respect to $\mathfrak{v}_0$, for $i = 1, \ldots, K$, one can extract the committed value since the commitment is essentially a linear encryption.

To commit to an exponent χ using randomness $\vec{t} \leftarrow \mathbb{Z}_p^K$, the prover computes $\vec{\mathfrak{a}}_\chi \leftarrow \mathsf{GSCom}((\vec{\mathfrak{u}}, \{\vec{\mathfrak{u}}_i\}_{i=1}^{K}), \chi; \vec{t}) = \vec{\mathfrak{u}}^{\chi} \prod_{j=1}^{K} \vec{\mathfrak{u}}_j^{t_j}$. Depending on the equation in which the witness is used, the exponent can be committed using the other set of vectors from `crs`, i.e., as $\vec{\mathfrak{b}}_\chi \leftarrow \mathsf{GSCom}((\vec{\mathfrak{v}}, \{\vec{\mathfrak{v}}_i\}_{i=1}^{K}), \chi; \vec{t})$. Committed exponents cannot be extracted efficiently like group elements despite the commitments being perfectly binding for a real common reference string.

One-sided Multi-exponentiation Equations

For an equation of the following type:

$$\mathbf{g}_0 = \mathbf{g}_1^{\chi_1} \mathbf{g}_2^{\chi_2} \ldots \mathbf{g}_n^{\chi_n}$$

where $\mathbf{g}_0, \ldots, \mathbf{g}_n \in \mathbb{G}_2$ are constants (one could view an equation being described by those constants) and $\chi_1, \ldots, \chi_n \in \mathbb{Z}_p$ are variables (the witness for which the equation is satisfiable), the proof elements are $\mathfrak{p}_1, \ldots, \mathfrak{p}_K$:

$$\mathfrak{p}_j = \prod_{i=1}^{n} \mathbf{g}_i^{t_{i,j}}, \ j = 1, \ldots, K,$$

where $\vec{t_i}$ is the randomness used to commit to χ_i, i.e. $\vec{\mathfrak{a}}_{\chi_i} = \mathsf{GSCom}(\chi_i; \vec{t_i})$.

When verifying a proof, for each equation $\mathbf{g}_0 = \mathbf{g}_1^{\chi_1} \mathbf{g}_2^{\chi_2} \ldots \mathbf{g}_n^{\chi_n}$ the verifier checks that the proof elements corresponding to the equation and the commitments satisfy

$$\prod_{i=1}^{n} \hat{E}(\vec{\mathfrak{a}}_i, \mathbf{g}_i) = \hat{E}(\vec{\mathfrak{u}}, \mathbf{g}_0) \cdot \prod_{j=1}^{K} \hat{E}(\vec{\mathfrak{u}}_j, \mathfrak{p}_j),$$

where $E : \mathbb{G}_1^{K+1} \times \mathbb{G}_2 \to \mathbb{G}_T^{K+1}$, sending $((\mathbf{a}_0, \ldots, \mathbf{a}_K), \mathbf{b})$ to $(\hat{e}(\mathbf{a}_0, \mathbf{b}), \ldots, \hat{e}(\mathbf{a}_K, \mathbf{b}))$, is a bilinear map.

The proofs for multi-exponentiation equations are zero knowledge (ZK). The size of a proof for set of M such equations being satisfiable with a witness of size N is $(K+1)N + KM$ group elements. Note again that $K = 1$ when working under SXDH and $K = 2$ under DLIN.

(One-sided) Pairing Product Equations

A pairing-product equation over variables $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{G}_1$ and $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n \in \mathbb{G}_2$ is defined as:

$$\prod_{i=1}^{n} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{x}}_i) \prod_{i=1}^{m} \hat{e}(\mathbf{x}_i, \tilde{\mathbf{g}}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} \hat{e}(\mathbf{x}_i, \tilde{\mathbf{x}}_j)^{c_{i,j}} \;=\; \mathbf{T}$$

where $\{\mathbf{g}_i\}_{i=1}^{n} \in \mathbb{G}_1$, $\{\tilde{\mathbf{g}}_i\}_{i=1}^{m} \in \mathbb{G}_2$, $\{c_{i,j}\}_{i=1,j=1}^{m,\ n} \in \mathbb{Z}_p$, and $\mathbf{T} \in \mathbb{G}_T$ are constants.

When the equations involve variables only in one of the groups, we could use simpler, one-sided, equations which also yield more efficient proofs:

$$\prod_{i=1}^{n} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{x}}_i) = \mathbf{T}.$$

We use this particular type in Section 4.3 and Section 5.6. The proof elements $\mathfrak{p}_0, \ldots, \mathfrak{p}_K$ are computed as:

$$\mathfrak{p}_j = \prod_{i=1}^{n} \mathbf{g}_i^{s_{i,j}},\ j = 0, \ldots, K,$$

where $\vec{s}_i$ is the randomness used to commit to $\tilde{\mathbf{x}}_i$, i.e. $\vec{\mathfrak{b}}_{\tilde{\mathbf{x}}_i} = \mathsf{GSCom}(\tilde{\mathbf{x}}_i; \vec{s}_i)$. When verifying a proof, for each equation $\prod_{i=1}^{n} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{x}}_i) = \mathbf{T}$ the verifier checks that the proof elements corresponding to the equation and the commitments satisfy

$$\prod_{i=1}^{n} \hat{E}(\mathbf{g}_i, \vec{\mathfrak{b}}_i) = (\mathbf{T}, 1, 1, \ldots, 1) \cdot \prod_{j=0}^{K} \hat{E}(\mathfrak{p}_j, \vec{\mathfrak{v}}_j),$$

where $E : \mathbb{G}_1 \times \mathbb{G}_2^{K+1} \to \mathbb{G}_T^{K+1}$, sending $(\mathbf{a}, (\mathbf{b}_0, \ldots, \mathbf{b}_K))$ to $(\hat{e}(\mathbf{a}, \mathbf{b}_0), \ldots, \hat{e}(\mathbf{a}, \mathbf{b}_K))$, is a bilinear map.

These proofs are only witness indistinguishable (WI), and for a set of M pairing product equations satisfiable with a witness of size N, the proof size is $(K+1)(M+N)$. When representation of $\mathbf{T}$ as a pairing product is know it could be transformed into ZK [71] but resulting in somewhat larger proofs.

In Section 4.3, we use $\mathbf{T}$ represented as $\mathbf{T}^{-1} = \hat{e}(\mathbf{g}_0, \tilde{\mathbf{x}}_0)$ where both $\mathbf{g}_0$ and $\tilde{\mathbf{x}}_0$ are constants. So, we could transform the above equation into an equation $\prod_{i=0}^{n} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{x}}_i) = 1$ and give a WI proof accordingly treating $\tilde{\mathbf{x}}_0$ as a part of the witness. Then, we produce a second commitment of $\vec{\mathfrak{b}}'_{\tilde{\mathbf{x}}_0} = \mathsf{GSCom}(\tilde{\mathbf{x}}_0; \vec{s}'_0)$, include its randomness $\vec{s}'_0$ and a NIZK proof that $\vec{\mathfrak{b}}_{\tilde{\mathbf{x}}_0}$ and $\vec{\mathfrak{b}}'_{\tilde{\mathbf{x}}_0}$ are commitment to the same message using a set of one-sided multi-exponentiation equations. This way, when the simulator has to produce a ZK proof for the equation, it samples any $(\tilde{\mathbf{x}}'_1, \ldots, \tilde{\mathbf{x}}'_n)$ along with the appropriate $\tilde{\mathbf{x}}'_0$, and gives a simulated proof that $\vec{\mathfrak{b}}_{\tilde{\mathbf{x}}_0} = \mathsf{GSCom}(\tilde{\mathbf{x}}'_0; \vec{s}_0)$ and $\vec{\mathfrak{b}}'_{\tilde{\mathbf{x}}_0} = \mathsf{GSCom}(\tilde{\mathbf{x}}_0; \vec{s}'_0)$ are commitments to the same message. That results in additional $2(K+1)^2$ group elements and $(K+1)$ scalars per equation to achieve ZK. (The count is as follows: $(K+1)$ group elements for $\vec{\mathfrak{b}}_{\tilde{\mathbf{x}}_0}$, $(K+1)$ $\mathbb{Z}_p$-elements for $\vec{s}'_0$, $(K+1)^2$ group elements for the commitments to each component of $\vec{s}_0$, and $K(K+1)$ group elements for the NIZK proof of $\vec{\mathfrak{b}}'_{\tilde{\mathbf{x}}_0}$ and $\vec{\mathfrak{b}}'_{\tilde{\mathbf{x}}_0}$ being commitments of the same value (using $(K+1)$ one-sided multi-exponentiation equations).

So, under DLIN we get ZK proofs of size $3N + 21M$ *elements in* $\mathbb{G}$ *and* $3M$ *elements in* $\mathbb{Z}_p$ for a set of such M equations being satisfiable with a witness which has size N.

In the SXDH setting, the equation is no longer one-sided as $\mathbf{T} = \hat{e}(\mathbf{y}, \tilde{\mathbf{g}})$ and $\mathbf{y} \in \mathbb{G}_1$ whereas $\tilde{\mathbf{x}}_i \in \mathbb{G}_2$. However, we could still apply the idea of treating $\mathbf{y}$ as a part of the witness and computing a second commitment $\vec{\mathfrak{a}}'_{\mathbf{y}} = \mathsf{GSCom}(\mathbf{y}; \vec{s}'_{\mathbf{y}})$, and

then showing that the commitments $\vec{\mathbf{a}}_{\mathbf{y}}$ and $\vec{\mathbf{a}}'_{\mathbf{y}}$ are commitments of the same message. According to [71], the WI GS proofs under SXDH are of size $2N + 8M$ for a set of M equations being satisfiable and the witness being of size N. Combining this with the extra group elements we need per equations to achieve ZK, we get proofs of size $2N + 16M$ *group elements and* $2M$ *scalars* when working under SXDH.

2.5 Pairing Randomization Techniques

Abe et al. [3] introduced techniques that randomize elements in a pairing or a pairing product without changing their value in $\mathbb{G}_T$. These useful techniques are used throughout this work.

- **Inner Randomization** $(\mathbf{x}', \mathbf{y}') \leftarrow \mathsf{Rand}(\mathbf{x}, \mathbf{y})$**:** A pairing $\mathbf{A} = \hat{e}(\mathbf{x}, \mathbf{y}) \neq \mathbf{1}$ is randomized as follows. Choose $\gamma \leftarrow \mathbb{Z}_p^*$ and let $(\mathbf{x}', \mathbf{y}') = (\mathbf{x}^{\gamma}, \mathbf{y}^{1/\gamma})$. It then holds that $(\mathbf{x}', \mathbf{y}')$ is distributed uniformly over $\mathbb{G}_1 \times \mathbb{G}_2$ under the condition of $\mathbf{A} = \hat{e}(\mathbf{x}', \mathbf{y}')$. If $\mathbf{A} = \mathbf{1}$, then first flip a coin and pick $\hat{e}(\mathbf{1}, \mathbf{1})$ with probability $1/(2p-1)$. If it is not selected, flip a coin and pick either $\hat{e}(\mathbf{1}, \mathbf{y}')$ or $\hat{e}(\mathbf{x}', \mathbf{1})$ with probability $1/2$, and, respectively, $\mathbf{y}'$ or $\mathbf{x}'$ uniformly from the corresponding group except for $\mathbf{1}$. This way too $(\mathbf{x}', \mathbf{y}')$ is distributed uniformly over $\mathbb{G}_1 \times \mathbb{G}_2$ conditioned on $\mathbf{1} = \hat{e}(\mathbf{x}', \mathbf{y}')$.

- **Sequential Randomization** $\{\mathbf{x}'_i, \mathbf{y}'_i\}_{i=1}^k \leftarrow \mathsf{RandSeq}(\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^k)$**:** A pairing product $\mathbf{A} = \hat{e}(\mathbf{x}_1, \mathbf{y}_1)\, \hat{e}(\mathbf{x}_2, \mathbf{y}_2) \ldots \hat{e}(\mathbf{x}_k, \mathbf{y}_k)$ is randomized into

  $$\mathbf{A} = \hat{e}(\mathbf{x}'_1, \mathbf{y}'_1)\, \hat{e}(\mathbf{x}'_2, \mathbf{y}'_2) \ldots \hat{e}(\mathbf{x}'_k, \mathbf{y}'_k) \text{ as follows :}$$

  Let $(\gamma_1, \ldots, \gamma_{k-1}) \leftarrow \mathbb{Z}_p^{k-1}$. We begin with randomizing the first pairing by using the second pairing as follows. First verify that $\mathbf{y}_1 \neq \mathbf{1}$ and $\mathbf{x}_2 \neq \mathbf{1}$.

If $\mathbf{y}_1 = \mathbf{1}$, replace the first pairing $\hat{e}(\mathbf{x}_1, \mathbf{1})$ with $\hat{e}(\mathbf{1}, \mathbf{y}_1)$ with a new random $\mathbf{y}_1 (\neq \mathbf{1})$. The case of $\mathbf{x}_2 = \mathbf{1}$ is handled in the same manner. Then multiply $\mathbf{1} = \hat{e}(\mathbf{x}_2^{-\gamma_1}, \mathbf{y}_1)\, \hat{e}(\mathbf{x}_2, \mathbf{y}_1^{\gamma_1})$ to both sides of the formula. We thus obtain

$$\mathbf{A} = \hat{e}(\mathbf{x}_1 \mathbf{x}_2^{-\gamma_1}, \mathbf{y}_1)\, \hat{e}(\mathbf{x}_2, \mathbf{y}_1^{\gamma_1} \mathbf{y}_2)\, \hat{e}(\mathbf{x}_3, \mathbf{y}_3) \ldots \hat{e}(\mathbf{x}_k, \mathbf{y}_k). \tag{2.4}$$

Next we randomize the second pairing by using the third one while leaving the first one in tact. As before, if $\mathbf{y}_1^{\gamma_1} \mathbf{y}_2 = \mathbf{1}$ or $\mathbf{x}_3 = \mathbf{1}$, replace its pairing with a random one which also equals $\mathbf{1}$. Then multiply $\mathbf{1} = \hat{e}(\mathbf{x}_3^{-\gamma_2}, \mathbf{y}_1^{\gamma_1} \mathbf{y}_2)\, \hat{e}(\mathbf{x}_3, (\mathbf{y}_1^{\gamma_1} \mathbf{y}_2)^{\gamma_2})$. We thus have

$$\mathbf{A} = \hat{e}(\mathbf{x}_1 \mathbf{x}_2^{-\gamma_1}, \mathbf{y}_1)\, \hat{e}(\mathbf{x}_2 \mathbf{x}_3^{-\gamma_2}, \mathbf{y}_1^{\gamma_1} \mathbf{y}_2)\, \hat{e}(\mathbf{x}_3, (\mathbf{y}_1^{\gamma_1} \mathbf{y}_2)^{\gamma_2} \mathbf{y}_3) \ldots \hat{e}(\mathbf{x}_k, \mathbf{y}_k). \tag{2.5}$$

This continues up to the $(k-1)$-st pairing. When done, the value of the i-th pairing distributes uniformly in $\mathbb{G}_T$ due to the uniform choice of γ_i. The k-th pairing follows the distribution determined by A and preceding $k-1$ pairings. To complete the randomization, every pairing is processed by the inner randomization.

The sequential randomization can be used to extend a product of k pairings a product of arbitrary k', $k' \geq k$, pairings by appending $\hat{e}(\mathbf{1}, \mathbf{1})$ before randomization. By $\{\mathbf{x}'_i, \mathbf{y}'_i\}_{i=1}^{k'} \leftarrow \mathsf{RandExtend}(\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{k})$ for $k' (> k)$ we denote the sequential randomization with extension. Parameters k and k' should be clear from the input and the output.

- **One-side Randomization** $\{\mathbf{x}'_i\}_{i=1}^{k} \leftarrow \mathsf{RandOneSide}(\{\mathbf{g}_i, \mathbf{x}_i\}_{i=1}^{k})$**:** Let $\mathbf{g}_i$ be an element in $\mathbb{G}_1^*$ of symmetric setting Λ_{sym}. A pairing product

  $\mathbf{A} = \hat{e}(\mathbf{g}_1, \mathbf{x}_1)\, \hat{e}(\mathbf{g}_2, \mathbf{x}_2) \ldots \hat{e}(\mathbf{g}_k, \mathbf{x}_k)$ is randomized into

$\mathbf{A} = \hat{e}(\mathbf{g}_1, \mathbf{x}'_1)\,\hat{e}(\mathbf{g}_2, \mathbf{x}'_2)\ldots\hat{e}(\mathbf{g}_k, \mathbf{x}'_k)$ as follows. Let $(\gamma_1, \ldots, \gamma_{k-1}) \leftarrow \mathbb{Z}_p^{k-1}$. First multiply $\mathbf{1} = \hat{e}(\mathbf{g}_1, \mathbf{g}_2^{\gamma_1})\,\hat{e}(\mathbf{g}_2, \mathbf{g}_1^{-\gamma_1})$ to both sides of the formula. We thus obtain

$$\mathbf{A} = \hat{e}(\mathbf{g}_1, \mathbf{x}_1\mathbf{g}_2^{\gamma_1})\,\hat{e}(\mathbf{g}_2, \mathbf{x}_2\mathbf{g}_1^{-\gamma_1})\,\hat{e}(\mathbf{g}_3, \mathbf{x}_3)\ldots\hat{e}(\mathbf{g}_k, \mathbf{x}_k). \tag{2.6}$$

Next multiply $\mathbf{1} = \hat{e}(\mathbf{g}_2, \mathbf{g}_3^{\gamma_2})\,\hat{e}(\mathbf{g}_3, \mathbf{g}_2^{-\gamma_2})$. We thus have

$$\mathbf{A} = \hat{e}(\mathbf{g}_1, \mathbf{x}_1\mathbf{g}_2^{\gamma_1})\,\hat{e}(\mathbf{g}_2, \mathbf{x}_2\mathbf{g}_1^{-\gamma_1}\mathbf{g}_3^{\gamma_2})\,\hat{e}(\mathbf{g}_3, \mathbf{x}_3\mathbf{g}_2^{-\gamma_2})\ldots\hat{e}(\mathbf{g}_k, \mathbf{x}_k). \tag{2.7}$$

This continues until γ_{k-1} and we eventually have $\mathbf{A} = \hat{e}(\mathbf{g}_1, \mathbf{x}'_1)\ldots\hat{e}(\mathbf{g}_k, \mathbf{x}'_k)$. Observe that every $\mathbf{x}'_i$ for $i = 1, \ldots, k-1$ distributes uniformly in $\mathbb{G}$ due to the uniform multiplicative factor $\mathbf{g}_{i+1}^{\gamma_i}$. In the k-th pairing, $\mathbf{x}'_k$ follows the distribution determined by $\mathbf{A}$ and the preceding $k-1$ pairings. Thus $(\mathbf{x}'_1, \ldots, \mathbf{x}'_k)$ is uniform over $\mathbb{G}^k$ under constraint of being evaluated to $\mathbf{A}$.

Note that the algorithms yield uniform elements and thus may include pairings that evaluate to $\mathbf{1}_{\mathbb{G}_T}$. If it is not preferable, it can be avoided by repeating that particular step once again excluding the bad randomness.

Chapter 3

Simulation Extractability

We revisit the notion of simulation extractable NIZK arguments [97, 43, 90, 89, 66], and define a new primitive called *true-simulation extractable* NIZK arguments. Apart from satisfying the three properties described earlier, Definition 10-12, an NIZK argument Π is simulation extractable if there exists a polynomial time algorithm *extractor* $\Pi.\mathsf{Ext}$ which (when given an additional extraction trapdoor $\mathtt{ek}$ associated with the $\mathtt{crs}$) extracts a witness x' from any proof π produced by a malicious prover $\mathcal{A}$, *even* if $\mathcal{A}$ has previously seen some *simulated proofs* for other statements. We make an important distinction between our new definition of *true*-simulation extractability, where all simulated proofs seen by $\mathcal{A}$ are only of *true* statements, and the stronger notion of *any*-simulation extractability, where $\mathcal{A}$ can also see proofs of *false* statements. As we will see, the former notion is often simpler to construct and sufficient in our applications.

3.1 Definitions

We generalize our definition to $\mathtt{f}$-*extractability*, where $\Pi.\mathsf{Ext}$ only needs to output some function $\mathtt{f}(x')$ of a valid witness x'. We further extend this definition to support *labels*, so that the $\Pi.\mathsf{Prove}$, $\Pi.\mathsf{Vrf}$, $\Pi.\mathsf{Sim}$, and $\Pi.\mathsf{Ext}$ algorithms now also take a public label L as input, and the correctness, soundness, and zero-knowlegde properties are adjusted accordingly. If $\Pi = (\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ is an NIZK argument with a simulator $\Pi.\mathsf{Sim}$ and extractor $\Pi.\mathsf{Ext}$, we write $\Pi.\mathsf{Prove}^L, \Pi.\mathsf{Vrf}^L, \Pi.\mathsf{Sim}^L, \Pi.\mathsf{Ext}^L$ to denote proof, verification, simulation, and extraction under label L, respectively.

Definition 18 (Non-Interactive Simulation-Extractable Zero-Knowledge)**.** *Let* $\mathtt{R}$ *be an NP relation with a corresponding language* $\mathfrak{L}_{\mathtt{R}}$ *as defined in Definition 9. A NIZK argument system with labels* Π *is a tuple of algorithms* $(\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ *defined as follows:*

$\Pi.\mathsf{Crs}(1^\lambda)$: *A randomized algorithm generating a common reference string* $\mathtt{crs}$ *and corresponding simulation and extraction keys, respectively* $\mathtt{tk}$ *and* $\mathtt{ek}$*.*

$\Pi.\mathsf{Prove}^L(\mathtt{crs}, y, x)$: *A randomized algorithm that outputs a proof* π *to show the validity of the statement* $\mathtt{R}(x, y') = 1$*, where* $y' = (y, L)$*.*

$\Pi.\mathsf{Vrf}^L(\mathtt{crs}, y, \pi)$: *A verification algorithm that verifies whether proof* π *that* $y' \in \mathfrak{L}^L_{\mathtt{R}}$ *is correct, where* $y' = (y, L)$*. If it is, the algorithm outputs* 1*; otherwise,* 0*.*

$\Pi.\mathsf{Sim}$ is defined in the same way it is in Section 2.1.4; and $\Pi.\mathsf{Ext}^L(\mathtt{crs}, y, \pi, \mathtt{ek})$ takes as input a statement and a valid proof, and using $\mathtt{ek}$ extracts a witness x' for which $(x', (y, L)) \in \mathtt{R}$. For the sake of clarity, we write $\Pi.\mathsf{Prove}$, $\Pi.\mathsf{Vrf}$, $\Pi.\mathsf{Sim}$, and $\Pi.\mathsf{Ext}$ without the $\mathtt{crs}$ parameter when using a single proof system and it is clear from the context.

We now give a formal definition of true-simulation extractability. First, let us define a simulation oracle $\mathcal{O}_{\mathsf{sim}}(\cdot)$. A query to the simulation oracle consists of a witness x and a statement $y' = (y, L)$. The oracle checks if $(x, y') \in \mathtt{R}$. If true, it ignores x and outputs a simulated argument $\Pi.\mathsf{Sim}^{L}(\mathtt{crs}, y, \mathtt{tk})$; and outputs $\perp$ otherwise.

Definition 19 (True-Simulation $\mathtt{f}$-Extractability)**.** *Let* $\mathtt{f}$ *be a fixed efficiently computable function and let* $\Pi = (\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ *be an NIZK argument for a relation* $\mathtt{R}$*, satisfying the completeness, soundness and zero-knowledge properties defined above. We say that* Π *is* true-simulation $\mathtt{f}$-extractable *(*$\mathtt{f}$-tSE*) with labels if for all polynomial time adversaries* $\mathcal{A}$*, the following experiment returns 1 with negligible probability.*

$\mathrm{Exp}_{\textit{SIM-Extractability}}$:

$(\mathtt{crs}, \mathtt{tk}, \mathtt{ek}) \leftarrow \Pi.\mathsf{SimCrs}(1^{\lambda})$

$(y^{\star}, L^{\star}, \pi^{\star}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sim}}(\cdot)}(\mathtt{crs})$

$z^{\star} \leftarrow \Pi.\mathsf{Ext}^{L^{\star}}(y^{\star}, \pi^{\star}, \mathtt{ek})$

output 1 if:

(a) $\mathcal{O}_{\mathsf{sim}}(\cdot)$ was never queried with a statement $(y^{\star}, L^{\star})$,

(b) $\Pi.\mathsf{Vrf}^{L^{\star}}(y^{\star}, \pi^{\star}) = 1$,

(c) and for all x' such that $\mathtt{f}(x') = z^{\star}$ it holds that $\mathtt{R}(x', (y^{\star}, L^{\star})) = 0$;

otherwise, return 0.

In the case when $\mathtt{f}$ *is the identity function, we simply say that* Π *is* true-simulation extractable (tSE).

In other words, the adversary wins if the extractor fails to extract a good value

$z^\star$ which corresponds to at least one valid witness x'. Note that $\mathtt{f}$ is the identity function the third requirement could be rephrased as $\mathtt{R}(z^\star, (y^\star, L^\star)) = 0$.

We give several variations of this new primitive. First, we define *one-time* simulation extractability, in which the adversary $\mathcal{A}$ is only given *a single* query to the simulation oracle $\mathcal{O}_{\mathsf{sim}}(\cdot)$.

Second, we define the notion of *strong* simulation extractability by changing the winning condition so that $\mathcal{A}$ is now required to output a new statement/argument pair instead of a new statement, similarly to the way sEUF-CMA strengthens EUF-CMA the security requirements for digital signatures. More formally, the first winning requirement becomes: the tuple $(y^\star, L^\star, \pi^\star)$ is new, that is, either $(y^\star, L^\star)$ was not the statement of any query to the simulation oracle; or if it was, $\pi^\star$ is different from the proofs returned to $\mathcal{A}$ by $\mathcal{O}_{\mathsf{sim}}(\cdot)$ on those queries. We observe that we can generically construct strong $\mathtt{f}$-tSE NIZK arguments from (standard) $\mathtt{f}$-tSE NIZK arguments if we additionally use a sEUF-CMA-secure one-time signature. In particular, the prover now computes the standard $\mathtt{f}$-tSE argument, signs it, and attaches the verification key $\mathtt{vk}$ to the public label. To verify, we first check that the signature is valid and then verify the $\mathtt{f}$-tSE argument.

Finally, we say that an NIZK argument Π is *any-simultation* $\mathtt{f}$*-extractable (*$\mathtt{f}$-aSE*)* (similar to the notion of simulation-sound extractability of [66]) if the adversary $\mathcal{A}$ instead has access to a modified simulation oracle $\mathcal{O}_{\mathsf{sim}}(\cdot))$ that responds to all simulation queries without checking that $\mathtt{R}(x, (y, L)) = 1$ (and hence might also give simulated arguments of false statements). In this work, we do not make use of this variation, but state it here because as we will see, this notion has been implicitly used in prior works. However, $\mathtt{f}$-aSE is a stronger notion than $\mathtt{f}$-tSE and is *not needed*, as we will show that $\mathtt{f}$-tSE is sufficient in constructing leakage-resilient signatures and CCA encryption.

3.2 Generic Construction of f-tSE

Let $\mathtt{f}$ be any efficiently computable function, and let $\mathtt{R}$ be an NP relation. We show how to construct an $\mathtt{f}$-tSE NIZK argument Ψ from any labeled CCA encryption scheme and (standard) NIZK arguments.

Let $\mathcal{E} = (\mathcal{E}.\mathsf{Key}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be a CCA encryption scheme supporting labels, and let $\Pi = (\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ be an NIZK argument for the relation

$$\hat{\mathtt{R}} = \left\{ ((x, r)\ ,\ (y, \mathfrak{c}, \mathtt{pk}, L)) \ \mid \mathtt{R}(x, y) = 1 \wedge \mathfrak{c} = \mathcal{E}.\mathsf{Enc}^{L||y}(\mathtt{f}(x); r) \right\}, \tag{3.1}$$

where $L||y$ is the label L appended with y. We define $\mathtt{f}$-tSE NIZK argument Ψ (supporting labels) as follows:

- $\Psi.\mathsf{Crs}(1^\lambda)$: Compute $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathcal{E}.\mathsf{Key}(1^\lambda)$, $(\mathtt{crs}, \mathtt{tk}) \leftarrow \Pi.\mathsf{Crs}_\Pi(1^\lambda)$ and output $\mathtt{crs}_\Psi = (\mathtt{crs}, \mathtt{pk})$, $\mathtt{tk}_\Psi = \mathtt{tk}$, $\mathtt{ek}_\Psi = \mathtt{sk}$.

- $\Psi.\mathsf{Prove}^L(y, x; r)$: Let $\mathfrak{c} \leftarrow \mathcal{E}.\mathsf{Enc}^{L'}(\mathtt{f}(x); r)$, where $L' = L||y$, and π be a proof that $(y, \mathfrak{c}, \mathtt{pk}, L) \in \mathcal{L}_{\hat{\mathtt{R}}}$, i.e., $\pi \leftarrow \Pi.\mathsf{Prove}((y, \mathfrak{c}, \mathtt{pk}, L),\ (x, r))$. Output the proof $\psi = (\mathfrak{c}, \pi)$.

- $\Psi.\mathsf{Vrf}^L(y, \psi)$: Parse $\psi = (\mathfrak{c}, \pi)$ and return the result of $\Pi.\mathsf{Vrf}((y, \mathfrak{c}, \mathtt{pk}, L), \pi)$.

Theorem 4. *The above described proof system Ψ is $\mathtt{f}$-tSE NIZK argument for the relation $\mathtt{R}$ if $\mathcal{E}$ is a labeled IND-CCA encryption scheme and Π is an NIZK argument for the relation $\hat{\mathtt{R}}$.*

Proof. Correctness and soundness follow from the correctness and soundness properties of Π. We show that the zero-knowledge and true-simulation extractability hold as well.

Zero-Knowledge. We construct $\Psi.\mathsf{Sim}$ as follows: On input $(y, \mathtt{tk}_\Psi)$ and label L, compute $\mathfrak{c} \leftarrow \mathcal{E}.\mathsf{Enc}^{L'}(0)$ and $\pi \leftarrow \Pi.\mathsf{Sim}(y, \mathfrak{c}, \mathtt{pk}, L)$, and output $\psi = (\mathfrak{c}, \pi)$. By the IND-CCA of $\mathcal{E}$ and the zero-knowledge of Π, we have that the distribution of a simulated argument $\Psi.\mathsf{Sim}^L(y, \mathtt{tk}_\Psi)$ is computationally indistinguishable from a real argument $\Psi.\mathsf{Prove}^L(y, x; r)$.

True-Simulation Extractability. We construct $\Psi.\mathsf{Ext}$ which takes a label label L and input $(y, \psi, \mathtt{ek}_\Psi)$, where $\psi = (\mathfrak{c}, \pi)$ and $\mathtt{ek}_\Psi = \mathtt{sk}$, and outputs $x \leftarrow \mathcal{E}.\mathsf{Dec}^L(\mathfrak{c}, \mathtt{sk})$. To show that it satisfies the security requirements we consider the following sequence of games in which we perform $\mathrm{Exp}_{\textit{SIM-Extractability}}$.

Game 1**:** In this game, the experiment is as described in Definition 19. Let the simulation queries asked by $\mathcal{A}$ are $(x_1, (y_1, L_1)), \ldots, (x_q, (y_q, L_q))$ and let $(y^\star, L^\star, \psi^\star)$, where $\psi^\star = (\mathfrak{c}^\star, \pi^\star)$, be the output of $\mathcal{A}$. Note that the simulator uses x_j only to check $\mathtt{R}(x_j, (y_j, L_j)) = 1$; in other words, the answer $\psi_j = (\mathfrak{c}_j, \pi_j)$ to query $(x_j, (y_j, L_j))$ is a simulated argument and therefore contains an encryption of 0, rather than of $\mathtt{f}(x_j)$, and a simulated argument π_j.

Game 2**:** We change the experiment in the way the simulation oracle answers queries. For the j-th submitted query $(x_j, (y_j, L_j))$, $j = 1, \ldots, q$, if it satisfies the relation, i.e. $\mathtt{R}(x_j, (y_j, L_j)) = 1$, the oracle sets $\mathfrak{c}_j \leftarrow \mathcal{E}.\mathsf{Enc}^{L_j}(\mathtt{f}(x_j))$ and $\pi_j \leftarrow \Pi.\mathsf{Sim}((y_j, \mathfrak{c}_j, \mathtt{pk}, L_j), \mathtt{tk})$, and outputs $\psi_j = (\mathfrak{c}_j, \pi_j)$; otherwise it returns $\perp$. To see that Game 1 and Game 2 are computationally indistinguishable we perform a sequence of hybrid experiments. The i-th hybrid experiment, $i = 0, \ldots, q$, is defined like in Game 1 with the difference that the j-th query, $j \leq i$, is answered like in Game 2. Clearly, this sequence of experiments transform the experiment from Game 1 into the one from Game 2.

As each two consecutive hybrid experiments are computationally indistinguishable by the IND-CCA of the encryption $\mathcal{E}$, so are Game 1 and Game 2. If the former does not hold and an adversary $\mathcal{A}$ can distinguish between two hybrids, we could construct an adversary $\mathcal{B}$ that given $\mathtt{pk}$ runs $(\mathtt{crs}, \mathtt{tk}) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$, computes all $\mathfrak{c}_j$ for $j \neq i$ accordingly, sets $\mathfrak{c}_i$ to be its challenge ciphertext (which is an encryption of 0 or $\mathtt{f}(x_i)$ under label $L'_i = L_i||y_i$), and simulates all the proofs for $\mathcal{A}$. Notice that we need to rely on the stronger notion of IND-CCA, instead of the weaker IND-CPA, since $\mathcal{B}$ needs one decryption query in order to extract the plaintext $z^\star$ of $\mathfrak{c}^\star$ and check the $\mathtt{f}$-tSE winning condition. Moreover, note that $\mathcal{B}$ is allowed to query its decryption oracle with $\mathfrak{c}^\star$ and the corresponding label because the fact that $(y^\star, L^\star)$ was never queried by $\mathcal{A}$ guarantees $L^\star||y^\star \neq L'_i$.

Game 3**:** We change the simulator oracle so that the queries $(x_j, (y_j, L_j))$, for $j = 1 \ldots q$, are answered as follows: if $R(y_j, x_j) = 0$, it returns $\perp$ as before; but if $R(x_j, (y_j, L_j)) = 1$, it computes ciphertext $\mathfrak{c}_j \leftarrow \mathcal{E}.\mathsf{Enc}^{L'_j}(\mathtt{f}(x_j))$ and proof $\pi_j \leftarrow \Pi.\mathsf{Prove}((y_j, \mathfrak{c}_j, \mathtt{pk}, L_j), x_j)$, and outputs $\psi_j = (\mathfrak{c}_j, \pi_j)$. Games 2 and 3 are indistinguishable by the zero-knowledge of Π.

Notice that if adversary $\mathcal{A}$ wins in the experiment from Game 3, then it must be the case that $\Psi.\mathsf{Vrf}^{L^\star}(y^*, \psi^*) = 1$ and the extracted value $z^\star$ has no corresponding $x^\star$ for which $\mathtt{f}(x^\star) = z^\star$ and $R(x^\star, y^\star) = 1$. However, that would contradict the soundness of Π. Hence, any polynomial time adversary can have only negligible probability of breaking the true-simulation extractability property if $\mathcal{E}$ is IND-CCA and Π is an NIZK argument.

□

3.3 Comparison with Previous Works and the Naor-Yung Paradigm

We view tSE NIZK proofs as having two major advantages.

First, it makes the generic constructions of (leakage resilient) encryption and signature somewhat more intuitive, both for proving and understanding. For example, the traditional "double-encryption" paradigm of Naor-Yung [85] for designing IND-CCA schemes from semantically secure (IND-CPA) schemes, also used by [83] in the context of key leakage, can be stated as "CPA-encrypting message m under two keys and proving plaintext equality". Using our more general "simulation-extractability view", it is now stated as "CPA-encrypting m and proving that one knows the plaintext". We believe that the latter view is not only more general, but also more intuitive as a way of explaining "CPA-to-CCA" transformation. It also follows the original intuition of Rackoff and Simon [93], who combine IND-CPA encryption with NIZK proofs of knowledge to achieve IND-CCA encryption, but in the model where the sender also has a secret key.

A similar discussion is true for our leakage-resilient signature constructions when compared with that of [75]. Furthermore, using our "simulation-extractability view" of signatures, i.e., "a signature on m is a tSE proof of a preimage of a (leakage-resilient) one-way function with a label m", we also encompass the structure-preserving signature scheme of [66]. So, all one would need to instantiate this generic construction will be efficient building blocks.

The second point is about efficiency. Clearly, `f`-tSE NIZK proofs can be constructed from ss-NIZK proofs and CPA encryption scheme. However, as we showed in the last section, there is a generic way to build tSE-NIZKs which *avoids using (more expensive) ss-NIZK proofs*. Instead, our method uses regular NIZK proofs and *any*

CCA-secure encryption scheme. Given the current state-of-the-art NIZK and CCA schemes, the combination "CCA + NIZK" appears to be more efficient in practice than the combination "CPA + ss-NIZK".[1]

Moreover, in the case of leakage-resilient signatures, this more intuitive view allowed us to construct efficient underlying primitives, prior to which there was *no efficient way to realize* the construction of [75] regardless of the flavor of NIZK. As a result, we were able to provide a general framework for building leakage-flexible signature and CCA-encryption schemes, eventually allowing us to efficiently instantiate our schemes (by avoiding using ss-NIZKs). We summarize our results in Section 3.4 and Section 4.3.3.

Finally, note that our generic construction of tSE NIZK did not require the full power of CCA encryption. The reduction algorithm needed only one decryption query after seeing the challenge ciphertext. Although there are weakened notions of IND-CCA such as RCCA [42] which would suffice for our purposes, as observed in [66], we do not benefit from that as current constructions do not provide better efficiency.

3.4 Application: Efficient Leakage-Resilient Encryption

In this section, we give a generic construction of leakage-resilient IND-CCA encryption from leakage-resilient IND-CPA encryption and strong `f`-tSE NIZK arguments. Then, we instantiate it under the K-linear assumption in the Λ_{sxdh} and Λ_{sym} settings.

[1] Indirectly, the same realization was made by Groth [67] and Camenisch et al. [31] in different concrete contexts. While using ss-NIZK is an alternative which degrades efficiency "only" by a (small) constant factor, such constants are not be ignored for efficiency.

3.4.1 Generic Construction

Let $\mathsf{LR}\text{-}\mathcal{E}^\dagger = (\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Key}^\dagger, \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger, \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Dec}^\dagger)$ be an ℓ-LR-CPA encryption scheme and let $\Pi = (\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ be a strong one-time $\mathtt{f}$-tSE NIZK argument for the relation

$$\mathtt{R} = \{\big((m, r),\ (\mathtt{pk}^\dagger, \mathtt{c}^\dagger)\big) \ \mid\ \mathtt{c}^\dagger = \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(\mathtt{pk}^\dagger, m; r)\},$$

where $\mathtt{f}(m, r) = m$, i.e., the extractor only needs to extract the message m, but not the randomness r used to encrypt. We show how to construct an ℓ-LR-CCA encryption scheme $\mathsf{LR}\text{-}\mathcal{E}$ out of $\mathsf{LR}\text{-}\mathcal{E}^\dagger$ and Π. Define $\mathsf{LR}\text{-}\mathcal{E}$ as follows:

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Key}(1^\lambda)$: Let $(\mathtt{pk}^\dagger, \mathtt{sk}^\dagger) \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Key}^\dagger(1^\lambda)$ and $(\mathtt{crs}, \mathtt{tk}, \mathtt{ek}) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$. Output $\mathtt{pk} = (\mathtt{pk}^\dagger, \mathtt{crs})$ and $\mathtt{sk} = \mathtt{sk}^\dagger$.

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}(\mathtt{pk}, m, r)$: Encrypt m with $\mathsf{LR}\text{-}\mathcal{E}^\dagger$, i.e., $\mathtt{c}^\dagger \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(m, r)$ and produce an argument for the statement $(\mathtt{pk}^\dagger, \mathtt{c}^\dagger)$, i.e. $\pi \leftarrow \Pi.\mathsf{Prove}((\mathtt{pk}^\dagger, \mathtt{c}^\dagger),\ (m, r))$. Output $\mathtt{c} = (\mathtt{c}^\dagger, \pi)$.

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Dec}(\mathtt{sk}, \mathtt{c})$: Parse $\mathtt{c} = (\mathtt{c}^\star, \pi)$. If the argument π verifies, return the output of $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Dec}^\dagger(\mathtt{sk}^\dagger, \mathtt{c}^\dagger)$; otherwise, output $\perp$.

Theorem 5. *If* $\mathsf{LR}\text{-}\mathcal{E}^\dagger$ *is* ℓ*-LR-CPA and* Π *is a strong one-time* $\mathtt{f}$-tSE *NIZK argument for the relation* $\mathtt{R}$ *defined above, where for any witness* (m, r) *the function* $\mathtt{f}$ *returns* $\mathtt{f}(m, r) = m$*, then* $\mathsf{LR}\text{-}\mathcal{E}$ *is* ℓ*-LR-CCA secure.*

Before proving the theorem, we note that if Π supports labels, then we can naturally extend our construction to yield ℓ-LR-CCA encryption *with labels* by passing the encryption labels to Π (and using them to verify the proofs).

Proof. To prove the theorem, we consider a series of games and their corresponding experiments, all of which are variants of the experiment from Definition 15. In all of

them the adversary is given a correctly generated public key $\mathtt{pk} = (\mathtt{pk}^\dagger, \mathtt{crs})$, and any leakage queries are answered using the correctly generated secret key $\mathtt{sk}$. The games differ in the way the challenge ciphertext $\mathfrak{c}^\star$ is generated and how the decryption oracle $\mathcal{O}_{\mathsf{dec}}(\cdot)$ answers queries $\mathfrak{c} = (\mathfrak{c}^\dagger, \pi)$.

Game 1**:** This is the original experiment of ℓ-LR-CCA encryption. The challenge ciphertext and the decryption queries are, respectively, generated and answered correctly. That is, $\mathfrak{c}^\star = (\left(\mathfrak{c}^\dagger\right)^\star, \pi^\star)$, where

$$\left(\mathfrak{c}^\dagger\right)^\star \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(m_b, r) \text{ and } \pi^\star \leftarrow \Pi.\mathsf{Prove}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ (m_b, r));$$

and decryption queries $\mathfrak{c}$ are answered with the output of $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Dec}(\mathtt{sk}, \mathfrak{c})$.

Game 2**:** In this game, we modify the experiment so that the $\mathtt{tk}$ is remembered and all arguments π are simulated, i.e., computed using $\Pi.\mathsf{Sim}(\cdot, \mathtt{tk})$. Then, the challenge ciphertext is computed as follows:

$$\left(\mathfrak{c}^\dagger\right)^\star \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(m_b, r) \text{ and } \pi^\star \leftarrow \Pi.\mathsf{Sim}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \mathtt{tk});$$

and decryption queries $\mathfrak{c}$ are answered with $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Dec}(\mathtt{sk}, \mathfrak{c})$. Note that the adversary sees only one simulated argument and it is of a true statement.

Game 1 and Game 2 are indistinguishable by the zero-knowledge property of the argument system Π.

Game 3**:** The experiment is further modified so that $\mathtt{ek}$ is stored, and any decryption query $\mathfrak{c} = (\mathfrak{c}^\dagger, \pi)$ is answered with the output of $\Pi.\mathsf{Ext}(\cdot, \cdot, \mathtt{ek})$ which returns a message m according to the definition of $\mathtt{f}$ above. The challenge ciphertext is computed as before. That is, $\mathfrak{c}^\star = (\left(\mathfrak{c}^\dagger\right)^\star, \pi^\star)$, where

$$\left(\mathfrak{c}^\dagger\right)^\star \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(m_b, r) \text{ and } \pi^\star \leftarrow \Pi.\mathsf{Sim}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \mathtt{tk});$$

and decryption queries $\mathfrak{c}$ are answered with $\Pi.\mathsf{Ext}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \pi,\ \mathtt{ek})$.

Games 2 and Game 3 are indistinguishable by the strong one-time tSE $\mathtt{f}$-extractability of Π. That is, the adversary only gets a single simulated argument $\pi^\star$ of a true statement, and, therefore, cannot produce any new statement-argument pair $((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \pi)$ for which $\Pi.\mathsf{Vrf}$ verifies but $\Pi.\mathsf{Ext}$ fails to extract a correct message.

Game 4**:** In this game, the challenge ciphertext $\mathfrak{c}^\star$ is computed by encrypting 0 (or a random message from the message space). This way, we have: $\mathfrak{c}^\star = (\left(\mathfrak{c}^\dagger\right)^\star, \pi^\star)$, where

$$\left(\mathfrak{c}^\dagger\right)^\star \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}^\dagger(0, r) \text{ and } \pi^\star \leftarrow \Pi.\mathsf{Sim}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \mathtt{tk});$$

and decryption queries $\mathfrak{c}$ are answered with $\Pi.\mathsf{Ext}((\mathtt{pk}^\dagger, \mathfrak{c}^\dagger),\ \pi,\ \mathtt{ek})$.

Games 3 and 4 are indistinguishable by the ℓ-LR-CPA security of $\mathsf{LR}\text{-}\mathcal{E}^\dagger$. Recall that leakage queries are always answered using $\mathtt{sk}$ and so we need to rely on leakage-resilience here. However, IND-CPA security now suffices since the decryption secret-key $\mathtt{sk}$ is never used otherwise in Games 3 or Game 4.

Notice that the experiment in Game 4 chooses a bit b at random but it is completely independent from the challenge ciphertext, hence from the output bit $b^\star$. Therefore, it returns 1 with probability $\frac{1}{2}$. As all the games are computationally indistinguishable, any polynomial time adversary $\mathcal{A}$ wins the experiment from the security definition of ℓ-LR-CCA with probability $\frac{1}{2} + \mathsf{negl}(\lambda)$. □

3.4.2 Instantiation

In order to use the construction from the last section, we need a $(1-\epsilon)|\mathtt{sk}|$-LR-CPA encryption scheme $\mathcal{E}_1 = (\mathcal{E}_1.\mathsf{Key}, \mathcal{E}_1.\mathsf{Enc}, \mathcal{E}_1.\mathsf{Dec})$ and a strong $\mathtt{f}$-tSE NIZK argument

Reference	Attack	Model	Leakage	Efficient?
[5, 83]	*CPA*	Standard	1	Yes
[83]	CCA	Standard	*1/6*	Yes
[83]	CCA	Standard	1	*No*
This Work	CCA	Standard	1	Yes

Table 3.1: Comparison of previous leakage-resilient encryption schemes with our work.

(see Section 3.1), which we can construct from IND-CCA encryption scheme supporting labels $\mathcal{E}_2 = (\mathcal{E}_2.\mathsf{Key}, \mathcal{E}_2.\mathsf{Enc}, \mathcal{E}_2.\mathsf{Dec})$, a strong one-time signature scheme, and an NIZK argument Π for the relation

$$\mathtt{R}_{\mathsf{eq}} = \{((w_1, w_2), (\mathfrak{c}_1, \mathfrak{c}_2, L)) \mid \exists m \text{ s.t. } \mathfrak{c}_1 \leftarrow \mathcal{E}_1.\mathsf{Enc}(m; w_1) \wedge \mathfrak{c}_2 \leftarrow \mathcal{E}_2.\mathsf{Enc}^L(m; w_2)\}.$$

The same technique was used in [31] to construct an efficient IND-CCA encryption scheme with key-dependent message (KDM) security from a IND-CPA version of the scheme. We use the same technique in the leakage-setting, to achieve leakage-resilient IND-CCA encryption from leakage-resilient IND-CPA encryption.

LR-CPA Encryption.

We use a $(1-\epsilon)|sk|$-leakage resilient CPA-secure encryption scheme based on the K-linear assumption. Similar versions of this scheme appear in [83] and [31] (based on the KDM scheme of [26]), but we use the more efficient variant due to [51] which has shorter public key and ciphertext by a factor of $\log p$.

Let $\mathbb{G}$ be a group of primer order p, and let J be an integer. We define the encryption scheme $\mathsf{LR}\text{-}\mathcal{E}$ as follows:

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Key}(1^\lambda)$: Choose $\mathbf{f}_{0,1}, \ldots, \mathbf{f}_{0,J}, \mathbf{f}_1, \ldots \mathbf{f}_K \leftarrow \mathbb{G}$ and $\vec{x} \leftarrow \mathbb{Z}_p^{K+J}$. Define vec-

tors $\vec{\mathbf{f}}_1, \ldots, \vec{\mathbf{f}}_K \in \mathbb{G}^{K+J}$ as follows:

$$\begin{aligned}
\vec{\mathbf{f}}_1 &= (\mathbf{f}_{0,1}, \ldots, \mathbf{f}_{0,J}, \mathbf{f}_1, \mathbf{1}, \ldots, \mathbf{1}) \\
\vec{\mathbf{f}}_2 &= (\mathbf{f}_{0,1}, \ldots, \mathbf{f}_{0,J}, \mathbf{1}, \mathbf{f}_2, \ldots, \mathbf{1}) \\
&\vdots \\
\vec{\mathbf{f}}_K &= (\mathbf{f}_{0,1}, \ldots, \mathbf{f}_{0,J}, \mathbf{1}, \mathbf{1}, \ldots, \mathbf{f}_K)
\end{aligned}$$

Let $\mathbf{h}_i = \langle \vec{\mathbf{f}}_i, \vec{x} \rangle$, for $i = 1, \ldots, K$, and let $\vec{\mathbf{h}} = (\mathbf{h}_1, \ldots, \mathbf{h}_K)$.

Output $\mathtt{pk} = (\{\vec{\mathbf{f}}_i\}_{i=1}^K, \vec{\mathbf{h}})$ and $\mathtt{sk} = \vec{x}$.

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}(\mathtt{pk}, \mathbf{m})$: Choose $\vec{s} \leftarrow \mathbb{Z}_p^K$, and compute $\vec{\mathbf{f}} = \prod_{i=1}^K \vec{\mathbf{f}}_i^{s_i}$ and the one-time pad of the message $\mathbf{a} = \mathbf{m} \cdot \langle \vec{\mathbf{h}}, \vec{s} \rangle$. Output $\mathfrak{c} = (\vec{\mathbf{f}}, \mathbf{a})$.

- $\mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc2}(\mathtt{sk}, \mathfrak{c})$: Parse $\mathfrak{c}$ as $(\vec{\mathbf{f}}, \mathbf{a})$ and output $\mathbf{m} \leftarrow \mathbf{a} / \langle \vec{\mathbf{f}}, \vec{x} \rangle$.

Theorem 6. *[51] For any $\epsilon > 0$, if $J \geq \frac{1}{\epsilon}(K + \lambda/\log(p) + 1)$, then the above encryption scheme is ℓ-LR-CPA, where $\ell = (1 - \epsilon)|sk|$, if the K-linear assumption holds.*

For the instantiation, we use $\mathsf{LR}\text{-}\mathcal{E}$ as $\mathcal{E}_1$ working in the group $\mathbb{G}_2$, with all elements of the public key $\mathtt{pk}_1 = (\{\vec{\mathbf{f}}_i\}_{i=1}^K, \vec{\mathbf{h}})$ being in $\mathbb{G}_2$. We encrypt $\mathbf{m} \in \mathbb{G}_2$ under randomness $\vec{s} \in \mathbb{Z}_p^K$ as the ciphertext $\mathfrak{c}_1 = (\mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,(J+K+1)})$ which is computed as:

$$\mathfrak{c}_1 \leftarrow \mathsf{LR}\text{-}\mathcal{E}.\mathsf{Enc}(\mathtt{pk}_1, \mathbf{m}; \vec{s}) = (\mathbf{f}_{0,1}^{\sum_{i=1}^K s_i}, \ldots, \mathbf{f}_{0,J}^{\sum_{i=1}^K s_i}, \mathbf{f}_1^{s_1} \ldots, \mathbf{f}_K^{s_K}, \ \mathbf{m} \prod_{i=1}^K \mathbf{h}_i^{s_i})$$

The size of the ciphertext is $J + K + 1$.

IND-CCA Encryption.

We review the K-linear Cramer-Shoup encryption scheme from [99], modified to support labels as in [31]. We extend it to be multi-message randomness-reuse encryption

using the paradigm of [14], which we further optimize by reusing the validity ciphertext element.

Let $\mathbb{G}$ be a group of prime order p, $H : \{0,1\} \to \mathbb{Z}_p$ be a collision resistant hash function, which defines the label space as $\{0,1\}^*$, and n be a positive integer.

- CS-$\mathcal{E}$.Key(1^λ):

  1. Choose $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K \xleftarrow{\$} \mathbb{G}$ and choose $\vec{x}_1, \ldots, \vec{x}_n, \vec{y}, \vec{z} \xleftarrow{\$} \mathbb{Z}_p^{K+1}$.
  2. Define vectors $\vec{\mathbf{g}}_1, \ldots, \vec{\mathbf{g}}_K \in \mathbb{G}^{K+1}$ as follows:

     $$\vec{\mathbf{g}}_1 = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{1}, \ldots, \mathbf{1}), \vec{\mathbf{g}}_2 = (\mathbf{g}_0, \mathbf{1}, \mathbf{g}_2, \mathbf{1}, \ldots, \mathbf{1}), \ldots, \vec{\mathbf{g}}_K = (\mathbf{g}_0, \mathbf{1}, \ldots, \mathbf{1}, \mathbf{g}_K)$$

  3. For $i = 1, \ldots, K$ and $j = 1, \ldots, n$:
     let $\mathbf{d}_{ji} \leftarrow \langle \vec{\mathbf{g}}_i, \vec{x}_j \rangle$, $\mathbf{e}_i \leftarrow \langle \vec{\mathbf{g}}_i, \vec{y} \rangle$, $\mathbf{f}_i \leftarrow \langle \vec{\mathbf{g}}_i, \vec{z} \rangle$
  4. Output a public key $\mathtt{pk} = (\{\mathbf{g}_i\}_{i=0}^K, \{\mathbf{d}_{ji}\}_{i=1,j=1}^{K,n}, \{\mathbf{e}_i\}_{i=1}^K, \{\mathbf{f}_i\}_{i=1}^K)$ and a secret key $\mathtt{sk} = (\vec{x}_1, \ldots, \vec{x}_n, \vec{y}, \vec{z})$.

- CS-$\mathcal{E}$.Enc$^L(\mathtt{pk}, (\mathbf{m}_1, \ldots, \mathbf{m}_n))$:

  Define $\vec{\mathbf{g}}_i$ as in CS-$\mathcal{E}$.Key, For $i = 1, \ldots, K$, and pick $\vec{r} \leftarrow \mathbb{Z}_p^K$ Output

  $$\begin{aligned} \mathfrak{c} &= (\vec{\mathbf{g}}, \mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{b}) \\ &= \left(\prod_{i=1}^K \vec{\mathbf{g}}_i^{r_i},\ \mathbf{m}_1 \cdot \prod_{i=1}^K \mathbf{d}_{1i}^{r_i},\ \ldots,\ \mathbf{m}_n \cdot \prod_{i=1}^K \mathbf{d}_{n,i}^{r_i},\ \prod_{i=1}^K (\mathbf{e}_i \mathbf{f}_i^t)^{r_i} \right), \end{aligned}$$

  where $t = H(\vec{\mathbf{g}}, \mathbf{a}_1, \ldots, \mathbf{a}_n, L)$

- CS-$\mathcal{E}$.Dec$^L(\mathtt{sk}, \mathfrak{c})$: Parse $\mathfrak{c} = (\vec{\mathbf{g}}, \mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{b})$ and let $t \leftarrow H(\vec{\mathbf{g}}, \mathbf{a}_1, \ldots, \mathbf{a}_n, L)$. If $\mathbf{b} \neq \langle \vec{\mathbf{g}}\ ,\ \vec{y} + t\vec{z} \rangle$, output $\perp$. Else, for $j = 1, \ldots, n$, let $\mathbf{m}_j \leftarrow \mathbf{a}_j / \langle \vec{\mathbf{g}}, \vec{x}_j \rangle$, and outputs $\vec{\mathbf{m}} = (\mathbf{m}_1, \ldots, \mathbf{m}_n)$.

Theorem 7. *[31] If the K-*linear *assumptions holds for $\mathbb{G}$, then the above encryption scheme is IND-CCA.*

For the instantiation, we use the K-linear encryption scheme in $\mathbb{G}_2$ as $\mathcal{E}_2$. For a public key $\mathtt{pk}_2 \leftarrow \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Key}(1^\lambda)$, we encrypt a message $\mathbf{m} \in \mathbb{G}_2$ under randomness $\vec{r} \in \mathbb{Z}_p^K$ and label L as $\mathfrak{c}_2 = (\mathbf{c}_{2,1}, \ldots, \mathbf{c}_{2,(K+3)}) \leftarrow \mathcal{E}_2.\mathsf{Enc}^L(\mathbf{m}; \vec{r})$. The size of the ciphertext is $K + 3$ group elements.

NIZK Argument System.

We use the NIZK argument system described in Section 2.4. Let $\mathfrak{c}_1, \mathfrak{c}_2$ be as described above. To prove that there exists $(\mathbf{m}, w_1, w_2)$ such that $((w_1, w_2), (\mathfrak{c}_1, \mathfrak{c}_2, L)) \in \mathtt{R}_{\mathsf{eq}}$, we use a system of one-sided multi-exponentiation equations.

$$\begin{aligned}
\mathbf{c}_{2,1} &= \mathbf{g}_0^{\sum_{i=1}^K r_i}; \\
\mathbf{c}_{2,(i+1)} &= \mathbf{g}_i^{r_i}, \quad \text{for } i = 1, \ldots, K; \\
\mathbf{c}_{2,(K+3)} &= \textstyle\prod_{i=1}^K (\mathbf{e}_i \mathbf{f}_i^t)^{r_i}; \\
\mathbf{c}_{1,(J+K+1)}/\mathbf{c}_{2,(K+2)} &= \textstyle\prod_{i=1}^K \mathbf{h}_i^{s_i} (\mathbf{d}_i^{-1})^{r_i}; \\
\mathbf{c}_{1,j} &= \mathbf{f}_{0,j}^{\sum_{i=1}^K s_i}, \quad \text{for } j = 1, \ldots, J; \\
\mathbf{c}_{1,(J+i)} &= \mathbf{f}_i^{s_i}, \quad \text{for } i = 1, \ldots, K.
\end{aligned}$$

This corresponds to a system of $J + 2K + 3$ equations with witness $(r_1, \ldots, r_K, s_1, \ldots, s_K)$. The Groth-Sahai proof system yields an argument consisting of $2K$ commitments, each of size $(K + 1)$ group elements, and K proof elements for each equation. So, the total argument size is $K(J + 4K + 5)$.

Based on SXDH: If we work in Λ_{sxdh} setting, hence DDH holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$, $K = 1$ and the size of the argument is $J + 9$ proof elements.

Based on DLIN: If we work in Λ_{sym} and assume DLIN to hold for $\mathbb{G}$, $K = 2$ and the size of the argument is $2J + 26$ group elements.

One-Time Signature.

We use the strong one-time signature of [66]. Let $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ be a collision-resistant hash function.

- $\Sigma.\mathsf{Key}(1^\lambda, \Lambda)$: Output $\mathtt{vk} = (\tilde{\mathbf{f}}, \tilde{\mathbf{h}}, \tilde{\mathbf{a}})$ and $\mathtt{sk} = (\gamma_1, \gamma_2, \delta_1, \delta_2)$, where:

$$\gamma_1, \gamma_2, \delta_1, \delta_2 \leftarrow \mathbb{Z}_p^* \ , \ \tilde{\mathbf{f}} \leftarrow \tilde{\mathbf{g}}^{\gamma_1}, \tilde{\mathbf{h}} \leftarrow \tilde{\mathbf{g}}^{\gamma_2} \ , \ \tilde{\mathbf{a}} \leftarrow \tilde{\mathbf{f}}^{\delta_1}\tilde{\mathbf{h}}^{\delta_2}$$

  .

- $\Sigma.\mathsf{Sign}(\mathtt{sk}, m)$: Choose $r \leftarrow \mathbb{Z}_p$, compute

$$s \leftarrow \left(\frac{(\gamma_1(\delta_1 - r) + \gamma_2\delta_2 - H_2(m))}{\gamma_2}\right),$$

  and output $\sigma = (r, s)$, where.

- $\Sigma.\mathsf{Vrf}(\mathtt{vk}, m, \sigma)$: Parse $\sigma = (r, s)$ and verify that $\tilde{\mathbf{a}} = \tilde{\mathbf{g}}^{H_2(m)}\tilde{\mathbf{f}}^r\tilde{\mathbf{h}}^s$.

The size of the one-time signature is 2 elements in $\mathbb{Z}_p$.

Putting Everything Together. Combining both ciphertexts, together with the NIZK argument, and the one-time signature, we have that the size of the ciphertext is $2J + 18$ group elements and 2 elements in $\mathbb{Z}_p$ in the SXDH case, and $3J + 37$ group elements and 2 elements in $\mathbb{Z}_p$ in the DLIN case. From Theorem 6 we need $J \geq \frac{1}{\epsilon}(K + \lambda/\log(p) + 1)$. This gives us the following ciphertext size:

Based on SXDH: The size of the ciphertext is $(2/\epsilon)(2 + \lambda/\log p) + 18$ group elements and 2 elements in $\mathbb{Z}_p$.

Based on DLIN: The size of the ciphertext is $(3/\epsilon)(3 + \lambda/\log p) + 37$ group elements and 2 elements in $\mathbb{Z}_p$.

Chapter 4

Structure-Preserving Commitments

4.1 Constructions

This section presents several homomorphic trapdoor commitment schemes in bilinear settings. All of them have specific properties which find different applications later on. We note that all the schemes in this section can also work as a chameleon hash. Namely, it is possible to equivocate any commitment generated by TC.Com rather than the ones simulated by TC.Sim. Indeed, we integrate TC1 as a chameleon hash in the construction of SSIG in Section 5.3.

By $(\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{D})$ we denote spaces for commitment-keys, messages, commitments, and decommitments. The commitment-key refers to elements not included in the group description Λ. Table 4.1 shows a summary of the schemes in their space parameters and performance in verifying the correct opening. For comparison, we list schemes from [68] and [44] which are the only homomorphic trapdoor commitment schemes we aware in the literature whose messages are group elements and the veri-

Scheme	Λ	$\mathcal{K}$	$\mathcal{M}$	$\mathcal{C}$	$\mathcal{D}$	#pairings	#PPEs	assump.
TC1	any	$2k + 2^{[1]}$	$k^{[2]}$	$2^{[T]}$	$2^{[2]}$	$2k + 2$	2	SDP
TC2	$\Lambda_{\mathsf{xdh,sxdh}}$	$k + 1^{[1]}$	$k^{[2]}$	$1^{[T]}$	$1^{[2]}$	$k + 1$	1	DBP
Gro09[68]	any	$2k + 4^{[1]}$	$k^{[2]}$	$2^{[T]}$	$2^{[2]}$	$2k + 4$	2	STP
TC3	Λ_{sym}	$2k + 2^{[1]}$	$k^{[1]}$	$2k + 2^{[1]}$	$2^{[1]}$	2k+ 2	2	SDP
CLY09[44]	Λ_{sym}	$5^{[1]}$	$1^{[1]}$	$3^{[1]}$	$3^{[1]}$	9	3	DLIN
TC4	any	$2^{[2]}$	$1^{[p]}$	$1^{[2]}$	$1^{[1]}$	2	1	XDHI

Table 4.1: Summary of homomorphic trapdoor commitments. Columns from $\mathcal{K}$ to $\mathcal{D}$ count the number of elements and indicating the groups they belong to ([1], [2], [T], and [p] respectively for $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, and $\mathbb{Z}_p$). #pairings and #PPEs count the number of pairings and pairing product equations in the verification predicate. On top are the multi-message schemes committing to k group elements at once; in the middle are the schemes *not* using any group element in $\mathbb{G}_T$; and at the bottom is the efficient scheme when the message is in $\mathbb{Z}_p$ and the other components are in $\mathbb{G}_1$ and $\mathbb{G}_2$.

fication is done by checking pairing product equations.

TC3 and [44] are fully GS-compatible schemes whose components are all in the base groups. In particular, TC3 is the first such commitment scheme that commits to a message composed of k group elements at the same time; its commitment has $2k + 2$ group elements. In contrast, [44] needs to commit each element separately, thus commitment is of size $3k$. It is an interesting open problem to construct a constant-size commitment scheme while being compatible with GS-proofs.

Scheme TC2 is independently found in [69], the updated version of [68], and is included in [1].

For multi-message commitment schemes, TC1, TC2, TC3, let $\vec{\mathbf{m}} = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\} \in {\mathbb{G}_2}^k$ be a message. For single-message commitment scheme, TC4, let m be an element

of $\mathbb{Z}_p$. In the following description, we assume that Λ is given to all algorithms implicitly.

4.1.1 Scheme TC1

TC1.Key(1^λ): Choose random generators $\mathbf{g}_r, \mathbf{h}_u$ from $\mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose γ_i and δ_i from $\mathbb{Z}_p^*$ and compute $\mathbf{g}_i = \mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_u^{\delta_i}$. Output commitment-key $\mathtt{ck} = (\mathbf{g}_r, \mathbf{h}_u, \mathbf{g}_1, \mathbf{h}_1, \ldots, \mathbf{g}_k, \mathbf{h}_k)$ and trapdoor $\mathtt{tk} = (\gamma_1, \delta_1 \ldots, \gamma_k, \delta_k)$.

TC1.Com($\mathtt{ck}, \vec{\mathbf{m}}$): Choose $\mathbf{r}$ and $\mathbf{u}$ randomly from $\mathbb{G}_2$, and compute

$$\mathbf{C}_1 = \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \quad \text{and} \quad \mathbf{C}_2 = \hat{e}(\mathbf{h}_u, \mathbf{u}) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i). \tag{4.1}$$

Output commitment $\mathfrak{c} = (\mathbf{C}_1, \mathbf{C}_2)$ and decommitment $\mathfrak{d} = (\mathbf{r}, \mathbf{u})$.

TC1.Vrf($\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathfrak{d}$): Parse c into $(\mathbf{C}_1, \mathbf{C}_2)$ and $\mathfrak{d}$ into $(\mathbf{r}, \mathbf{u})$. Output 1 if (4.1) holds. Output 0, otherwise.

TC1.Sim($\mathtt{ck}, \mathtt{tk}$): Choose $\mathbf{r}$ and $\mathbf{u}$ randomly from $\mathbb{G}_2$ and compute $\mathbf{C}_1 = \hat{e}(\mathbf{g}_r, \mathbf{r})$ and $\mathbf{C}_2 = \hat{e}(\mathbf{h}_u, \mathbf{u})$. Output commitment $\mathfrak{c} = (\mathbf{C}_1, \mathbf{C}_2)$ and equivocation-key $\mathtt{ek} = (\mathtt{tk}, \mathbf{r}, \mathbf{u})$.

TC1.Open($\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathtt{ek}$): Parse $\mathtt{ek}$ into $(\mathtt{tk}, \mathbf{r}, \mathbf{u})$ and $\mathtt{tk}$ into $(\gamma_1, \delta_1 \ldots, \gamma_k, \delta_k)$. Then compute $\mathbf{r}' = \mathbf{r} \cdot \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}$, and $\mathbf{u}' = \mathbf{u} \cdot \prod_{i=1}^{k} \mathbf{m}_i^{-\delta_i}$. Then output decommitment $\mathfrak{d} = (\mathbf{r}', \mathbf{u}')$.

The above scheme shares many similarities with that of Groth [68], but the security is based on a different computational assumption, i.e., SDP. It should be noted that both assumptions are implied by DLIN.

Theorem 8. *Trapdoor commitment scheme TC1 is perfectly hiding, perfect trapdoor, and computationally binding under the* SDP *assumption.*

Proof. For perfect hiding and perfect trapdoor, observe that, for any $(\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{G}_T^2$, any $\vec{\mathbf{m}} \in \mathbb{G}_2^k$, there exits a unique $(\mathbf{r}, \mathbf{u}) \in \mathbb{G}_2^2$ that fulfills relation (4.1).

For computational binding, suppose that there exists an adversary that successfully opens a commitment to two distinct messages. We show that one can break SDP by using such an adversary. Given an instance of SDP, $(\Lambda, \mathbf{g}_r, \mathbf{h}_u, \mathbf{g}_z, \mathbf{h}_z)$, do as follows.

- Set $\mathbf{g}_i = \mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_z^{\chi_i}\mathbf{h}_u^{\delta_i}$ for $i = 1, \ldots, k$, where $\chi_i, \gamma_i, \delta_i \leftarrow \mathbb{Z}_p$. As the probability that any $\mathbf{g}_i$ or $\mathbf{h}_i$, $i = 1, \ldots, k$, is equal to $1_{\mathbb{G}_1}$ is negligible, the reduction algorithm simply aborts in such cases. Otherwise, all group elements are from $\mathbb{G}_1^*$ and chosen uniformly at random, like in the key generation algorithm. Run the adversary with $\mathtt{ck} = (\mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k)$.

- Given two openings $(\vec{\mathbf{m}}, \mathbf{r}, \mathbf{u})$ and $(\vec{\mathbf{m}}', \mathbf{r}', \mathbf{u}')$ from the adversary, compute

$$\mathbf{z}^\star = \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\chi_i}, \quad \mathbf{r}^\star = \frac{\mathbf{r}}{\mathbf{r}'} \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\gamma_i}, \quad \mathbf{u}^\star = \frac{\mathbf{u}}{\mathbf{u}'} \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\delta_i}. \tag{4.2}$$

- Output $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star)$.

Since the openings fulfills (4.1), we have

$$\begin{aligned} 1 &= \hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}}{\mathbf{r}'}\right) \prod \hat{e}\left(\mathbf{g}_i, \frac{\mathbf{m}_i}{\mathbf{m}_i'}\right) \\ &= \hat{e}\left(\mathbf{g}_z, \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\chi_i}\right) \hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}}{\mathbf{r}'} \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\gamma_i}\right) \;=\; \hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\, \hat{e}(\mathbf{g}_r, \mathbf{r}^\star) \end{aligned}$$

and

$$\begin{aligned} 1 &= \hat{e}\left(\mathbf{h}_u, \frac{\mathbf{u}}{\mathbf{u}'}\right) \prod \hat{e}\left(\mathbf{h}_i, \frac{\mathbf{m}_i}{\mathbf{m}_i'}\right) \\ &= \hat{e}\left(\mathbf{h}_z, \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\chi_i}\right) \hat{e}\left(\mathbf{h}_u, \frac{\mathbf{u}}{\mathbf{u}'} \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\delta_i}\right) \;=\; \hat{e}(\mathbf{h}_z, \mathbf{z}^\star)\, \hat{e}(\mathbf{h}_u, \mathbf{u}^\star). \end{aligned}$$

But $\vec{\mathbf{m}} \neq \vec{\mathbf{m}}'$, so there exists i such that $\mathbf{m}_i/\mathbf{m}'_i \neq 1$. Also, χ_i is independent from the view of the adversary. That is, for every choice of χ_i, there exist corresponding γ_i and δ_i that gives the same $\mathbf{g}_i$ and $\mathbf{h}_i$. Therefore, $\mathbf{z}^\star = \prod_i (\mathbf{m}_i/\mathbf{m}'_i)^{\chi_i} \neq 1$ with overwhelming probability. Hence $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star)$ is a valid answer to the instance of SDP. □

4.1.2 Scheme TC2

This is the most efficient scheme both in computation and storage. The scheme virtually 'half' the scheme of TC1. Let $\Lambda \in \{\Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$.

TC2.Key(1^λ): Choose random generators $\mathbf{g}_r$ from $\mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose γ_i from $\mathbb{Z}_p^*$ and compute $\mathbf{g}_i = \mathbf{g}_r^{\gamma_i}$. Output commitment-key $\mathtt{ck} = (\Lambda, \mathbf{g}_r, \mathbf{g}_1, \ldots, \mathbf{g}_k)$ and trapdoor $\mathtt{tk} = (\gamma_1, \ldots, \gamma_k)$.

TC2.Com($\mathtt{ck}, \vec{\mathbf{m}}$): Choose $\mathbf{r}$ randomly from $\mathbb{G}_2$, and compute

$$\mathfrak{c} = \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i). \tag{4.3}$$

Output commitment c and decommitment $\mathfrak{d} = \mathbf{r}$.

TC2.Vrf($\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathfrak{d}$): Output 1 if (4.3) holds. Output 0, otherwise.

TC2.Sim($\mathtt{ck}, \mathtt{tk}$): Choose $\mathbf{r}$ randomly from $\mathbb{G}_2$ and compute $\mathfrak{c} = \hat{e}(\mathbf{g}_r, \mathbf{r})$. Output commitment c and equivocation-key $\mathtt{ek} = (\mathtt{tk}, \mathbf{r})$.

TC2.Open($\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathtt{ek}$): Parse $\mathtt{ek}$ as $(\mathtt{tk}, \mathbf{r})$ and $\mathtt{tk}$ as $(\gamma_1, \ldots, \gamma_k)$. Then compute $\mathbf{r}' = \mathbf{r} \cdot \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}$, and Then output decommitment $\mathfrak{d} = \mathbf{r}'$.

Theorem 9. *TC2 is perfectly hiding, perfect trapdoor, and computationally binding if the* DBP *assumption holds for* Λ.

Proof. The hiding and trapdoor properties holds because, for any commitment $c \in \mathbb{G}_T$ and any $\vec{\mathbf{m}} \in \mathbb{G}_2^k$, there exists consistent $t \in \mathbb{G}_2$ that fulfills relation (4.3).

The binding property is shown similarly to Theorem 8:
Given an instance of DBP, $(\Lambda, \mathbf{g}_z, \mathbf{g}_r)$, do as follows.

- Set $\mathbf{g}_i = \mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}$. Run the adversary with $\mathtt{ck} = (\mathbf{g}_r, \{\mathbf{g}_i\}_{i=1}^k)$.
- Given two openings $(\vec{\mathbf{m}}, \mathbf{r})$ and $(\vec{\mathbf{m}}', \mathbf{r}')$ from the adversary, compute

$$\mathbf{z}^\star = \prod_{i=1}^{k}(\mathbf{m}_i/\mathbf{m}_i')^{\chi_i}, \qquad \mathbf{r}^\star = (\mathbf{r}/\mathbf{r}')\prod_{i=1}^{k}(\mathbf{m}_i/\mathbf{m}_i')^{\gamma_i}. \tag{4.4}$$

- Output $(\mathbf{z}^\star, \mathbf{r}^\star)$.

Since the openings fulfills (4.1), we have

$$1 = \hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}}{\mathbf{r}'}\right)\prod \hat{e}\left(\mathbf{g}_i, \frac{\mathbf{m}_i}{\mathbf{m}_i'}\right) = \hat{e}\left(\mathbf{g}_z, \prod_{i=1}^{k}\left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\chi_i}\right)\hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}}{\mathbf{r}'}\prod_{i=1}^{k}\left(\frac{\mathbf{m}_i}{\mathbf{m}_i'}\right)^{\gamma_i}\right)$$
$$= \hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\,\hat{e}(\mathbf{g}_r, \mathbf{r}^\star).$$

But $\vec{\mathbf{m}} \neq \vec{\mathbf{m}}'$, so there exists i such that $\mathbf{m}_i/\mathbf{m}_i' \neq 1$. Also, χ_i is independent from the view of the adversary. That is, for every choice of χ_i, there exist corresponding γ_i that gives the same $\mathbf{g}_i$. Therefore, $\mathbf{z}^\star = \prod_i(\mathbf{m}_i/\mathbf{m}_i')^{\chi_i} \neq 1$ with overwhelming probability. Hence $(\mathbf{z}^\star, \mathbf{r}^\star)$ is a valid answer to the instance of DBP. □

4.1.3 Scheme TC3

All components of this scheme are in $\mathbb{G}_1$ and $\mathbb{G}_2$. The underlying idea is to use TC1 and, instead of publishing a commitment in $\mathbb{G}_T$, we publish the decommitment and the message in a randomized way by applying the one-side randomization RandOneSide from Section 2.5. This construction works in the Λ_{sym} setting.

$\mathsf{TC3.Key}(1^\lambda)$: Choose random generators $\mathbf{g}_r, \mathbf{h}_u$ from $\mathbb{G}_2^*$. For $i = 1, \ldots, k$, choose γ_i and δ_i from $\mathbb{Z}_p^*$ and compute $\mathbf{g}_i = \mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_u^{\delta_i}$. Output commitment-key $\mathtt{ck} = (\Lambda, \mathbf{g}_r, \mathbf{h}_u, \ldots, \mathbf{g}_k, \mathbf{h}_k)$ and trapdoor $\mathtt{tk} = (\gamma_1, \delta_1 \ldots, \gamma_k, \delta_k)$.

$\mathsf{TC3.Com}(\mathtt{ck}, \vec{\mathbf{m}})$: Choose $\mathbf{r}$ and $\mathbf{u}$ randomly from $\mathbb{G}_2$, and compute

$$\{\mathbf{c}_{a,i}\}_{i=0}^k \leftarrow \mathsf{RandOneSide}((\mathbf{g}_r, \mathbf{r}), (\mathbf{g}_1, \mathbf{m}_1), \ldots, (\mathbf{g}_k, \mathbf{m}_k)), \text{ and} \tag{4.5}$$

$$\{\mathbf{c}_{b,i}\}_{i=0}^k \leftarrow \mathsf{RandOneSide}((\mathbf{h}_u, \mathbf{u}), (\mathbf{h}_1, \mathbf{m}_1), \ldots, (\mathbf{h}_k, \mathbf{m}_k)). \tag{4.6}$$

Output commitment $\mathfrak{c} = (\{\mathbf{c}_{a,i}\}_{i=0}^k, \{\mathbf{c}_{b,i}\}_{i=0}^k)$ and decommitment $\mathfrak{d} = (\mathbf{r}, \mathbf{u})$.

$\mathsf{TC3.Vrf}(\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathfrak{d})$: Parse c into $(\{\mathbf{c}_{a,i}\}_{i=0}^k, \{\mathbf{c}_{b,i}\}_{i=0}^k) \in \mathbb{G}_2^{2k+2}$ and $\mathfrak{d}$ into $(\mathbf{r}, \mathbf{u}) \in \mathbb{G}_2^2$. Output 1 if they satisfy the following predicates. Output 0, otherwise.

$$1 = \hat{e}(\mathbf{g}_r, \mathbf{r}/\mathbf{c}_{a,0}) \prod_{i=1}^k \hat{e}(\mathbf{g}_i, \mathbf{m}_i/\mathbf{c}_{a,i}) \quad \text{and} \quad 1 = \hat{e}(\mathbf{h}_u, \mathbf{u}/\mathbf{c}_{b,0}) \prod_{i=1}^k \hat{e}(\mathbf{h}_i, \mathbf{m}_i/\mathbf{c}_{b,i})$$

$\mathsf{TC3.Sim}(\mathtt{ck}, \mathtt{tk})$: Do the same as $\mathsf{TC3.Com}$ with $\vec{\mathbf{m}} = (1, \ldots, 1)$ and set $\mathtt{ek} = (\mathtt{tk}, \mathbf{r}, \mathbf{u})$.

$\mathsf{TC3.Open}(\mathtt{ck}, \mathfrak{c}, \vec{\mathbf{m}}, \mathtt{ek})$: Parse $\mathtt{ek}$ into $(\mathtt{tk}, \mathbf{r}, \mathbf{u})$ and $\mathtt{tk}$ into $(\gamma_1, \delta_1 \ldots, \gamma_k, \delta_k)$. Then compute $\mathbf{r}' = \mathbf{r} \cdot \prod_{i=1}^k \mathbf{m}_i^{-\gamma_i}$, and $\mathbf{u}' = \mathbf{u} \cdot \prod_{i=1}^k \mathbf{m}_i^{-\delta_i}$. Then output decommitment $\mathfrak{d} = (\mathbf{r}', \mathbf{u}')$.

Theorem 10. *Trapdoor commitment scheme TC3 is perfectly hiding, perfect trapdoor, and computationally binding under the* SDP *assumption.*

The hiding property is clear from the uniform output property of RandOneSide and that of TC1, and so is the perfect trapdoor property due to the uniqueness of the opening of a commitment to any message. The binding property is taken over from TC1 and can be proven in the same way as for TC1.

One can have a variant of TC2 whose commitment is in $\mathbb{G}_1$ and $\mathbb{G}_2$ in a similar way we convert TC1 to TC3. Unlike the previous case, however, RandOneSide cannot be used as TC2 is in $\Lambda = \Lambda_{\mathsf{sxdh}}$. So we instead use RandSeq keeping $\mathbf{g}_r$ and $\mathbf{h}_u$ intact. This modification results in $2k+1$ group elements in a commitment, which is 1 element less than that of TC3. However, depending on the applications, this may be less efficient since the verification predicate is not one-sided.

4.1.4 Scheme TC4

Like the previous commitment scheme, this one has both its commitments and decommitments to be group elements of the base groups. However, the message is not a group element but a scalar. This allows us to construct a more efficient scheme than the last one, and we use it as a building block in Section 5.6.1.

Let $\mathbf{g} \in \mathbb{G}_1$ and $\tilde{\mathbf{g}} \in \mathbb{G}_2$ be random bases. Common parameter Λ is given to all algorithms described below.

$\mathsf{TC4.Key}(1^\lambda)$: Select $\gamma \in \mathbb{Z}_p$ and set $\tilde{\mathbf{f}} = \tilde{\mathbf{g}}^\gamma$. Output commitment key $\mathtt{ck} = (\Lambda, \tilde{\mathbf{f}})$ and trapdoor $\mathtt{tk} = \gamma$.

$\mathsf{TC4.Com}(\mathtt{ck}, m)$: Choose random $\delta \in \mathbb{Z}_p$ and compute commitment $\mathfrak{c} = \tilde{\mathbf{g}}^m \tilde{\mathbf{f}}^\delta \in \mathbb{G}_2$ and decommitment $\mathfrak{d} = \mathbf{g}^\delta \in \mathbb{G}_1$. Output $\mathfrak{c}$ and $\mathfrak{d}$.

$\mathsf{TC4.Vrf}(\mathtt{ck}, \mathfrak{c}, m, \mathfrak{d})$: Output 1 if $\hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^m) = \hat{e}(\mathfrak{d}, \tilde{\mathbf{f}})$. Output 0, otherwise.

$\mathsf{TC4.Sim}(\mathtt{ck}, \mathtt{tk})$: Choose random $\delta \in \mathbb{Z}_p$ and output a commitment $\mathfrak{c} = \tilde{\mathbf{f}}^\delta$ and an equivocation-key $\mathtt{ek} = (\mathtt{tk}, \delta)$.

$\mathsf{TC4.Open}(\mathtt{ck}, \mathfrak{c}, m, \mathtt{ek})$: Parse $\mathtt{ek}$ as (γ, δ). Output $\mathfrak{d} = \mathbf{g}^{\delta - m/\gamma}$.

The correctness follows since

$$\hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^m) = \hat{e}(\mathbf{g}, \tilde{\mathbf{f}}^\delta) = \hat{e}(\mathbf{g}^\delta, \tilde{\mathbf{f}}) = \hat{e}(\mathfrak{d}, \tilde{\mathbf{f}}).$$

The trapdoor property holds because

$$\hat{e}(\mathfrak{d}, \tilde{\mathbf{f}}) = \hat{e}(\mathbf{g}^{\delta - m/\gamma}, \tilde{\mathbf{f}}) = \hat{e}(\mathbf{g}, \tilde{\mathbf{g}}^{-m}\tilde{\mathbf{f}}^\delta) = \hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^{\mathbf{m}}).$$

To prove computational binding property, we assume that the following variant of Diffie-Hellman inversion problem (XDHI) is hard with respect to Λ.

Assumption 8 (XDHI)**.** *Given* Λ *and* $(\mathbf{g}, \tilde{\mathbf{g}}, \tilde{\mathbf{g}}^\alpha) \in \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{G}_2^*$ *for random* $\alpha \in \mathbb{Z}_p^*$*, it is hard to compute* $\mathbf{g}^{1/\alpha} \in \mathbb{G}_1$*.*

Depending on setting Λ, the XDHI assumption is implied by basic assumptions: Computational Diffie-Hellman Assumption (CDH), Computational Co-Diffie-Hellman Assumption (co-CDH), and Decisional Diffie-Hellman Assumption in $\mathbb{G}_2$ ($\mathrm{DDH}_{\mathbb{G}_2}$), as follows. Note that, CDH is implied by DLIN in Λ_{sym} and $\mathrm{DDH}_{\mathbb{G}_2}$ is implied by SXDH in Λ_{sxdh}.

Lemma 1. CDH $\Rightarrow$ XDHI *for* Λ_{sym}*.* co-CDH $\Rightarrow$ XDHI *for* Λ_{xdh}*.* $\mathrm{DDH}_{\mathbb{G}_2} \Rightarrow$ XDHI *for* Λ_{sxdh}*.*

Proof. Let $\mathcal{A}$ be an XDHI adversary. In Λ_{sym}, given an CDH instance $(\mathbf{g}, \mathbf{g}^\alpha, \mathbf{g}^\beta)$, input $(\mathbf{g}^\alpha, \mathbf{g}^\beta, \mathbf{g})$ to $\mathcal{A}$. It outputs $g^{\alpha\beta}$, which is the answer to the CDH instance. Next, in Λ_{xdh}, given an co-CDH instance $(\mathbf{g}^\alpha, \tilde{\mathbf{g}}, \tilde{\mathbf{g}}^\beta) \in \mathbb{G}_2^{*3}$, input $(\mathbf{g}^\alpha, \tilde{\mathbf{g}}^\beta, \tilde{\mathbf{g}})$ to $\mathcal{A}$. It outputs $g^{\alpha\beta}$, which is the answer to the co-CDH instance. In Λ_{sxdh}, observe that, on input $(\mathbf{g}, \tilde{\mathbf{g}}^\delta, \tilde{\mathbf{g}})$, adversary $\mathcal{A}$ outputs g^δ. Thus $\mathcal{A}$ provides a mapping from $\mathbb{G}_2$ to $\mathbb{G}_1$. Now, given an instance $(\tilde{\mathbf{g}}, \tilde{\mathbf{g}}^\alpha, \tilde{\mathbf{g}}^\beta, \tilde{\mathbf{g}}^\gamma)$ of $\mathrm{DDH}_{\mathbb{G}_2}$, input $(\mathbf{g}, \tilde{\mathbf{g}}^\alpha, \tilde{\mathbf{g}})$ to $\mathcal{A}$ and receive g^α. Then $\gamma = \alpha\beta$ can be tested by checking if $\hat{e}(\mathbf{g}^\alpha, \tilde{\mathbf{g}}^\beta) = \hat{e}(\mathbf{g}, \tilde{\mathbf{g}}^\gamma)$ holds or not. □

Theorem 11. *Trapdoor commitment scheme* TC4 *is perfectly hiding. It is binding if the* XDHI *assumption holds for* Λ.

Proof. The perfect hiding and perfect trapdoor properties hold from the fact that, for any $c \in \mathbb{G}_2$, for every $m \in \mathbb{Z}_p$ there exists a single consistent $\delta \in \mathbb{Z}_p$.

The binding property is proven by showing a reduction to XDHI. Given an instance of XDHI, $(\mathbf{g}, \tilde{\mathbf{g}}, \tilde{\mathbf{g}}^{\alpha})$, set $\tilde{\mathbf{f}} = \tilde{\mathbf{g}}^{\alpha}$. Suppose that an adversary outputs a commitment $\mathfrak{c}$ correctly opened to $(m, \mathfrak{d})$ and $(m', \mathfrak{d}')$ for $m \neq m'$. Then $\hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^{\mathbf{m}}) = \hat{e}(\mathfrak{d}, \tilde{\mathbf{f}})$ and $\hat{e}(\mathbf{g}, \mathfrak{c}/\tilde{\mathbf{g}}^{m'}) = \hat{e}(\mathfrak{d}', \tilde{\mathbf{f}})$ hold. By dividing both sides of the equations, we have $\hat{e}(\mathbf{g}, \tilde{\mathbf{g}}^{m-m'}) = \hat{e}(\mathfrak{d}'/\mathfrak{d}, \tilde{\mathbf{f}}) = \hat{e}(\mathfrak{d}'/\mathfrak{d}, \tilde{\mathbf{g}}^{\alpha})$. Thus $(\mathfrak{d}'/\mathfrak{d})^{1/m-m'} = \mathbf{g}^{1/\alpha}$, which is a correct answer to the XDHI instance. □

4.2 One-Time Signatures

The above described trapdoor commitment scheme TC1 and TC2 can easily be extended into one-time signature schemes. Next we present these signature schemes which sign messages of fixed length k. Then, we discuss how to modify them to sign messages of *a priori* unbounded length deferring the technique details to Section 5.2.

4.2.1 A One-Time Signature Scheme in Any Setting

Let $\Lambda \in \{\Lambda_{\mathsf{sym}}, \Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$.

- $\mathsf{OTS1.Key}(1^{\lambda})$: Choose random generators $\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u \leftarrow \mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose χ_i, γ_i, $\delta_i \leftarrow \mathbb{Z}_p$ and compute $(\mathbf{g}_i, \mathbf{h}_i) = (\mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}, \mathbf{h}_z^{\chi_i}\mathbf{h}_u^{\delta_i})$. Also choose $\zeta, \rho, \varphi \leftarrow \mathbb{Z}_p$ and set $\mathbf{a} = \mathbf{g}_z^{\zeta}\mathbf{g}_r^{\rho}$ and $\mathbf{b} = \mathbf{h}_z^{\zeta}\mathbf{h}_u^{\varphi}$. Set $\mathtt{vk} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k, \mathbf{a}, \mathbf{b})$ and $\mathtt{sk} = (\mathtt{vk}, \zeta, \rho, \varphi, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^k)$. Output $(\mathtt{vk}, \mathtt{sk})$.

- $\mathsf{OTS1.Sign}(\mathtt{sk}, \vec{\mathbf{m}})$: Compute

$$\mathbf{z} = \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k} \mathbf{m}_i^{-\chi_i}, \quad \mathbf{r} = \tilde{\mathbf{g}}^{\rho} \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}, \quad \mathbf{u} = \tilde{\mathbf{g}}^{\varphi} \prod_{i=1}^{k} \mathbf{m}_i^{-\delta_i}.$$

  Output $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{u})$ as a signature.

- $\mathsf{OTS1.Vrf}(\mathtt{vk}, \vec{\mathbf{m}}, \sigma)$: Parse σ into $(\mathbf{z}, \mathbf{r}, \mathbf{u})$. Output 1 if the following equations hold. Output 0, otherwise.

$$\hat{e}(\mathbf{a}, \tilde{\mathbf{g}}) = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \tag{4.7}$$

$$\hat{e}(\mathbf{b}, \tilde{\mathbf{g}}) = \hat{e}(\mathbf{h}_z, \mathbf{z})\ \hat{e}(\mathbf{h}_u, \mathbf{u})\ \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i) \tag{4.8}$$

Theorem 12. *One-time signature scheme OTS1 is strongly unforgeable against one-time chosen message attacks if* SDP *holds for* Λ.

Proof. Suppose that there is an adversary, $\mathcal{A}$, that finds a forged signature $\sigma^{\dagger} = (\mathbf{z}^{\dagger}, \mathbf{r}^{\dagger}, \mathbf{u}^{\dagger})$ for message $\vec{\mathbf{m}}^{\dagger}$ after seeing a one-time signature $(\mathbf{z}, \mathbf{r}, \mathbf{u})$ for message $\vec{\mathbf{m}}$ of its choice. We construct a reduction algorithm to SDP as follows.

Given an instance $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u)$ of SDP, do the same as $\mathsf{OTS1.Key}$ by using the input instance as the bases. When $\mathcal{A}$ submit message $\vec{\mathbf{m}}$, run $\mathsf{OTS1.Sign}$ and return $(\mathbf{z}, \mathbf{r}, \mathbf{u})$ to $\mathcal{A}$. Given output $(\mathbf{z}^{\dagger}, \mathbf{r}^{\dagger}, \mathbf{u}^{\dagger})$ and $\vec{\mathbf{m}}^{\dagger}$ from $\mathcal{A}$, compute

$$\mathbf{z}^{\star} = \left(\frac{\mathbf{z}^{\dagger}}{\mathbf{z}}\right) \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i^{\dagger}}{\mathbf{m}_i}\right)^{\chi_i}, \ \mathbf{r}^{\star} = \left(\frac{\mathbf{r}^{\dagger}}{\mathbf{r}}\right) \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i^{\dagger}}{\mathbf{m}_i}\right)^{\gamma_i}, \ \mathbf{u}^{\star} = \left(\frac{\mathbf{u}^{\dagger}}{\mathbf{u}}\right) \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i^{\dagger}}{\mathbf{m}_i}\right)^{\delta_i}. \tag{4.9}$$

Then output $(\mathbf{z}^{\star}, \mathbf{r}^{\star}, \mathbf{u}^{\star})$. This completes the description of the reduction algorithm.

Suppose that adversary $\mathcal{A}$ is successful. By dividing both sides of (4.7) with

respect to $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star)$ and $(\mathbf{z}, \mathbf{r}, \mathbf{u})$, we have

$$\begin{aligned}
1 &= \hat{e}(\mathbf{g}_z, \mathbf{z}^\dagger/\mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}^\dagger/\mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i^\dagger/\mathbf{m}_i) \\
&= \hat{e}(\mathbf{g}_z, \mathbf{z}^\dagger/\mathbf{z} \prod_{i=1}^{k} (\mathbf{m}_i^\dagger/\mathbf{m}_i)^{\chi_i})\, \hat{e}(\mathbf{g}_r, \mathbf{r}^\dagger/\mathbf{r} \prod_{i=1}^{k} (\mathbf{m}_i^\dagger/\mathbf{m}_i)^{\gamma_i}) \\
&= \hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\, \hat{e}(\mathbf{g}_r, \mathbf{r}^\star).
\end{aligned}$$

Similarly, with respect to (4.8), we have

$$\begin{aligned}
1 &= \hat{e}(\mathbf{h}_z, \mathbf{z}^\dagger/\mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u}^\dagger/\mathbf{u}) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i^\dagger/\mathbf{m}_i) \\
&= \hat{e}(\mathbf{h}_z, \mathbf{z}^\dagger/\mathbf{z} \prod_{i=1}^{k} (\mathbf{m}_i^\dagger/\mathbf{m}_i)^{\chi_i})\, \hat{e}(\mathbf{h}_u, \mathbf{u}^\dagger/\mathbf{u} \prod_{i=1}^{k} (\mathbf{m}_i^\dagger/\mathbf{m}_i)^{\delta_i}) \\
&= \hat{e}(\mathbf{h}_z, \mathbf{z}^\star)\, \hat{e}(\mathbf{h}_u, \mathbf{u}^\star).
\end{aligned}$$

Hence $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star)$ is a correct answer to the SDP instance.

What remains to show is $\mathbf{z}^\star \neq 1$. We first consider the case of $\vec{\mathbf{m}} = \vec{\mathbf{m}}^\dagger$. In this case, $(\mathbf{z}^\dagger, \mathbf{r}^\dagger, \mathbf{u}^\dagger) \neq (\mathbf{z}, \mathbf{r}, \mathbf{u})$ must hold. Observe that $\mathbf{z}^\dagger = \mathbf{z}$ cannot be the case since it implies $\mathbf{r}^\dagger = \mathbf{r}$ and $\mathbf{u}^\dagger = \mathbf{u}$ to fulfill (4.7) and (4.8). Thus we have $\mathbf{z}^\dagger \neq \mathbf{z}$ and $\mathbf{z}^\star = \mathbf{z}^\dagger/\mathbf{z} \neq 1$. Next we consider the case of $\vec{\mathbf{m}} \neq \vec{\mathbf{m}}^\dagger$. In this case, there exists $i^\star$ for which $\mathbf{m}_{i^\star} \neq \mathbf{m}_{i^\star}^\dagger$ holds. For such $i^\star$, parameter $\chi_{i^\star}$ is information theoretically hidden from the view of the adversary. Namely, for any view of the adversary and for any $\chi_{i^\star}$, there exists a consistent coin toss which yields the same view. This can be verified by seeing that $(\mathbf{a}, \mathbf{b})$, and $(\mathbf{g}_{i^\star}, \mathbf{h}_{i^\star})$ are perfectly hiding commitments of ζ and $\chi_{i^\star}$, and the one-time signature does not identify them despite establishing relation between them. Therefore, due to the term $(\mathbf{m}_{i^\star}^\dagger/\mathbf{m}_{i^\star})^{\chi_{i^\star}}$, for $\mathbf{m}_{i^\star}^\dagger \neq \mathbf{m}_{i^\star}$, the probability that $\mathbf{z}^\star = 1$ is negligible. □

4.2.2 More Efficient Scheme in the Asymmetric Setting

In the case of $\Lambda \in \{\Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$ we can construct a more efficient scheme, call it $\mathsf{OTS2}$, that halves $\mathsf{OTS1}$ just like $\mathsf{TC2}$ does for $\mathsf{TC1}$. The scheme is defined as follows:

- $\mathsf{OTS2.Key}(1^\lambda)$: Choose random generators $\mathbf{g}_z, \mathbf{g}_r \leftarrow \mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose $\chi_i, \gamma_i \leftarrow \mathbb{Z}_p$ and compute $\mathbf{g}_i = \mathbf{g}_z^{\chi_i} \mathbf{g}_r^{\gamma_i}$. Also choose $\zeta, \rho \leftarrow \mathbb{Z}_p$ and set $\mathbf{a} = \mathbf{g}_z^{\zeta} \mathbf{g}_r^{\rho}$. Set $\mathtt{vk} = (\mathbf{g}_z, \mathbf{g}_r, \{\mathbf{g}_i\}_{i=1}^k, \mathbf{a})$ and $\mathtt{sk} = (\mathtt{vk}, \zeta, \rho, \{\chi_i, \gamma_i\}_{i=1}^k)$. Output $(\mathtt{vk}, \mathtt{sk})$.

- $\mathsf{OTS2.Sign}(\mathtt{sk}, \vec{\mathbf{m}})$: Compute

$$\mathbf{z} = \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k} \mathbf{m}_i^{-\chi_i} \qquad \text{and} \qquad \mathbf{r} = \tilde{\mathbf{g}}^{\rho} \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}$$

  Output $\sigma = (\mathbf{z}, \mathbf{r})$ as a signature.

- $\mathsf{OTS2.Vrf}(\mathtt{vk}, \vec{\mathbf{m}}, \sigma)$: Parse σ into $(\mathbf{z}, \mathbf{r})$. Output 1 if the following equation holds. Output 0, otherwise.

$$\hat{e}(\mathbf{a}, \tilde{\mathbf{g}}) = \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \tag{4.10}$$

Theorem 13. *One-time signature scheme $\mathsf{OTS2}$ is strongly unforgeable against one-time chosen message attacks if* DBP *holds for* Λ.

The proof is analogous to that of Theorem 12 and omitted.

4.2.3 Signing Unbounded-Size Messages

Using $\mathsf{OTS1}$ from Section 4.2.1 we construct $\mathsf{OTS1u}$ that can sign unbounded-size message. (Thus it is an automorphic one-time signature scheme.) The idea is to sign a block of message together with a fresh verification-key used to sign the next message block. A problem is that the verification-key of $\mathsf{OTS1}$ is too large and not

covered by its message space. We can get around the problem by reusing the bases $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k)$ and only renew $(\mathbf{a}, \mathbf{b})$ for every message block. One way to observe that is by viewing the former part of the key as a commitment key and the latter one as a trapdoor commitment. The same trick is used in Section 5.5. The unforgeability against one-time chosen message attacks can be proven based on SDP. The proof is almost the same as that for $\mathsf{OTS1}$ and omitted. (Since fresh $\mathbf{a}$ and $\mathbf{b}$ brings new randomness ζ, the information theoretic nature exploited in the proof is preserved.)

In the asymmetric case $\Lambda = \Lambda_{\mathsf{sxdh}}$, one can do the similar construction based on $\mathsf{OTS2}$. Since $\mathbf{a}$ is not in the message space, we use dual signature scheme as in Section 5.4 and sign messages in $\mathbb{G}_2$ and $\mathbb{G}_1$ in alternating manner.

4.3 Applications

Our structure-preserving commitment schemes have various applications. For example, $\mathsf{TC1}$ and its corresponding one-time signature scheme $\mathsf{OTS1}$ are a major part of the construction of EUF-CMA structure-preserving signature scheme in Section 5.1; $\mathsf{TC4}$ is an essential building block of the round-optional blind signature scheme in Section 5.6.1; $\mathsf{TC1}$ and $\mathsf{TC2}$ are the key elements of the construction of leakage-resilient hard relations, which combined with tSE NIZK yield the first efficient EUF-CMA leakage-resilient signatures in the standard model; etc. In the rest of this section, we focus on the construction of the leakage-resilient signature scheme.

We begin by showing how to generically construct leakage-resilient hard relations from *second-preimage resistant (SPR)* relations. Informally, we say that a relation $\mathtt{R}$ is SPR if given a random $(x, y) \in \mathtt{R}$ it is hard to find $x' \neq x$ such that $(x', y) \in \mathtt{R}$. Then we give a generic construction of leakage-resilient signatures form leakage resilient

hard relations and tSE-NIZKs. And, finally, we instantiate it efficiently.

4.3.1 Leakage-Resilient Hard Relation

Let us first define the notion of an SPR relation:

Definition 20 (SPR Relation)**.** *A relation* $\mathtt{R}$ *with a randomized sampling algorithm* Sample *is a* second-preimage resistant *if:*

- *For any* $(y, x) \leftarrow \mathsf{Sample}(1^\lambda)$ *it holds that* $(x, y) \in \mathtt{R}$*.*
- *There is a polynomial-time algorithm that decides whether* $(x, y) \in \mathtt{R}$*.*
- *For any polynomial time adversary* $\mathcal{A}$ *the following experiment returns* 1 *with negligible probability.*

$\mathrm{Exp}_{\mathrm{SPR}}$*:*

$(y, x) \leftarrow \mathsf{Sample}(1^\lambda)$

$x^\star \leftarrow \mathcal{A}(y, x)$

output 1 *if* $\mathtt{R}(x^\star, y) = 1 \ \wedge \ x^\star \neq x$*; and* 0 *otherwise.*

Before proving our main theorem in this section, we need several definitions and a lemma that will be used in the proof of security.

Definition 21 (Min-Entropy)**.** *The* min-entropy *of a random variable* X*, denoted as* $\mathbf{H}_\infty(X)$*, is defined to be* $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$.

Definition 22 (Average-Conditional Min-Entropy [52])**.** *The* average-conditional min-entropy *of a random variable* X *conditioned on* Z*, denoted as* $\widetilde{\mathbf{H}}_\infty(X|Z)$ *is:*

$$\widetilde{\mathbf{H}}_\infty(X|Z) = -\log\left(\mathbb{E}_{z\leftarrow Z}\left[\max_x \Pr[X = x|Z = z]\right]\right) = -\log\left(\mathbb{E}_{z\leftarrow Z}\left[2^{\mathbf{H}_\infty[X|Z=z]}\right]\right)$$

Lemma 2 ([52]). *Let X, Y, Z be random variables where Z takes values in a set of size at most 2^ℓ. Then $\widetilde{\mathbf{H}}_\infty(X|(Y,Z)) \geq \widetilde{\mathbf{H}}_\infty((X,Y)|Z) - \ell \geq \widetilde{\mathbf{H}}_\infty(X|Z) - \ell$, and in particular, $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - \ell$*

Next we define the average-case pre-image entropy of the SPR relation to be $\mathbf{H}_{avg}(\mathtt{R}) = \widetilde{\mathbf{H}}_\infty(X \mid Y)$, where the random variables (X, Y) are distributed according to $\mathsf{Sample}(1^\lambda)$, and prove that any SPR relation is also a leakage resilient:

Theorem 14. *If $\mathtt{R}$ is an SPR relation, then it is also an ℓ-leakage resilient hard relation for $\ell = \mathbf{H}_{avg}(\mathtt{R}) - \omega(\log \lambda)$, where λ is the security parameter.*

Proof. We assume, for the sake of contradiction, that there exists an adversary $\mathcal{A}$ that succeeds in breaking the security of leakage-resilient hard relation $\mathtt{R}$ with non-negligible probability ϵ. We construct $\mathcal{B}$ that breaks the security of the SPR relation with non-negligible probability. On input (x, y), $\mathcal{B}$ simulates the environment of $\mathrm{Exp}_{LR\text{-}\mathtt{R}}$ for $\mathcal{A}$ on input y and responds to $\mathcal{A}$'s leakage queries using x. When $\mathcal{A}$ eventually outputs $x^\star$, $\mathcal{B}$ also outputs $x^\star$.

We know that $\Pr[R(x^\star, y) = 1] = \epsilon$ but we need to compute $\Pr[x^\star \neq x]$ since $\mathcal{B}$ only breaks the SPR property if $x^\star \neq x$. Notice that:

$$\begin{aligned}
&\Pr[\mathcal{B} \text{ succeeds}] \\
&= \Pr[\mathcal{A} \text{ succeeds} \wedge x \neq x^\star] \\
&\geq \Pr[\mathcal{A} \text{ succeeds}] - \Pr[x = x^\star] \\
&= \epsilon - \Pr[x = x^\star]
\end{aligned}$$

Notice that the only information that $\mathcal{A}$ has about x comes from y and the leakage queries. Let X, Y be the random variables for x, y respectively, and let Z be the random variable for the total leakage learned by $\mathcal{A}$. Then $\widetilde{\mathbf{H}}_\infty(X|(Y,Z)) \geq \widetilde{\mathbf{H}}_\infty(X|Y) - \ell$

and

$$\Pr[x = x^\star] \leq 2^{-\widetilde{\mathbf{H}}_\infty(X|Y)+\ell} = 2^{-\mathbf{H}_{avg}(\mathtt{R})+\ell}.$$

Assuming that $\ell \leq \mathbf{H}_{avg}(\mathtt{R}) - \omega(\log(\lambda))$ we have that $\Pr[\mathcal{B} \text{ succeeds}] \geq \epsilon - 2^{-\omega(\log(\lambda))}$, which is non-neglibible. □

4.3.2 Structure-Preserving SPR Relation

Previous constructions of leakage-resilient primitives often use the SPR function

$$f(x_1, \ldots, x_n) = \mathbf{g}_1^{x_1}\mathbf{g}_2^{x_2}\ldots\mathbf{g}_n^{x_n},$$

but it does not allow an efficient extraction of the witness $(x_n, \ldots, x_n)$ when using GS proofs. The only way to make it structure-preserving would be if the witness is committed bit by bit, but that would result in proofs growing linearly in the security parameter, among other things. To overcome this problem, we use SPR functions based on bilinear maps derived from TC1 and TC2. As we need our proofs to be not only witness indistinguishable but also zero-knowledge, we prefer the notion of an SPR relation to an SPR function. In particular, we use the structure-preserving SPR relation $\mathtt{R}((\mathbf{x}_1, \ldots, \mathbf{x}_n), \mathbf{y})$:

$$\hat{e}(\mathbf{h}_1, \mathbf{x}_1)\hat{e}(\mathbf{h}_2, \mathbf{x}_2)\ldots\hat{e}(\mathbf{h}_n, \mathbf{x}_n) = \hat{e}(\mathbf{y}, \tilde{\mathbf{g}})$$

for Λ_{sxdh}. Whereas for Λ_{sym} we use $\mathtt{R}((\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mathbf{y}_1, \mathbf{y}_2))$:

$$\hat{e}(\mathbf{h}_1, \mathbf{x}_1)\hat{e}(\mathbf{h}_3, \mathbf{x}_3)\ldots\hat{e}(\mathbf{h}_n, \mathbf{x}_n) = \hat{e}(\mathbf{g}, \mathbf{y}_1)$$
$$\hat{e}(\bar{\mathbf{h}}_2, \mathbf{x}_2)\hat{e}(\bar{\mathbf{h}}_3, \mathbf{x}_3)\ldots\hat{e}(\bar{\mathbf{h}}_n, \mathbf{x}_n) = \hat{e}(\mathbf{g}, \mathbf{y}_2)$$

As we show next, both constructions satisfy the properties of a structure-preserving cryptography primitive.

Based on SXDH. Let $\Lambda \in \{\Lambda_{\mathsf{sxdh}}, \Lambda_{\mathsf{sxdh}}\}$, $n \geq 2$ and $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n \in \mathbb{G}_1$ be fixed random elements for $\mathtt{R}$. We construct the SPR relation:

- $\mathsf{Sample}(\Lambda, \{\mathbf{h}_i\}_{i=1}^n)$: Choose $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$; set $\mathbf{y} = \prod_{i=1}^n \mathbf{h}_i^{r_i}$ and, for $i = 1, \ldots, n$, $\mathbf{x}_i = \tilde{\mathbf{g}}^{r_i}$; and output $(\mathbf{y}, \vec{\mathbf{x}})$.
- $\mathtt{R}(\vec{\mathbf{x}}, \mathbf{y})$: Output 1 if $\prod_{i=1}^n \hat{e}(\mathbf{h}_i, \mathbf{x}_i) = e(\mathbf{y}, \tilde{\mathbf{g}})$ and 0 otherwise.

Claim 1. *If* DBP *holds for* Λ*, the relation* $\mathtt{R}$ *described above is SPR with average-case preimage entropy* $\mathbf{H}_{avg}(\mathtt{R}) = (n-1)\log(p)$.

Proof. For any fixed choice of $\mathbf{y}$, the conditional distribution of $\vec{\mathbf{x}}$ is uniform over some $n-1$ dimensional subspace of $\mathbb{G}_2^n$, which gives us the average-case preimage entropy of $(n-1)\log(p)$.

To see that the relation is SPR, note that any adversary $\mathcal{A}$ which wins Exp_{SPR} immediately implies an adversary $\mathcal{B}$ which wins $\mathrm{Exp}_{Comp.Binding}$ for $\mathsf{TC2}$:

On input $\mathtt{ck} = (\Lambda, \mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$, define the corresponding relation $\mathtt{R}$, and sample $(\mathbf{y}, \vec{\mathbf{x}}) \leftarrow \mathsf{Sample}(\Lambda, \{\mathbf{h}_i\}_{i=1}^n)$. Let $\vec{\mathbf{x}}^\star \leftarrow \mathcal{A}(\mathbf{y}, \vec{\mathbf{x}})$ be the output of the adversary. Then, $\mathcal{B}$ outputs $(\mathfrak{c}, \vec{\mathbf{m}}_0, \vec{\mathbf{m}}_1, \mathfrak{d}_0, \mathfrak{d}_1)$, where $\mathfrak{c} = \hat{e}(\mathbf{y}, \tilde{\mathbf{g}})$, $\mathfrak{d}_0 = \mathbf{x}_1$, $\vec{\mathbf{m}}_0 = (\mathbf{x}_2, \ldots, \mathbf{x}_n)$, $\mathfrak{d}_1 = \mathbf{x}_1^\star$, and $\vec{\mathbf{m}}_1 = (\mathbf{x}_2^\star, \ldots, \mathbf{x}_n^\star)$. As $\mathcal{A}$ breaks the SPR property of $\mathtt{R}$, it must hold that $\mathtt{R}(\vec{\mathbf{x}}, \mathbf{y}) = \mathtt{R}(\vec{\mathbf{x}}^\star, \mathbf{y}) = 1$ and $\vec{\mathbf{x}} \neq \vec{\mathbf{x}}^\star$. But then $\mathsf{TC2.Open}(\mathfrak{c}, \vec{\mathbf{m}}_0, \mathfrak{d}_0) = \mathsf{TC2.Open}(\mathfrak{c}, \vec{\mathbf{m}}_1, \mathfrak{d}_1) = 1$ and $\vec{\mathbf{m}}_0 \neq \vec{\mathbf{m}}_1$, as $\vec{\mathbf{m}}_0 = \vec{\mathbf{m}}_1$ would imply $\mathfrak{d}_0 = \mathfrak{d}_1$ (and hence $\vec{\mathbf{x}} = \vec{\mathbf{x}}^\star$).

Then, by Theorem 9, it is true that $\mathtt{R}$ is SPR if the DBP assumption holds. □

As we know from Theorem 1, $\mathrm{DDH}_{\mathbb{G}_1}$ implies DBP; and together with Theorem 14, it gives us the following corollary.

Corollary 1. *The relation* $\mathtt{R}$ *described above is* ℓ*-leakage resilient hard relation for* $\ell = (n-1)\log(p) - \omega(\log\lambda)$ *under SXDH.*

Based on DLIN. Let $n \geq 3$ and $\mathbf{h}_1, \mathbf{h}_3, \ldots, \mathbf{h}_n, \bar{\mathbf{h}}_2, \ldots, \bar{\mathbf{h}}_n \in \mathbb{G}_1$ be fixed random elements for $\mathtt{R}$. We construct the SPR relation:

- $\mathsf{Sample}(\Lambda_{\mathsf{sym}}, \{\mathbf{h}_i\}, \{\bar{\mathbf{h}}_j\})$: Choose $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$; set $\mathbf{x}_i = \mathbf{g}^{r_i}$, for $i = 1, \ldots, n$, $\mathbf{y}_1 = \prod_{i \in \{1,3,\ldots,n\}} \mathbf{h}_i^{r_i}$, and $\mathbf{y}_2 = \prod_{i=2}^{n} \bar{\mathbf{h}}_i^{r_i}$; and output $((\mathbf{y}_1, \mathbf{y}_2), \vec{\mathbf{x}})$.
- $\mathtt{R}(\vec{\mathbf{x}}, \mathbf{y})$: Output 1 if $\prod_{i \in \{1,3,\ldots,n\}} \hat{e}(\mathbf{h}_i, \mathbf{x}_i) = e(\mathbf{g}, \mathbf{y}_1) \wedge \prod_{i=2}^{n} \hat{e}(\bar{\mathbf{h}}_i, \mathbf{x}_i) = e(\mathbf{g}, \mathbf{y}_2)$ and 0 otherwise.

Claim 2. *If* SDP *holds for* Λ_{sym}*, the relation* $\mathtt{R}$ *described above is SPR with average-case preimage entropy* $\mathbf{H}_{avg}(\mathtt{R}) = (n-2)\log(p)$.

Proof. For any fixed choice of $(\mathbf{y}_1, \mathbf{y}_2)$, the conditional distribution of $\vec{\mathbf{x}}$ is uniform over some $n-2$ dimensional subspace of $\mathbb{G}^n$, which gives us the average-case preimage entropy of $(n-2)\log(p)$.

To see that the relation is SPR, note that any adversary $\mathcal{A}$ which wins Exp_{SPR} immediately implies an adversary $\mathcal{B}$ which wins $\mathrm{Exp}_{Comp.Binding}$ for $\mathsf{TC1}$:

On input $\mathtt{ck} = (\Lambda_{\mathsf{sym}}, \mathbf{h}_1, \bar{\mathbf{h}}_2, \mathbf{h}_3, \bar{\mathbf{h}}_3, \ldots, \mathbf{h}_n, \bar{\mathbf{h}}_n)$, define the corresponding relation $\mathtt{R}$, and sample $((\mathbf{y}_1, \mathbf{y}_2), \vec{\mathbf{x}}) \leftarrow \mathsf{Sample}(\Lambda_{\mathsf{sym}}, \{\mathbf{h}_i\}, \{\bar{\mathbf{h}}_j\})$. Let $\vec{\mathbf{x}}^\star \leftarrow \mathcal{A}((\mathbf{y}_1, \mathbf{y}_2), \vec{\mathbf{x}})$ be the output of the adversary. Then, $\mathcal{B}$ outputs $(\mathfrak{c}, \vec{\mathbf{m}}_0, \vec{\mathbf{m}}_1, \mathfrak{d}_0, \mathfrak{d}_1)$, where $\mathfrak{c} = (\hat{e}(\mathbf{g}, \mathbf{y}_1), \hat{e}(\mathbf{g}, \mathbf{y}_2))$, $\mathfrak{d}_0 = (\mathbf{x}_1, \mathbf{x}_2)$, $\vec{\mathbf{m}}_0 = (\mathbf{x}_3, \ldots, \mathbf{x}_n)$, $\mathfrak{d}_1 = (\mathbf{x}_1^\star, \mathbf{x}_2^\star)$, and $\vec{\mathbf{m}}_1 = (\mathbf{x}_3^\star, \ldots, \mathbf{x}_n^\star)$. As $\mathcal{A}$ breaks the SPR property of $\mathtt{R}$, it must hold that $\mathtt{R}(\vec{\mathbf{x}}, (\mathbf{y}_1, \mathbf{y}_2)) = \mathtt{R}(\vec{\mathbf{x}}^\star, (\mathbf{y}_1, \mathbf{y}_2)) = 1$ and $\vec{\mathbf{x}} \neq \vec{\mathbf{x}}^\star$. But then

$$\mathsf{TC1.Open}(\mathfrak{c}, \vec{\mathbf{m}}_0, \mathfrak{d}_0) = \mathsf{TC1.Open}(\mathfrak{c}, \vec{\mathbf{m}}_1, \mathfrak{d}_1) = 1 \quad \text{and} \quad \vec{\mathbf{m}}_0 \neq \vec{\mathbf{m}}_1,$$

as $\vec{\mathbf{m}}_0 = \vec{\mathbf{m}}_1$ would imply $\mathfrak{d}_0 = \mathfrak{d}_1$ (and hence $\vec{\mathbf{x}} = \vec{\mathbf{x}}^\star$).

Then, by Theorem 8, it is true that $\mathtt{R}$ is SPR if the SDP assumption holds. □

As we know, DLIN implies SDP, so by Theorem 14 the following corollary holds.

Corollary 2. *The relation* $\mathtt{R}$ *described above is* ℓ*-leakage resilient hard relation for* $\ell = (n-2)\log(p) - \omega(\log\lambda)$ *in the* Λ_{sym} *setting under DLIN.*

4.3.3 Leakage-Resilient Signatures

In this section, we give a generic construction of leakage-resilient signatures based on leakage-resilient hard relations and tSE-NIZK arguments. Let $\mathtt{R}$ be an ℓ-leakage resilient hard relation with sampling algorithm $\mathsf{Sample}(1^\lambda)$. Let $\Pi = (\Pi.\mathsf{Crs}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrf})$ be a tSE-NIZK argument for a relation $\mathtt{R}$ supporting labels. Consider the following signature scheme:

- $\mathsf{LR\text{-}SIG.Key}(1^\lambda)$: Output $\mathtt{vk} = (\mathtt{crs}, y)$, where $(\mathtt{crs}, \mathtt{tk}, \mathtt{ek}) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$ and $(y, x) \leftarrow \mathsf{Sample}(1^\lambda)$, and $\mathtt{sk} = x$.
- $\mathsf{LR\text{-}SIG.Sign}(\mathtt{sk}, m)$: Compute $\pi \leftarrow \Pi.\mathsf{Prove}^m(y, x)$, where m is treated as a label, and output $\sigma = \pi$.
- $\mathsf{LR\text{-}SIG.Vrf}(\mathtt{vk}, m, \sigma)$: Output $\Pi.\mathsf{Vrf}^m(y, \sigma)$.

Theorem 15. *If* $\mathtt{R}(x, y)$ *is an* ℓ*-leakage resilient hard relation and* Π *is a labeled* tSE*-NIZK argument for* $\mathtt{R}$*, then the above signature scheme is an* ℓ*-leakage resilient signature scheme.*

Proof. Consider the following series of games and the success probability of an adversary $\mathcal{A}$ in the corresponding experiments:

Game 0**:** This is experiment as described in the definition of leakage-resilient signatures (Definition 14). Let $\mathcal{A}$'s output forgery be $(m^\star, \sigma^\star)$, where $\sigma^\star = \pi^\star$.

Game 1**:** We modify the signing oracle $\mathcal{O}_{\mathsf{sign}}(\cdot)$ in the way it answers $\mathcal{A}$'s queries. Instead of returning an argument π which is the output of $\Pi.\mathsf{Prove}$, it answers

each query for a message m with a simulated argument produced with $\Pi.\mathsf{Sim}$. Game 0 and Game 1 are indistinguishable by the zero-knowledge property of Π. Note that the simulated arguments given to $\mathcal{A}$ are always of true statements. As in the previous game, the experiment outputs 1 if $\mathcal{A}$ produces a valid forgery $(m^\star, \sigma^\star)$ such that $\Pi.\mathsf{Vrf}^{m^\star}(y, \sigma^\star) = 1$ and $m^\star$ is fresh, i.e., $m^\star \notin Q_m$.

Game 2: We change the winning condition: the experiment outputs 1 if $(m^\star, \sigma^\star)$ is a valid forgery *and* $\mathtt{R}(z^\star, y) = 1$, where $z^\star \leftarrow \Pi.\mathsf{Ext}(y, \pi^\star, \mathtt{ek})$. Game 1 and Game 2 are indistinguishable by the true-simulation extractability of Π.

We need to show that $\mathcal{A}$ has only a negligible probability of winning the experiment in Game 2, and because Game 0 and Game 2 are computationally indistinguishable, that will imply that $\mathcal{A}$ has a negligible probability of breaking LR-SIG.

For the sake of contradiction, assume that $\mathcal{A}$ wins the experiment in Game 2 with a non-negligible probability. We construct an adversary $\mathcal{B}$ which breaks the security of $\mathtt{R}$: on input y, $\mathcal{B}$ generates $(\mathtt{crs}, \mathtt{tk}, \mathtt{ek}) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$ and simulates the environment of the experiment in Game 2 for $\mathtt{vk} = (\mathtt{crs}, y)$. The queries to the signing oracle are answered using $\Pi.\mathsf{Sim}$ and the leakage queries are answered by $\mathcal{B}$'s own leakage oracle. In the end, $\mathcal{A}$ outputs a forgery $(m^\star, \pi^\star)$, for which $\mathcal{B}$ runs $z^\star \leftarrow \Pi.\mathsf{Ext}(y, \pi^\star, \mathtt{ek})$ and outputs $z^\star$. The probability that $\mathcal{B}$ outputs $z^\star$ for which $\mathtt{R}(z^\star, y) = 1$ is exactly that of $\mathcal{A}$ winning the experiment in Game 2. Therefore, $\mathcal{A}$ must have only a negligibility probability of forging a signature. □

Next, we instantiate the construction efficiently. Our construction is compared with previous works on EUF-CMA leakage-resilient signatures in Table 4.2. We point out that previously there was no efficient leakage resilient without resorting to the use of random oracles.

Reference	Unforgeability	Model	Leakage	Efficient?
[7]	Existential	*Random Oracle*	*1/2*	Yes
[7]	*Entropic*	*Random Oracle*	1	Yes
[75]	Existential	Standard	1	*No*
This Work	Existential	Standard	1	Yes

Table 4.2: Comparison of previous work on leakage-resilient signatures with our result.

4.3.4 Instantiation

In order to efficiently instantiate the construction from the last section, we need to give a leakage-resilient hard relation $\mathtt{R}$, a CCA encryption scheme, and an efficient NIZK argument for the relation $\hat{\mathtt{R}}$ defined in equation (3.1). We use the structure-preserving SPR relation, which yields a leakage-resilient hard relation as shown in Section 4.3.1, the K-linear Cramer-Shoup encryption described in Section 3.4.2 and the Groth-Sahai proof system reviewed in Section 2.4. The three of them share a common setup $\Lambda := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, \mathbf{g}, \tilde{\mathbf{g}})$ and are secure if the K-linear assumption hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$, for $K = 1$ or $K = 2$. So, in the former case, we set $\Lambda = \Lambda_{\mathsf{sxdh}}$; and, in the latter, $\Lambda = \Lambda_{\mathsf{sym}}$.

Instantiation 1: Based on SXDH.

SPR Relation. We use the above-described SPR relation in the Λ_{sxdh} setting. Recall its verification equation: $\hat{e}(\mathbf{h}_1, \mathbf{x}_1)\hat{e}(\mathbf{h}_2, \mathbf{x}_2) \ldots \hat{e}(\mathbf{h}_n, \mathbf{x}_n) = \hat{e}(\mathbf{y}, \tilde{\mathbf{g}})$. It has average-case preimage entropy of $(n-1)\log(p)$.

CCA-Secure Encryption. We use the multi-message Cramer-Shoup encryption scheme working in $\mathbb{G}_2$ to encrypt the relation witness. Formally, let the public key be $\mathtt{pk} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{d}_1, \ldots, \mathbf{d}_n, \mathbf{e}, \mathbf{f})$. To encrypt $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ under the same random-

ness r and label m, compute the ciphertext $\mathfrak{c} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \ldots, \mathbf{c}_{n+2}, \mathbf{c}_{n+3})$ as follows:

$$\mathfrak{c} \leftarrow \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Enc}^m(\mathbf{x}_1, \ldots, \mathbf{x}_n; r) = (\mathbf{g}_0^r, \mathbf{g}_1^r, \mathbf{x}_1\mathbf{d}_1^r, \ldots, \mathbf{x}_n\mathbf{d}_n^r, (\mathbf{e}\mathbf{f}^t)^r),$$

where $t = H(\mathbf{c}_1, \ldots, \mathbf{c}_{n+2}, m)$. The total size of the ciphertext is $n + 3$.

NIZK Argument. We use the Groth-Sahai proof system to produce an argument that "$\mathtt{R}(\vec{\mathbf{x}}, \mathbf{y}) = 1$ and $\mathfrak{c} = \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Enc}^m(\vec{\mathbf{x}}; r)$". First we show that $\mathtt{R}(\vec{\mathbf{x}}, \mathbf{y}) = 1$ by creating a commitment $\vec{\mathfrak{b}}_i = \mathsf{GSCom}(\mathbf{x}_i; (s_{i,0}, s_{i,1}))$ for each component $\mathbf{x}_i$ of the witness and producing proof elements which show that the committed values satisfy the pairing product equation $\prod_{i=1}^n \hat{e}(\mathbf{h}_i, \mathbf{x}_i) = \hat{e}(\mathbf{y}, \tilde{\mathbf{g}})$. Then, we show that $\mathfrak{c} = \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Enc}^m(\vec{\mathbf{x}}; r)$ using a system of one-sided multi-exponentiation equations with a witness $(r, \{s_{i,0}, s_{i,1}\}_{i=1}^n)$ to show that the plaintext encrypted in $\mathfrak{c}$ is equal to the committed values in $\vec{\mathfrak{b}}_i$, $i = 1, \ldots, n$. Details follow.

$$\text{Let} \qquad \vec{\mathfrak{b}}_1 = (\mathbf{x}_1, 1) \cdot \vec{\mathfrak{v}}_0^{\,s_{1,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{1,1}}, \quad \ldots, \quad \vec{\mathfrak{b}}_n = (\mathbf{x}_n, 1) \cdot \vec{\mathfrak{v}}_0^{\,s_{n,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{n,1}},$$

and, as defined above, $\mathfrak{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_{n+3})$. Proving equality of the plaintext and the committed values reduces to proving the satisfiability of the following system of $2n+3$ equations:

$$\frac{\vec{\mathfrak{b}}_1}{(\mathbf{c}_3, 1)} = \vec{\mathfrak{v}}_0^{\,s_{1,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{1,1}} \cdot (\mathbf{d}_1^{-1}, 1)^r\,, \quad \ldots \quad, \frac{\vec{\mathfrak{b}}_n}{(\mathbf{c}_{n+2}, 1)} = \vec{\mathfrak{v}}_0^{\,s_{n,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{n,1}} \cdot (\mathbf{d}_n^{-1}, 1)^r\,,$$
$$\mathbf{c}_1 = \mathbf{g}_0^r\,, \quad \mathbf{c}_2 = \mathbf{g}_1^r\,, \quad \mathbf{c}_{n+3} = \mathbf{e}^r(\mathbf{f}^t)^r\,.$$

The total size of the argument is $8n + 21$ group elements and 2 $\mathbb{Z}_p$-elements.

Combining the ciphertext and the NIZK argument makes the size of the signature $9n + 24$ group elements and 2 elements in $\mathbb{Z}_p$. By Theorem 14 and Theorem 15, we know that the above instantiation gives us a $((n-1)\log p - 1)$-leakage resilient signature scheme. To translate this into $(1-\epsilon)|sk|$ leakage tolerance, we need

$$n \geq \frac{1}{\epsilon} + \frac{\omega(\log \lambda)}{\epsilon \cdot \log p} = \frac{1}{\epsilon} \cdot \left(1 + \frac{\omega(\log \lambda)}{\log p}\right)$$

This gives us signatures of size $(9/\epsilon)(1+\omega(\log\lambda)/\log p)+24$ group elements and 2 elements in $\mathbb{Z}_p$.

Instantiation 2: Based on DLIN. In the case of $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, we give an instantiation under the DLIN assumption.

SPR Relation. We use the above-described structure-preserving SPR relation in the Λ_{sym} setting. Recall It has average-case preimage entropy of $(n-2)\log(p)$.

CCA-Secure Encryption. We use the multi-message Linear Cramer-Shoup encryption scheme to encrypt the witness. Formally, let the public key be:

$$\mathsf{pk} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{d}_{1,1}, \mathbf{d}_{1,2}, \ldots, \mathbf{d}_{n,1}, \mathbf{d}_{n,2}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{f}_1, \mathbf{f}_2).$$

To encrypt $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ under the same randomness (r_1, r_2) and label m, the compute the ciphertext$\mathfrak{c} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \ldots, \mathbf{c}_{n+2}, \mathbf{c}_{n+4})$ as follows:

$$\begin{aligned}\mathfrak{c} &\leftarrow \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Enc}^m(\vec{\mathbf{x}}, (r_1, r_2)) \\ &= (\mathbf{g}_0^{r_1+r_2}, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2},\ \mathbf{x}_1\mathbf{d}_{1,1}^{r_1}\mathbf{d}_{1,2}^{r_2}, \ldots, \mathbf{x}_n\mathbf{d}_{n,1}^{r_1}\mathbf{d}_{n,2}^{r_2}, (\mathbf{e}_1\mathbf{f}_1^t)^{r_1}(\mathbf{e}_2\mathbf{f}_2^t)^{r_2}),\end{aligned}$$

where $t = H(\mathbf{c}_1, \ldots, \mathbf{c}_{n+3}, m)$. The size of the ciphertext is $n+4$.

NIZK Argument. First we prove that $\mathtt{R}(\vec{\mathbf{x}}, (\mathbf{y}_1, \mathbf{y}_2)) = 1$ using the pairing product equations

$$\begin{aligned} e(\mathbf{h}_1, \mathbf{x}_1)e(\mathbf{h}_3, \mathbf{x}_3)\ldots e(\mathbf{h}_n, \mathbf{x}_n) &= e(\mathbf{g}, \mathbf{y}_1) \quad \text{and} \\ e(\bar{\mathbf{h}}_2, \mathbf{x}_2)e(\bar{\mathbf{h}}_3, \mathbf{x}_3)\ldots e(\bar{\mathbf{h}}_n, \mathbf{x}_n)) &= e(\mathbf{g}, \mathbf{y}_2).\end{aligned}$$

We create commitments $\vec{\mathfrak{b}}_i = \mathsf{GSCom}_\Pi(\mathbf{x}_i; \vec{s_i}) = (\mathbf{x}_i, 1, 1)\vec{\mathfrak{v}}_0^{s_{i,0}}\vec{\mathfrak{v}}_1^{s_{i,1}}\vec{\mathfrak{v}}_2^{s_{i,2}}$, for each component $\mathbf{x}_i$ of $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ using randomness $\vec{s_i} = (s_{i,0}, s_{i,1}, s_{i,2})$. Then we prove that the plaintext of $\mathfrak{c} = \mathsf{CS}\text{-}\mathcal{E}.\mathsf{Enc}^m(\vec{\mathbf{x}}; (r_1, r_2))$ equals the committed values in $\vec{\mathfrak{b}}_i$, for $i = 1, \ldots, n$, by proving that the following system of $3n+4$ one-sided multi-exponentiation equations is satisfiable with a witness $(r_1, r_2, \vec{s_1}, \ldots, \vec{s_n})$:

$$\frac{\vec{\mathfrak{b}}_1}{(\mathbf{c}_4,1,1)} = \vec{\mathfrak{v}}_0^{\,s_{1,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{1,1}} \cdot \vec{\mathfrak{v}}_2^{\,s_{1,2}} \cdot (\mathbf{d}_{11}^{-1}, 1, 1)^{r_1} \cdot (\mathbf{d}_{12}^{-1}, 1, 1)^{r_2} \ ,$$

$$\cdots$$

$$\frac{\vec{\mathfrak{b}}_n}{(\mathbf{c}_{n+3},1,1)} = \vec{\mathfrak{v}}_0^{\,s_{n,0}} \cdot \vec{\mathfrak{v}}_1^{\,s_{n,1}} \cdot \vec{\mathfrak{v}}_2^{\,s_{n,2}} \cdot (\mathbf{d}_{n1}^{-1}, 1, 1)^{r_1} \cdot (\mathbf{d}_{n2}^{-1}, 1, 1)^{r_2} \ ,$$

$$\mathbf{c}_1 = g_0^{r_1} g_0^{r_2} \ , \quad \mathbf{c}_2 = g_1^{r_1} \ , \quad \mathbf{c}_3 = g_2^{r_2} \ , \quad \mathbf{c}_{n+4} = (\mathbf{e}_1 \mathbf{f}_1^t)^{r_1} (\mathbf{e}_2 \mathbf{f}_2^t)^{r_2} \ .$$

The total size of the proof is 18n+66 group elements and 6 $\mathbb{Z}_p$-elements.

Combining the ciphertext and the NIZK argument makes the size of the signature $19n + 70$ group elements and 6 elements in $\mathbb{Z}_p$. By Theorem 14 and Theorem 15, we know that the above instantiation gives us a $((n-2)\log p - 1)$-leakage resilient signature scheme. To translate this into $(1-\epsilon)|sk|$ leakage tolerance, we need

$$n \geq \frac{2}{\epsilon} + \frac{\omega(\log \lambda)}{\epsilon \cdot \log p} = \frac{1}{\epsilon} \cdot \left(2 + \frac{\omega(\log \lambda)}{\log p}\right)$$

This gives us signature of size $(19/\epsilon)(2 + \omega(\log \lambda)/\log p) + 70$ group elements and 6 elements in $\mathbb{Z}_p$.

Chapter 5

Structure-Preserving Signatures

In this chapter we construct structure-preserving signature schemes and consider some of their applications. We start by describing our main scheme in detail and discussing several variations. Next, we extend the scheme to be able to sign messages of unlimited length without losing its structure-preserving properties. Also, we consider the extension of simulatable signatures in the common reference string model. Lastly, we construct a signature scheme with messages composed of group elements from both base groups when working with asymmetric bilinear map and a strongly unforgeable signature scheme.

To illustrate the applicability of such signature schemes, we consider a couple case studies from numerous possible applications. We construct the first efficient round-optimal blind signature scheme. The construction follows the generic framework of Fischlin [55], the efficient instantiation of which has been an open problem. Then, we present an efficient group signature scheme satisfying the strongest security requirements and constructed in a modular fashion.

5.1 Main Scheme

Combining a trapdoor commitment scheme (chameleon hash) and a strong assumption is a well-known approach for designing signature schemes [47, 21]. To bring this idea into a real structure-preserving construction, we need an appropriate trapdoor commitment scheme and a useful assumption which are compatible with each other. As it turns out, TC1 and the SFP assumption are an excellent match for that purpose.

A remaining technical issue is how to deal with the exception that $\mathbf{z}^\star \neq 1$ in SFP. The signature scheme should not inherit it since when proving a knowledge of a signature, the condition $\mathbf{z} \neq 1$ is not trivial to prove and affects the efficiency. We address this issue by involving another set of elements $(\mathbf{a}_0, \tilde{\mathbf{a}}_0)$ and $(\mathbf{b}_0, \tilde{\mathbf{b}}_0)$ in the verification predicate. In the proof of unforgeability, these elements hold a secret random offset $\tilde{\mathbf{g}}^\zeta$ that will be multiplied to $\mathbf{z}$ in a forged signature so that the answer to SFP, $\mathbf{z}^\star = \mathbf{z}\tilde{\mathbf{g}}^\zeta$, happens to be $\mathbf{1}$ only by chance. (The real proof is more involved.)

The randomization techniques from Section 2.5 also help the construction and the security proof in such a way so that the signature elements are uniform under the constraint that the verification predicates hold.

5.1.1 Construction

Let $\vec{\mathbf{m}} = (\mathbf{m}_1, \ldots, \mathbf{m}_k) \in \mathbb{G}_2^k$ be a message to be signed. Parameter k determines the length of a message and shorter messages are implicitly padded with $\mathbf{1}_{\mathbb{G}_2}$. Let $\Lambda \in \{\Lambda_{\mathsf{sym}}, \Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$. We recall that $\Lambda := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, \mathbf{g}, \tilde{\mathbf{g}})$ is an implicit input to the algorithms described below.

- Key Generation. $\mathsf{SIG.Key}(1^\lambda)$: Choose random generators $\mathbf{g}_r, \mathbf{h}_u \leftarrow \mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose $\gamma_i, \delta_i \leftarrow \mathbb{Z}_p^*$ and compute $\mathbf{g}_i = \mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_u^{\delta_i}$. Choose $\gamma_z, \delta_z \leftarrow \mathbb{Z}_p^*$ and compute $\mathbf{g}_z = \mathbf{g}_r^{\gamma_z}$ and $\mathbf{h}_z = \mathbf{h}_u^{\delta_z}$. Also choose

$\alpha, \beta \leftarrow \mathbb{Z}_p^*$ and compute $\{\mathbf{a}_i, \tilde{\mathbf{a}}_i\}_{i=0}^1 \leftarrow \mathsf{RandExtend}(\mathbf{g}_r, \tilde{\mathbf{g}}^\alpha)$ and $\{\mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^1 \leftarrow \mathsf{RandExtend}(\mathbf{h}_u, \tilde{\mathbf{g}}^\beta)$. Set $\mathtt{vk} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k, \{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^1)$ and $\mathtt{sk} = (\mathtt{vk}, \alpha, \beta, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^k)$. Output $(\mathtt{vk}, \mathtt{sk})$.

- Signature Issuing. $\mathsf{SIG.Sign}(\mathtt{sk}, \vec{\mathbf{m}})$: Choose $\zeta, \rho, \tau, \varphi, \omega$ randomly from $\mathbb{Z}_p$ and set:

$$\mathbf{z} = \tilde{\mathbf{g}}^\zeta, \quad \mathbf{r} = \tilde{\mathbf{g}}^{\alpha - \rho\tau - \gamma_z\zeta} \prod_{i=1}^k \mathbf{m}_i^{-\gamma_i}, \quad \mathbf{s} = \mathbf{g}_r^\rho, \quad \mathbf{t} = \tilde{\mathbf{g}}^\tau,$$

$$\mathbf{u} = \tilde{\mathbf{g}}^{\beta - \varphi\omega - \delta_z\zeta} \prod_{i=1}^k \mathbf{m}_i^{-\delta_i}, \quad \mathbf{v} = \mathbf{h}_u^\varphi, \quad \mathbf{w} = \tilde{\mathbf{g}}^\omega.$$

  Output $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ as a signature.

- Verification. $\mathsf{SIG.Vrf}(\mathtt{vk}, \vec{\mathbf{m}}, \sigma)$: Parse σ into $(\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$. Output 1 if

$$\mathbf{A} = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}, \mathbf{t})\ \prod_{i=1}^k \hat{e}(\mathbf{g}_i, \mathbf{m}_i), \text{ and} \tag{5.1}$$

$$\mathbf{B} = \hat{e}(\mathbf{h}_z, \mathbf{z})\ \hat{e}(\mathbf{h}_u, \mathbf{u})\ \hat{e}(\mathbf{v}, \mathbf{w})\ \prod_{i=1}^k \hat{e}(\mathbf{h}_i, \mathbf{m}_i) \tag{5.2}$$

  hold for $\mathbf{A} = \hat{e}(\mathbf{a}_0, \tilde{\mathbf{a}}_0)\ \hat{e}(\mathbf{a}_1, \tilde{\mathbf{a}}_1)$ and $\mathbf{B} = \hat{e}(\mathbf{b}_0, \tilde{\mathbf{b}}_0)\ \hat{e}(\mathbf{b}_1, \tilde{\mathbf{b}}_1)$. Output 0, otherwise.

5.1.2 Security

Theorem 16. SIG *in Section 5.1.1 is correct. It is EUF-CMA if* SFP *holds for* Λ.

Proof. CORRECTNESS. Observe that

$$\hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}, \mathbf{t})\ \prod_{i=1}^k \hat{e}(\mathbf{g}_i, \mathbf{m}_i)\ =$$

$$= \hat{e}\left(\mathbf{g}_r^{\gamma_z}, \tilde{\mathbf{g}}^\zeta\right)\ \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha - \rho\tau - \gamma_z\zeta} \prod_{i=1}^k \mathbf{m}_i^{-\gamma_i}\right)\ \hat{e}\left(\mathbf{g}_r^\rho, \tilde{\mathbf{g}}^\tau\right)\ \prod_{i=1}^k \hat{e}\left(\mathbf{g}_r^{\gamma_i}, \mathbf{m}_i\right) = \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^\alpha\right) = \mathbf{A}$$

holds. Thus (5.1) is fulfilled. Relation (5.2) is verified in the same manner.

UNFORGEABILITY. Let $\mathcal{A}$ be an adversary that has a non-negligible advantage of forging a signature for the above scheme on a message $\vec{\mathbf{m}}^{\dagger}$, $\vec{\mathbf{m}}^{\dagger} \notin \{\vec{\mathbf{m}}_j\}_{j=1}^{q}$, after adaptively querying the signing oracle on messages $\vec{\mathbf{m}}_j$, for $j = 1, \ldots, q$, and receiving signatures σ_j. We construct a reduction algorithm which takes an input Λ, $\mathbf{g}_z$, $\mathbf{h}_z$, $\mathbf{g}_r$, $\mathbf{h}_u$, $(\mathbf{a}, \tilde{\mathbf{a}}), (\mathbf{b}, \tilde{\mathbf{b}})$, and uniformly chosen tuples R_j for $j = 1, \ldots, q$ as defined in Assumption 6, and simulates the view of $\mathcal{A}$ in the attack environment as follows:

- (Simulating $\mathsf{SIG.Key}$) : Use $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u)$ as given in the input. For $i = 1, \ldots, k$ set $\mathbf{g}_i = \mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_z^{\chi_i}\mathbf{h}_u^{\delta_i}$, where $\chi_i, \gamma_i, \delta_i \leftarrow \mathbb{Z}_p$. As the probability that any $\mathbf{g}_i$ or $\mathbf{h}_i$, $i = 1, \ldots, k$, is equal to $\mathbf{1}_{\mathbb{G}_1}$ is negligible, the reduction algorithm simply aborts in such cases. Otherwise, all group elements are from $\mathbb{G}_1^*$ and chosen uniformly at random, like in the key generation algorithm. Then select ζ, ρ, $\varphi \leftarrow \mathbb{Z}_p$, and compute $((\mathbf{a}_0, \tilde{\mathbf{a}}_0), (\mathbf{a}_1, \tilde{\mathbf{a}}_1)) \leftarrow \mathsf{RandSeq}((\mathbf{g}_z^{\zeta}\mathbf{g}_r^{\rho}, \tilde{\mathbf{g}}), (\mathbf{a}, \tilde{\mathbf{a}}))$ and $((\mathbf{b}_0, \tilde{\mathbf{b}}_0), (\mathbf{b}_1, \tilde{\mathbf{b}}_1)) \leftarrow \mathsf{RandSeq}((\mathbf{h}_z^{\zeta}\mathbf{h}_u^{\varphi}, \tilde{\mathbf{g}}), (\mathbf{b}, \tilde{\mathbf{b}}))$. For convenience, denote $\mathbf{g}_z^{\zeta}\mathbf{g}_r^{\rho}$ with $\mathbf{a}'$ and $\mathbf{h}_z^{\zeta}\mathbf{h}_u^{\varphi}$ with $\mathbf{b}'$. The verification key is $\mathtt{vk} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^{k}, \{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^{1})$.

- (Simulating $\mathsf{SIG.Sign}$) : Given message $\vec{\mathbf{m}}$, take a fresh tuple $R_j = (\mathbf{z}_j, \mathbf{r}_j, \mathbf{s}_j, \mathbf{t}_j, \mathbf{u}_j, \mathbf{v}_j, \mathbf{w}_j)$ from the input instance. Then compute

$$\mathbf{z} = \mathbf{z}_j \, \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k} \mathbf{m}_i^{-\chi_i}, \quad \mathbf{r} = \mathbf{r}_j \, \tilde{\mathbf{g}}^{\rho} \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}, \quad \mathbf{s} = \mathbf{s}_j, \quad \mathbf{t} = \mathbf{t}_j,$$
$$\mathbf{u} = \mathbf{u}_j \, \tilde{\mathbf{g}}^{\varphi} \prod_{i=1}^{k} \mathbf{m}_i^{-\delta_i}, \quad \mathbf{v} = \mathbf{v}_j, \quad \mathbf{w} = \mathbf{w}_j.$$

  The signature is $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$. It is easy to verify that the signature satisfies the verification equations.

When $\mathcal{A}$ outputs $(\vec{\mathbf{m}}^\dagger, (\mathbf{z}^\dagger, \mathbf{r}^\dagger, \mathbf{s}^\dagger, \mathbf{t}^\dagger, \mathbf{u}^\dagger, \mathbf{v}^\dagger, \mathbf{w}^\dagger))$, compute

$$\mathbf{z}^\star = \mathbf{z}^\dagger \, \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\chi_i}, \quad \mathbf{r}^\star = \mathbf{r}^\dagger \, \tilde{\mathbf{g}}^{-\rho} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\gamma_i}, \quad \mathbf{u}^\star = \mathbf{u}^\dagger \, \tilde{\mathbf{g}}^{-\varphi} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\delta_i},$$

and set $\mathbf{s}^\star = \mathbf{s}^\dagger$, $\mathbf{t}^\star = \mathbf{t}^\dagger$, $\mathbf{v}^\star = \mathbf{v}^\dagger$, and $\mathbf{w}^\star = \mathbf{w}^\dagger$. If any of the parameters $\chi_1, \ldots, \chi_k$ is 0 the reduction algorithm aborts; otherwise, outputs $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{s}^\star, \mathbf{t}^\star, \mathbf{u}^\star, \mathbf{v}^\star, \mathbf{w}^\star)$. Like in the previous abort case, the chance for that is negligible because the parameters are chosen uniformly at random, and, therefore, we could ignore those cases in our analysis without affecting the overall outcome. This completes the description of the reduction algorithm.

The above signatures follow correct distribution. So, $\mathcal{A}$ outputs a successful forgery with a non-negligible probability. Then, for the output of the reduction algorithm, it holds that

$$\begin{aligned}
&\hat{e}(\mathbf{g}_z, \mathbf{z}^\star) \; \hat{e}(\mathbf{g}_r, \mathbf{r}^\star) \; \hat{e}(\mathbf{s}^\star, \mathbf{t}^\star) \\
&= \hat{e}\left(\mathbf{g}_z, \mathbf{z}^\dagger \, \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\chi_i}\right) \; \hat{e}\left(\mathbf{g}_r, \mathbf{r}^\dagger \, \tilde{\mathbf{g}}^{-\rho} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\gamma_i}\right) \; \hat{e}\left(\mathbf{s}^\dagger, \mathbf{t}^\dagger\right) \\
&= \hat{e}\left(\mathbf{g}_z^{-\zeta}\mathbf{g}_r^{-\rho}, \tilde{\mathbf{g}}\right) \; \hat{e}\left(\mathbf{g}_z, \mathbf{z}^\dagger\right) \; \hat{e}\left(\mathbf{g}_r, \mathbf{r}^\dagger\right) \; \hat{e}\left(\mathbf{s}^\dagger, \mathbf{t}^\dagger\right) \; \prod_{i=1}^{k} \hat{e}\left(\mathbf{g}_i, \mathbf{m}_i^\dagger\right) \\
&= \hat{e}\left(\mathbf{g}_z^{\zeta}\mathbf{g}_r^{\rho}, \tilde{\mathbf{g}}\right)^{-1} \; \prod_{i=0}^{1} \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i) = \hat{e}(\mathbf{a}, \tilde{\mathbf{a}}).
\end{aligned}$$

One can also verify that $\hat{e}(\mathbf{g}_z, \mathbf{z}^\star) \; \hat{e}(\mathbf{h}_u, \mathbf{u}^\star) \; \hat{e}(\mathbf{v}^\star, \mathbf{w}^\star) = \hat{e}(\mathbf{b}, \tilde{\mathbf{b}})$ holds in the same way.

What remains is to show that $\mathbf{z}^\star$ is not in $\{1, \mathbf{z}_1, \ldots, \mathbf{z}_q\}$. For that, first notice that the parameters ζ and $\{\chi_i\}_{i=1}^k$ are independent from the view of adversary $\mathcal{A}$, as proved in Lemma 3. Namely, for any view of the adversary and for any choice of ζ and χ_i, for $i = 1, \ldots, k$, there exist unique and consistent parameters ρ, φ, γ_i, δ_i, $i = 1, \ldots, k$ and $\mathbf{z}_j$, $\mathbf{r}_j$, $\mathbf{u}_j$, $j = 1, \ldots, q$.

First we show that the probability $\mathbf{z}^\star \in \{\mathbf{z}_1, \ldots, \mathbf{z}_q\}$ is negligible. For every $\mathbf{z}_j$ and signature $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ on a message $\vec{\mathbf{m}}$ simulated by using $\mathbf{z}_j$, it holds

that

$$\frac{\mathbf{z}^\star}{\mathbf{z}_j} \quad = \quad \frac{\mathbf{z}^\dagger \; \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\chi_i}}{\mathbf{z} \; \tilde{\mathbf{g}}^{-\zeta} \; \prod_{i=1}^{k} \mathbf{m}_i^{\chi_i}} \quad = \quad \frac{\mathbf{z}^\dagger}{\mathbf{z}} \prod_{i=1}^{k} \left(\frac{\mathbf{m}_i^\dagger}{\mathbf{m}_i}\right)^{\chi_i}.$$

Since $\vec{\mathbf{m}}^\dagger \neq \vec{\mathbf{m}}$, there exists i such that $\mathbf{m}_i^\dagger \neq \mathbf{m}_i$. Since $\chi_i \in \mathbb{Z}_p^*$ is information theoretically hidden from the view of the adversary, the probability that $\mathbf{z}^\star = \mathbf{z}_j$ is negligible due to the term $(\mathbf{m}_i^\dagger/\mathbf{m}_i)^{\chi_i}$ in the above equation. To show that $\mathbf{z}^\star = \left(\mathbf{z}^\dagger\right) \; \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^{k} \left(\mathbf{m}_i^\dagger\right)^{\chi_i}$ is equal to $\mathbf{1}_{\mathbb{G}_2}$ only with a negligible probability, notice that ζ is also independent from the view of the adversary and the claim holds due to the uniform choice of ζ. Therefore, when the reduction algorithm does not abort, the probability that $\mathbf{z}^\star \notin \{1, \mathbf{z}_1, \ldots, \mathbf{z}_q\}$ is overwhelming. □

Lemma 3. *The parameters ζ, χ_1, χ_2, …, χ_k chosen by the reduction algorithm in Theorem 16 are independent from $\mathcal{A}$'s view. That is, independent from the verification key, the signed messages, and the signatures.*

Proof. Let $\mathtt{vk} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^{k}, \{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^{1})$ be the verification key the adversary sees, $\vec{\mathbf{m}}_1$, …, $\vec{\mathbf{m}}_q$ be the messages with which $\mathcal{A}$ queries the signing oracle, and σ_1, …, σ_q be the corresponding signatures. Furthermore, let assume that $(\mathbf{a}, \tilde{\mathbf{a}})$ and $(\mathbf{b}, \tilde{\mathbf{b}})$ given to the reduction algorithm are also fixed, though $\mathcal{A}$ does not see them. That yields unique $\mathbf{a}'$ and $\mathbf{b}'$ such that

$$\begin{aligned} \mathbf{A} &= \hat{e}(\mathbf{a}_0, \tilde{\mathbf{a}}_0)\hat{e}(\mathbf{a}_1, \tilde{\mathbf{a}}_1) = \hat{e}(\mathbf{a}', \tilde{\mathbf{g}})\hat{e}(\mathbf{a}, \tilde{\mathbf{a}}) \quad \text{and} \\ \mathbf{B} &= \hat{e}(\mathbf{b}_0, \tilde{\mathbf{b}}_0)\hat{e}(\mathbf{b}_1, \tilde{\mathbf{b}}_1) = \hat{e}(\mathbf{b}', \tilde{\mathbf{g}})\hat{e}(\mathbf{b}, \tilde{\mathbf{b}}). \end{aligned}$$

For any choice $\hat{\zeta}, \hat{\chi}_i \in \mathbb{Z}_p$ of the parameters ζ, χ_i , for $i = 1, \ldots, k$, there exist a *unique* coin toss $\hat{\rho}$, $\hat{\varphi}$, $\hat{\gamma}_i$, $\hat{\delta}_i$ such that $\mathbf{a}' = \mathbf{g}_z^{\hat{\zeta}} \mathbf{g}_r^{\hat{\rho}}$, $\mathbf{b}' = \mathbf{h}_z^{\hat{\zeta}} \mathbf{h}_u^{\hat{\varphi}}$, $\mathbf{g}_i = \mathbf{g}_z^{\hat{\chi}_i} \mathbf{g}_r^{\hat{\gamma}_i}$, and $\mathbf{h}_i = \mathbf{h}_z^{\hat{\chi}_i} \mathbf{h}_u^{\hat{\delta}_i}$. This shows that the verification key and the parameters are independent. Next we show that the chosen parameters remain independent from $\mathcal{A}$'s view even after signing

q adaptively chosen messages due to the uniform choice of the tuples R_j, $j = 1, \ldots, q$, as defined in Assumption 6.

Let the j-th message be $\vec{\mathbf{m}}$ and the corresponding signature be $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$. From the specification of the reduction algorithm we know that $(\mathbf{s}, \mathbf{t}) = (\mathbf{s}_j, \mathbf{t}_j)$ and $(\mathbf{v}, \mathbf{w}) = (\mathbf{v}_j, \mathbf{w}_j)$, where $R_j = (\mathbf{z}_j, \mathbf{r}_j, \mathbf{s}_j, \mathbf{t}_j, \mathbf{u}_j, \mathbf{v}_j, \mathbf{w}_j)$ is the j-th tuple given as input. And for the fixed view, $\zeta, \{\chi_i\}_{i=1}^k$ determine *uniquely* the values of $\mathbf{z}_j = \mathbf{z}\ \tilde{\mathbf{g}}^{-\zeta}\ \prod_{i=1}^k \mathbf{m}_i^{\chi_i}$, $\mathbf{r}_j = \mathbf{r}\ \tilde{\mathbf{g}}^{-\rho}\ \prod_{i=1}^k \mathbf{m}_i^{\gamma_i}$, and $\mathbf{u}_j = \mathbf{u}\ \tilde{\mathbf{g}}^{-\varphi}\ \prod_{i=1}^k \mathbf{m}_i^{\delta_i}$. Regardless of the particular choice of parameters $\hat{\zeta}, \{\hat{\chi}_i\}_{i=1}^k$, since σ satisfies the signature verification equations

$$\mathbf{A} = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}, \mathbf{t})\ \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i) \quad \text{and}$$

$$\mathbf{B} = \hat{e}(\mathbf{h}_z, \mathbf{z})\ \hat{e}(\mathbf{h}_u, \mathbf{u})\ \hat{e}(\mathbf{v}, \mathbf{w})\ \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i),$$

it is true that the corresponding tuple $\hat{R}_j = (\hat{\mathbf{z}}_j, \hat{\mathbf{r}}_j, \mathbf{s}_j, \mathbf{t}_j, \hat{\mathbf{u}}_j, \mathbf{v}_j, \mathbf{w}_j)$ satisfies

$$\hat{e}(\mathbf{a}, \tilde{\mathbf{a}}) = \hat{e}(\mathbf{g}_z, \hat{\mathbf{z}}_j)\ \hat{e}(\mathbf{g}_r, \hat{\mathbf{r}}_j)\ \hat{e}(\mathbf{s}_j, \mathbf{t}_j) \quad \text{and} \tag{5.3}$$

$$\hat{e}(\mathbf{b}, \tilde{\mathbf{b}}) = \hat{e}(\mathbf{h}_z, \hat{\mathbf{z}}_j)\ \hat{e}(\mathbf{h}_u, \hat{\mathbf{u}}_j)\ \hat{e}(\mathbf{v}_j, \mathbf{w}_j). \tag{5.4}$$

What remains to show is that the uniform choice of $\hat{\zeta}, \{\hat{\chi}_i\}_{i=1}^k$ together with $\mathcal{A}$'s view yields uniform distribution for the tuples $\hat{R}_j$, for $j = 1, \ldots, q$, as specified by the assumption description. If that is indeed the case, each set of tuples which could have been given as input to the reduction algorithm is chosen with the same probability. And because for any choice of $\hat{\zeta}, \hat{\chi}_1, \ldots, \chi_k$, there exist *unique* set $\{\hat{R}_j\}_{j=1}^q$, those imply that each parameter selection looks equally likely for $\mathcal{A}$.

To see the uniformity of $\hat{R}_j$, note again that $(\mathbf{s}_j, \mathbf{t}_j)$ and $(\mathbf{v}_j, \mathbf{w}_j)$ are determined uniquely from the view regardless of the parameters choice. Then, let's define the

homomorphism ϕ:

$$\phi_{\tilde{\mathbf{g}},\vec{\mathbf{m}}}(\hat{\zeta},\hat{\chi}_1,\ldots,\hat{\chi}_k) = \tilde{\mathbf{g}}^{-\hat{\zeta}} \prod_{i=1}^{k} \mathbf{m}_i^{\hat{\chi}_i}.$$

It is easy to verify that for uniformly chosen parameters, the homomorphism's range is uniformly distributed over $\mathbb{G}_2$. This in turn implies that for a fixed $\mathbf{z}$ and uniformly chosen parameters, $\hat{\mathbf{z}}_j = \mathbf{z}\ \phi(\hat{\zeta},\hat{\chi}_1,\ldots,\hat{\chi}_k)$ is uniformly distributed over $\mathbb{G}_2$. And because $\hat{R}_j$ satisfies (5.3) – (5.4), the values of $\hat{\mathbf{r}}_j$ and $\hat{\mathbf{u}}_j$ are determined uniquely by the other tuple values, which for a fixed view means determined by $\hat{\mathbf{z}}_j$. To sum it up, for a fixed view, the uniform random choice of the parameters gives uniformly distributed $\hat{\mathbf{z}}_j$ which implies the uniformity of $\hat{R}_j$.

□

5.1.3 Notable Properties

Partial Perfect Randomizability. Given a signature $(\mathbf{z},\mathbf{r},\mathbf{s},\mathbf{t},\mathbf{u},\mathbf{v},\mathbf{w})$ one can randomize every element except for $\mathbf{z}$ by applying the sequential randomization technique with a small tweak as follows. Define the function $\mathsf{SigRand}$, $(\mathbf{r}',\mathbf{s}',\mathbf{t}',\mathbf{u}',\mathbf{v}',\mathbf{w}') \leftarrow \mathsf{SigRand}(\mathbf{r},\mathbf{s},\mathbf{t},\mathbf{u},\mathbf{v},\mathbf{w})$, as:

- Randomize $(\mathbf{r},\mathbf{s},\mathbf{t})$ into $(\mathbf{r}',\mathbf{s}',\mathbf{t}')$ as follows.
  - First, if $\mathbf{t} = 1$, set $\mathbf{s} = 1$ and choose $\mathbf{t} \leftarrow \mathbb{G}_2^*$.
  - Then, choose $\varrho \leftarrow \mathbb{Z}_p$ and compute

$$\mathbf{r}' = \mathbf{r}\,\mathbf{t}^{\varrho}, \quad (\mathbf{s}',\mathbf{t}') \leftarrow \mathsf{Rand}(\mathbf{sg}_r^{-\varrho},\mathbf{t}) \tag{5.5}$$

- Randomize $(\mathbf{u},\mathbf{v},\mathbf{w})$ into $(\mathbf{u}',\mathbf{v}',\mathbf{w}')$ analogously.

Lemma 4. *The above* $(\mathbf{r}',\mathbf{s}',\mathbf{t}',\mathbf{u}',\mathbf{v}',\mathbf{w}')$ *distributes uniformly over* $(\mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2)^2$ *under the constraint that* $\hat{e}(\mathbf{g}_r,\mathbf{r})\,\hat{e}(\mathbf{s},\mathbf{t}) = \hat{e}(\mathbf{g}_r,\mathbf{r}')\,\hat{e}(\mathbf{s}',\mathbf{t}')$ *and* $\hat{e}(\mathbf{h}_u,\mathbf{u})\,\hat{e}(\mathbf{v},\mathbf{w}) = \hat{e}(\mathbf{h}_u,\mathbf{u}')\,\hat{e}(\mathbf{v}',\mathbf{w}')$.

Proof. Uniformity of $\mathbf{r}' \in \mathbb{G}_2$ follows from $\mathbf{t} \neq 1$ and the uniformity of ϱ in (5.5). Under the described constraints, for any choice of $\mathbf{r}'$, their is a *unique* value $\hat{e}(\mathbf{s}', \mathbf{t}') = \hat{e}(\mathbf{g}_r, \mathbf{r})\,\hat{e}(\mathbf{s}, \mathbf{t})\,\hat{e}(\mathbf{g}_r, \mathbf{r}')^{-1}$. Then, uniformity of $\mathbf{s}'$ and $\mathbf{t}'$ holds from the property of Rand. The same is true for $(\mathbf{u}', \mathbf{v}', \mathbf{w}')$. □

The claim implies that $(\mathbf{s}', \mathbf{t}', \mathbf{v}', \mathbf{w}')$ is information theoretically independent of the signature element $\mathbf{z}$, the message, and the verification key. (In general, the same is true for publishing any two elements from $(\mathbf{r}', \mathbf{s}', \mathbf{t}')$ and $(\mathbf{u}', \mathbf{v}', \mathbf{w}')$ respectively.) This property is useful in reducing the task of combined proofs. See Section 5.6.1 for typical use of this property.

Signature Binding Property. This property states that no one but the signer can obtain two signatures which have the same $\mathbf{s}$ and $\mathbf{v}$. In the following formal statement, the adversary is allowed to submit both $\vec{\mathbf{m}}$ and $\vec{\mathbf{m}}'$ to the signing oracle. Hence the property is not implied by EUF-CMA in general.

Lemma 5. *Under adaptive chosen message attacks, no adversary can output* $(\vec{\mathbf{m}}, \sigma)$ *and* $(\vec{\mathbf{m}}', \sigma')$ *such that* $1 = \mathsf{SIG.Vrf}(\mathtt{vk}, \vec{\mathbf{m}}, \sigma) = \mathsf{SIG.Vrf}(\mathtt{vk}, \vec{\mathbf{m}}', \sigma')$, $\vec{\mathbf{m}} \neq \vec{\mathbf{m}}'$, *and* $(\mathbf{s}, \mathbf{v})$ *are shared in* σ *and* σ'.

Lemma 5 implies that publishing $(\mathbf{s}, \mathbf{v})$ together with the verification key works as a commitment of the signature and the message. (Recall that $\mathbf{s}$ and $\mathbf{v}$ are uniformly chosen in the signature generation algorithm.) This property is used in Section 5.2, and could find more applications.

Proof. Suppose that there is a successful adversary, $\mathcal{A}$ that outputs the signatures as in the lemma. We then construct an adversary $\mathcal{B}$ that breaks EUF-CMA of SIG.

Given $\mathtt{vk}$ and oracle access to $\mathcal{O}_{\mathsf{sign}}(\cdot)$, $\mathcal{B}$ invokes $\mathcal{A}$ with $\mathtt{vk}$. Every signing query from $\mathcal{A}$ is directly passed to $\mathcal{O}_{\mathsf{sign}}(\cdot)$ and the signatures are returned di-

rectly to $\mathcal{A}$. Hence $\mathcal{B}$'s simulation is perfect. Eventually, $\mathcal{A}$ terminates and outputs $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$, $\vec{\mathbf{m}} = (\mathbf{m}_1, \ldots, \mathbf{m}_k)$, $\sigma' = (\mathbf{z}', \mathbf{r}', \mathbf{s}, \mathbf{t}', \mathbf{u}', \mathbf{v}, \mathbf{w}')$, and $\vec{\mathbf{m}}' = (\mathbf{m}'_1, \ldots, \mathbf{m}'_k)$.

$\mathcal{B}$ then chooses $\varrho \leftarrow \mathbb{Z}_p$ and computes

$$\mathbf{z}^\star = \mathbf{z}\left(\tfrac{\mathbf{z}'}{\mathbf{z}}\right)^{\varrho}, \quad \mathbf{r}^\star = \mathbf{r}\left(\tfrac{\mathbf{r}'}{\mathbf{r}}\right)^{\varrho}, \quad \mathbf{t}^\star = \mathbf{t}\left(\tfrac{\mathbf{t}'}{\mathbf{t}}\right)^{\varrho},$$
$$\mathbf{u}^\star = \mathbf{u}\left(\tfrac{\mathbf{u}'}{\mathbf{u}}\right)^{\varrho}, \quad \mathbf{w}^\star = \mathbf{w}\left(\tfrac{\mathbf{w}'}{\mathbf{w}}\right)^{\varrho}, \quad \mathbf{m}_i^\star = \mathbf{m}_i\left(\tfrac{\mathbf{m}'_i}{\mathbf{m}_i}\right)^{\varrho}.$$

Then outputs $\sigma^\star = (\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{s}, \mathbf{t}^\star, \mathbf{u}^\star, \mathbf{v}, \mathbf{w}^\star)$ and $\vec{\mathbf{m}}^\star = (\mathbf{m}_1^\star, \ldots, \mathbf{m}_k^\star)$. This completes the specification of $\mathcal{B}$.

We verify the correctness of $\mathcal{B}$ as follows. Since these signatures are valid, they satisfy

$$\begin{aligned}\mathbf{A} &= \hat{e}(\mathbf{g}_z, \mathbf{z}')\ \hat{e}(\mathbf{g}_r, \mathbf{r}')\ \hat{e}(\mathbf{s}, \mathbf{t}')\ \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}'_i) \\ &= \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}, \mathbf{t})\ \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i)\end{aligned}$$

and

$$\begin{aligned}\mathbf{B} &= \hat{e}(\mathbf{h}_z, \mathbf{z}')\ \hat{e}(\mathbf{h}_u, \mathbf{u}')\ \hat{e}(\mathbf{v}, \mathbf{w}')\ \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}'_i) \\ &= \hat{e}(\mathbf{h}_z, \mathbf{z})\ \hat{e}(\mathbf{h}_u, \mathbf{u})\ \hat{e}(\mathbf{v}, \mathbf{w})\ \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i).\end{aligned}$$

Note that we can divide the verification equations of the signatures which gives us:

$$\mathbf{1}_{\mathbb{G}_T} = e\left(\mathbf{g}_z, \frac{\mathbf{z}'}{\mathbf{z}}\right)\ e\left(\mathbf{g}_r, \frac{\mathbf{r}'}{\mathbf{r}}\right)\ e\left(\mathbf{s}, \frac{\mathbf{t}'}{\mathbf{t}}\right)\ \prod_{i=1}^{k} e\left(\mathbf{g}_i, \frac{\mathbf{m}'_i}{\mathbf{m}_i}\right), \text{ and}$$
$$\mathbf{1}_{\mathbb{G}_T} = e\left(\mathbf{h}_z, \frac{\mathbf{z}'}{\mathbf{z}}\right)\ e\left(\mathbf{h}_u, \frac{\mathbf{u}'}{\mathbf{u}}\right)\ e\left(\mathbf{v}, \frac{\mathbf{w}'}{\mathbf{w}}\right)\ \prod_{i=1}^{k} e\left(\mathbf{h}_i, \frac{\mathbf{m}'_i}{\mathbf{m}_i}\right).$$

Exponentiating these equations with ϱ and multiplying them with one of the signatures yields $\sigma^\star = (\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{s}, \mathbf{t}^\star, \mathbf{u}^\star, \mathbf{v}, \mathbf{w}^\star)$ and $\vec{\mathbf{m}}^\star = (\mathbf{m}_1^\star, \ldots, \mathbf{m}_k^\star)$ which clearly satisfy the verification equations.

Since $\vec{\mathbf{m}}' \neq \vec{\mathbf{m}}$ there exists j such that $\mathbf{m}'_j \neq \mathbf{m}_j$. And due to the random choice of ϱ, $\mathbf{m}^\star_j = \mathbf{m}_j \left(\frac{\mathbf{m}'_j}{\mathbf{m}_i}\right)^\varrho$ distributes uniformly over $\mathbb{G}_2$. Accordingly, $\vec{\mathbf{m}}^\star$ is different from any message vector observed by $\mathcal{O}_{\mathsf{sign}}(\cdot)$ with overwhelming probability. Thus, $(\sigma^\star, \vec{\mathbf{m}}^\star)$ is a valid forgery to SIG. □

5.1.4 Variations

- We can replace $\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i$ with $\mathbf{A} = \hat{e}(\mathbf{g}_r, \tilde{\mathbf{g}}^\alpha)$ and $\mathbf{B} = \hat{e}(\mathbf{h}_u, \tilde{\mathbf{g}}^\beta)$ in a verification-key, and use the $\mathbf{A}$ and $\mathbf{B}$ directly in the verification equations (5.1) and (5.2). The reason we include a representation of $\mathbf{A}$ (and $\mathbf{B}$) in $\mathbb{G}_1$ and $\mathbb{G}_2$ is to address the needs to put the verification key into the base groups. The GS-proof system provides zero-knowledge property for statements that do not include elements from $\mathbb{G}_T$ except for $\mathbf{1}_{\mathbb{G}_T}$. When WI is of only concern, one can include $\mathbf{A}$ and $\mathbf{B}$ in $\mathtt{vk}$ and use them directly in the verification. We use this modification in Section 5.6.1. The same is possible for other constructions in this chapter.

- Let $\langle n \rangle$ denote a deterministic encoding of non-negative integer n, $n < p$, to an element of $\mathbb{G}_2^*$. By limiting the maximum message length to be $k-1$ and putting $\langle |\vec{\mathbf{m}}| \rangle$ at the beginning of the input message $\vec{\mathbf{m}}$, shorter messages can be treated. Since the encoding is deterministic and black-box that is independent of the representation of the elements in $\vec{\mathbf{m}}$, it does not impact the compatibility.

- As we observed in the very last stage of the security proof, $(\mathbf{a}_0, \tilde{\mathbf{a}}_0)$ and $(\mathbf{b}_0, \tilde{\mathbf{b}}_0)$ in a verification key is needed to handle the case where $\mathbf{z}^\dagger = 1$ and $\vec{\mathbf{m}}^\dagger = (\mathbf{1}, \ldots, \mathbf{1})$ happen at the same time. When such exception is not possible, for example when $\vec{\mathbf{m}}$ is encoded with with its length as $\vec{\mathbf{m}}^\dagger = (\langle n \rangle, \mathbf{1}, \ldots, \mathbf{1})$ and the deterministic encoding $\langle n \rangle$ is never 1, the elements $(\mathbf{a}_0, \tilde{\mathbf{a}}_0)$ and $(\mathbf{b}_0, \tilde{\mathbf{b}}_0)$ can

be removed from the scheme.

- In the asymmetric settings, one can swap $\mathbb{G}_1$ and $\mathbb{G}_2$ in the description of SIG to get the 'dual' scheme of SIG whose message space is $\mathbb{G}_1^k$.

- Dropping the flexible part $\hat{e}(\mathbf{s}, \mathbf{t})$ and $\hat{e}(\mathbf{v}, \mathbf{w})$ from the construction results in a strongly unforgeable one-time signature scheme based on the SDP assumption as described in Section 4.2.

5.2 Signing Unbounded-Size Messages

5.2.1 Overview

This section presents a method to sign a message $(\mathbf{m}_1, \ldots, \mathbf{m}_n)$ whose size n is not a-priori bounded by the public-key. While some generic domain extension methods are available, we present a specific and efficient construction based on a *chain of signatures* taking the advantage of the constant-size signature scheme from Section 5.1. The idea is that, first sign $\mathbf{m}_1$ to obtain σ_1, and next sign $\sigma_1||\mathbf{m}_2$ to obtain σ_2, then sign $\sigma_2||\mathbf{m}_3$ and so on. (Note that this rough description lacks some important details. In particular, signing only on $\mathbf{m}_1$ at the beginning results in an insecure scheme.)

A technical highlight is that, with our constant-size signature scheme, we only need to involve a part of a signature, elements $\mathbf{s}$ and $\mathbf{v}$, in each step of chaining to constitute a secure chain. This is possible due to the signature binding property of SIG as shown in Section 5.1.3.

5.2.2 Construction

Let SIG be the constant-size signature scheme from Section 5.1, whose message space is $\mathbb{G}_2^k$ for $k \geq 3$. We construct an unbounded-message signature scheme, $\mathsf{USIG1}$, as follows. Let $\Lambda = \Lambda_{\mathsf{sym}}$ be implicitly given to the functions described below. Recall that $\langle n \rangle$ is an encoding of n to an element of $\mathbb{G}_2^*$.

- $\mathsf{USIG1.Key}(1^\lambda)$: Choose random $(\mathbf{s}_{-1}, \mathbf{v}_{-1}) \leftarrow \mathbb{G}_1^2$. Invoke $(\mathtt{vk}', \mathtt{sk}) \leftarrow \mathsf{SIG.Key}(1^\lambda)$. Output $vk = (\mathtt{vk}', \mathbf{s}_{-1}, \mathbf{v}_{-1})$ and $\mathtt{sk}$.

- $\mathsf{USIG1.Sign}(\mathtt{sk}, \vec{\mathbf{m}})$: Parse $\vec{\mathbf{m}}$ into $(\mathbf{m}_1, \ldots, \mathbf{m}_n)$. Let $\ell = \lceil \frac{n+1}{k-2} \rceil$. Let $\mathbf{m}_0 = \langle n \rangle$ and $\mathbf{m}_i = \mathbf{1}_{\mathbb{G}_2}$ for $i = n+1, \ldots, \ell(k-2)$. For $i = 0, \ldots, \ell - 1$, compute the signature $\sigma_i = (\mathbf{z}_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{t}_i, \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i) \leftarrow \mathsf{SIG.Sign}(\mathtt{sk}, \vec{\mathbf{m}}_i)$ where $\vec{\mathbf{m}}_i = (\mathbf{s}_{i-1}, \mathbf{v}_{i-1}, \mathbf{m}_{i(k-2)}, \ldots, \mathbf{m}_{(i+1)(k-2)-1})$. Output $\sigma = (\sigma_0, \ldots, \sigma_{\ell-1})$.

- $\mathsf{USIG1.Vrf}(vk, \vec{\mathbf{m}}, \sigma)$: Parse σ into $(\sigma_0, \ldots, \sigma_{\ell-1})$ and $\vec{\mathbf{m}}$ into $(\mathbf{m}_1, \ldots, \mathbf{m}_n)$. Let $\mathbf{m}_0 = \langle n \rangle$ and $\mathbf{m}_i = \mathbf{1}_{\mathbb{G}_2}$ for $i = n+1, \ldots, \ell(k-2)$. For $i = 0, \ldots, \ell - 1$, compute $b_i = \mathsf{SIG.Vrf}(\mathtt{vk}', \vec{\mathbf{m}}_i, \sigma_i)$ where $\vec{\mathbf{m}}_i$ is formed in the same way as in $\mathsf{SIG.Sign}$. Output 1 if $b_i = 1$ for all $i = 0, \ldots, \ell - 1$. Output 0, otherwise.

The resulting signature is in the size of $7 \cdot \lceil \frac{n+1}{k-2} \rceil$.

Remarks. Filling $\mathbf{1}_{\mathbb{G}}$ to the empty slots of the message space is for notational consistency. It does not increase either computation or storage. Setting $\Lambda = \Lambda_{\mathsf{sym}}$ is needed as $(\mathbf{s}_i, \mathbf{v}_i)$ is in $\mathbb{G}_1^2$ while the message space is $\mathbb{G}_2^k$. It can be modified for the case of $\Lambda = \Lambda_{\mathsf{sxdh}}$ using the signature scheme described in Section 5.4 (but not for the case of $\Lambda = \Lambda_{\mathsf{xdh}}$). If $\vec{\mathbf{m}}$ is given as an on-line stream and the length is not known in advance, one can use the trapdoor commitment scheme $\mathsf{TC4}$ from Section 4.1 so that $\mathbf{m}_0$ is set to a random commitment and later opened to n when n is fixed. The opening information is included as a part of a signature. Since the opening information is a group

element and the commitment verification predicate is a pairing product equation, the resulting verification predicate for USIG1 remains as a conjunction of pairing product equations.

Theorem 17. *If* SIG *is EUF-CMA, so is USIG1.*

Proof. Suppose that there is a successful adversary, say $\mathcal{A}$, that launches chosen message attacks and outputs a valid forgery, $((\mathbf{m}_1^\dagger, \ldots, \mathbf{m}_n^\dagger), (\sigma_0^\dagger, \ldots, \sigma_{\ell-1}^\dagger))$. Let $\vec{\mathbf{m}}_i^\dagger$ be the message vector associated to $\sigma_i^\dagger$. We then have two cases.

Type-I. There is $\vec{\mathbf{m}}_i^\dagger$ that has never been signed by the signing oracle.

Type-II. Every $\vec{\mathbf{m}}_i^\dagger$ has been signed by the signing oracle (in separate queries).

Type-I forgery trivially breaks the unforgeability of SIG. For Type-II forgery, we show a reduction to the unforgeability of SIG as follows. Given verification key $\mathtt{vk}'$ of SIG and access to the signing oracle of SIG, we construct a simulator that uses adversary $\mathcal{A}$ and simulates USIG1 as follows. Let $\mathcal{O}_{\mathsf{sign}}(\cdot)$ be the signing oracle of SIG with respect to $\mathtt{vk}'$.

- (Simulating USIG1.Key): Generate a random message vector $\vec{\mathbf{m}}_{-1}$ of size k and send it to $\mathcal{O}_{\mathsf{sign}}(\cdot)$. Receive signature $(\mathbf{z}_{-1}, \mathbf{r}_{-1}, \mathbf{s}_{-1}, \mathbf{t}_{-1}, \mathbf{u}_{-1}, \mathbf{v}_{-1}, \mathbf{w}_{-1})$ and output $vk = (\mathtt{vk}', \mathbf{s}_{-1}, \mathbf{v}_{-1})$.
- (Simulating USIG1.Sign): On input $\vec{\mathbf{m}}$, follow the legitimate signing algorithm by asking $\mathcal{O}_{\mathsf{sign}}(\cdot)$ to compute SIG.Sign. Then output the resulting signature.

Observe that $\mathbf{s}_{-1}$ and $\mathbf{v}_{-1}$ generated in the simulated USIG1.Key are uniform and independent of $\vec{\mathbf{m}}_{-1}$. Simulation for USIG1.Sign is clearly perfect as it follows the legitimate procedure.

Suppose that adversary $\mathcal{A}$ outputs a valid forgery for $\mathsf{USIG1}$. Then there exists a signing query (to the signing oracle of $\mathsf{USIG1}$) in which $\vec{\mathbf{m}}^{\dagger}_{\ell-1}$ is observed. Let $((\mathbf{m}_1, \ldots, \mathbf{m}_{n'}), (\sigma_0, \ldots, \sigma_{\ell'-1}))$ be the message and the signature with respect to the query and let $\vec{\mathbf{m}}_i$ denote a message vector associated to σ_i. Let $i^\star$ be the index where $\vec{\mathbf{m}}^{\dagger}_{\ell-1} = \vec{\mathbf{m}}_{i^\star}$ happens. If $\ell - 1 = 0$, then $i^\star = 0$ is not the case because the message in the valid forgery must be fresh. In the case of $\ell - 1 \neq 0$ and $i^\star = 0$, it happens that $\vec{\mathbf{m}}^{\dagger}_{\ell-2} \neq \vec{\mathbf{m}}_{-1}$ with overwhelming probability since $\vec{\mathbf{m}}_{-1}$ is chosen randomly and information theoretically independent from the view of the adversary. The same is true for the case of $\ell - 1 = 0$ and $i^\star > 0$. In the case of $\ell - 1 \neq 0$ and $i^\star > 0$, since the messages are prefix-free, there exists $j^\star$ such that $\vec{\mathbf{m}}^{\dagger}_{\ell-1-j^\star} \neq \vec{\mathbf{m}}_{i^\star-j^\star}$ happens for the first time when $j^\star$ is increased from 0 to $\min(\ell - 1, i^\star) + 1$. In any of the cases ($j^\star$ is set to 1 for the case of $i^\star = 0$ or $\ell - 1 = 0$), signature $\sigma^{\dagger}_{\ell-1-j^\star}$ shares $\mathbf{s}$ and $\mathbf{v}$ with $\sigma_{i^\star-j^\star}$ as they are included in $\vec{\mathbf{m}}_{i^\star-j^\star+1} (= \vec{\mathbf{m}}^{\dagger}_{\ell-1-j^\star+1})$. This contradicts to the signature binding property of SIG as claimed in Lemma 5. □

5.3 Simulatable Signatures

5.3.1 Overview

A *simulatable signature scheme* is a signature scheme in the CRS model that makes it possible to create valid signatures without the signing-key but with a trapdoor associated to the common reference string. The notion is introduced in [4] but in an informal way dedicated for their purposes. We elaborate the notion and present a formal treatment with a reasonable construction in this section.

A simulatable signature is a useful tool in combination with a witness indistinguishable (WI) proof system. Unlike zero-knowledge (ZK) proofs, WI proof system

does not have a simulator. So when a signature is a part of the witness and the signer is corrupt and useless, simulatable signature can provide a correct witness to the entity having the trapdoor. This situation happens in reality, for instance, when we attempt to instantiate Fischlin's round-optimal blind signature scheme [55] (modified to use WI as suggested in [73, 4]).

It is known that a simulatable signature scheme can be unconditionally constructed from any regular signature scheme by modifying the verification predicate in such a way that a signature is accepted if it passes regular verification with respect to the signer's verification key *or* the verification key included in the CRS. This generic construction, however, inherently involves disjunction in the resulting verification predicate.

Our construction also uses the idea of two keys, but we use a trapdoor commitment scheme and a signature scheme combined. The commitment key is part of the CRS whereas the signing key is used for real signing. Also, a reference signature on a default message is required as part of the verification key. When a signature has to be simulated, the trapdoor for the commitment scheme is used to "equivocate" the reference signature to the required message. Since our main scheme in Section 5.1 already integrates a trapdoor commitment scheme in its construction, it would seem plausible to be able to move the commitment part of the verification key into the CRS. Ultimately, that is what we do. A formal proof, however, reveals that we need to have k flexible pairings to sign messages of size k, $k \geq 1$, without the trapdoor for the commitment part. This results in relying on k-SFP rather than SFP when dealing with messages of size $k \geq 2$.

5.3.2 Definitions

Definition 23 (Simulatable Signature Scheme)**.** *A simulatable signature scheme* SSIG *consists of algorithms* $\mathsf{SSIG}.\{\mathsf{Crs}, \mathsf{Key}, \mathsf{Chk}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Sim}\}$ *where* $\mathsf{SSIG}.\{\mathsf{Key}, \mathsf{Sign}, \mathsf{Vrf}\}$ *constitute a regular signature scheme (except that they take the CRS), and the extra algorithms works as follows.*

$\mathsf{SSIG.Crs}(1^\lambda)$: *A CRS generation algorithm that, on input security parameter* λ*, outputs a common reference string* $\mathtt{crs}$ *and a trapdoor* $\mathtt{tk}$*.*

$\mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk})$: *A verification key checking algorithm that, on input a verification key, returns* 1 *or* 0*.*

$\mathsf{SSIG.Sim}(\mathtt{crs}, \mathtt{vk}, m, \mathtt{tk})$: *A signature simulation algorithm that computes a signature* σ *for message* m *by using trapdoor* $\mathtt{tk}$*.*

By $\mathcal{M}_{\mathtt{vk}}$*, we denote the message space associated to* $\mathtt{vk}$*. By* $\mathcal{K}$*, we denote the set of* $(\mathtt{vk}, \mathtt{sk})$ *that can be generated by* $\mathsf{SSIG.Key}(\mathtt{crs})$*. Also by* $\mathcal{S}_{\mathtt{sk},m}$ *we denote the set of signatures that can be generated by* $\mathsf{SSIG.Sign}(\mathtt{crs}, \mathtt{sk}, m)$*.*

Completeness is defined in a standard way; with respect to correctly generated CRS, verification keys, and signatures, the verification function outputs 1 with probability 1.

Signature simulatability is defined in such a way that whenever adversary selects an appropriate message and verification key, then, by using the trapdoor of the CRS, it is possible to generate a signature that could have been generated by the proper signing operation. Formal definition follows.

Definition 24 (Signature-Simulatability)**.** *A signature scheme in the CRS model is simulatable if, for every CRS* $\mathtt{crs}$ *generated by* $(\mathtt{crs}, \mathtt{tk}) \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$*, for any*

$(m, \mathtt{vk})$, *if* $1 = \mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk}) \wedge m \in \mathcal{M}_{\mathtt{vk}}$, *then there exists* $\mathtt{sk}$ *such that* $(\mathtt{vk}, \mathtt{sk}) \in \mathcal{K}$, *and* $1 = \mathsf{SSIG.Vrf}(\mathtt{crs}, \mathtt{vk}, m, \sigma)$ *holds for any* $\sigma \leftarrow \mathsf{SSIG.Sim}(\mathtt{crs}, \mathtt{vk}, m, \mathtt{tk})$.

A relaxation would allow a negligible error in $\mathsf{SSIG.Vrf}$ for a message and a verification key chosen by an adversary. Note that the signature simulatability does not require simulated signatures be indistinguishable from the real ones. It is considered as a role of witness indistinguishable proof system coupled with the signature scheme.

Unforgeability is defined with respect to adaptive chosen message attacks. In the CRS model, however, a CRS is used for generating many keys and therefore, we must be careful that the keys should not be badly affected each other. By reflecting this concern, we allow an adversary to access an oracle that outputs correctly generated verification keys with respect to the same CRS. Furthermore, in our potential applications, the adversary is given a witness indistinguishable proof of holding a correct signature with respect to a given message and verification key. Let $\pi \leftarrow \Pi\ .\mathsf{Prove}((\mathtt{crs}, \mathtt{vk}_i, m), \sigma)$ denote the proof system for this purpose. Here $(\mathtt{crs}, \mathtt{vk}_i, m)$ is public, and σ is the witness, and π is the proof. We do not give much details to the proof system as the only property needed in this formulation is witness indistinguishability. The CRS for this proof system is implicitly given to the adversary. In summary the attack model includes the following three oracles.

- (Key Generation Oracle $\mathcal{O}_{\mathsf{vk}}(\cdot)$): On receiving i-th request, compute $(\mathtt{vk}_i, \mathtt{sk}_i) \leftarrow \mathsf{SSIG.Key}(\mathtt{crs})$, and return $\mathtt{vk}_i$. Record $\mathtt{vk}_i$ to Q_K.

- (Signing Oracle $\mathcal{O}_{\mathsf{sign}}(\cdot)$): On input $(\mathtt{vk}_i, m)$, return $\perp$ if $\mathtt{vk}_i$ is not recorded. Otherwise, compute $\sigma \leftarrow \mathsf{SSIG.Sign}(\mathtt{crs}, \mathtt{sk}_i, m)$ and return σ. Record m to $Q_m^{\mathtt{vk}_i}$.

- (Proof Oracle $\mathcal{O}_{\mathsf{wi}}(\cdot)$): On input $(\mathtt{vk}_i, m)$, return $\perp$ if $0 \leftarrow \mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk}_i)$

or $m \notin \mathcal{M}_{\mathtt{vk}_i}$. Otherwise, compute $\sigma \leftarrow \mathsf{SSIG.Sim}(\mathtt{crs}, \mathtt{vk}_i, m, \mathtt{tk})$, and $\pi \leftarrow \Pi.\mathsf{Prove}((\mathtt{crs}, \mathtt{vk}_i, m), \sigma)$. Then return π.

Definition 25 (Unforgeability with WI-Simulation)**.** *A signature scheme in the CRS model is unforgeable against adaptive chosen message and random verification key attacks with witness-indistinguishable simulation if, for any polynomial-time adversary $\mathcal{A}$, the following experiment returns* 1 *with negligible probability.*

$\mathrm{Exp}_{\textit{CRS EUF-CMA}}$:

$(\mathtt{crs}, \mathtt{tk}) \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$

$(m^\star, \sigma^\star, \mathtt{vk}^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}(\cdot), \mathcal{O}_{\mathsf{vk}}(\cdot), \mathcal{O}_{\mathsf{wi}}(\cdot)}(\mathtt{crs})$

Return 1 if $\mathtt{vk}^\star \in Q_K$ and $m \notin Q_m^{\mathtt{vk}^\star}$ and $1 \leftarrow \mathsf{SSIG.Vrf}(\mathtt{crs}, \mathtt{vk}^\star, m^\star, \sigma^\star)$.

Return 0, otherwise.

5.3.3 Construction

Let $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, \mathbf{g}, \tilde{\mathbf{g}}) \in \{\Lambda_{\mathsf{sym}}, \Lambda_{\mathsf{xdh}}, \Lambda_{\mathsf{sxdh}}\}$ be implicitly given to the algorithms below.

- $\mathsf{SSIG.Crs}(1^\lambda)$: Choose random generators $\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u$ from $\mathbb{G}_1^*$. For $i = 1, \ldots, k$, choose $\chi_i, \gamma_i, \delta_i$ from $\mathbb{Z}_p$, and compute $\mathbf{g}_i = \mathbf{g}_z^{\chi_i} \mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_z^{\chi_i} \mathbf{h}_u^{\delta_i}$. The common reference string is set to $\mathtt{crs} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k)$, and the trapdoor key is $tk = (\chi_1, \gamma_1, \delta_1, \ldots, \chi_k, \gamma_k, \delta_k)$.

- $\mathsf{SSIG.Key}(\mathtt{crs})$: Choose $\alpha, \beta \leftarrow \mathbb{Z}_p$ and compute $\{\mathbf{a}_i, \tilde{\mathbf{a}}_i\}_{i=0}^k \leftarrow \mathsf{RandExtend}(\mathbf{g}_r, \tilde{\mathbf{g}}^\alpha)$ and $\{\mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^k \leftarrow \mathsf{RandExtend}(\mathbf{h}_u, \tilde{\mathbf{g}}^\beta)$. Let $\mathtt{sk} = (\alpha, \beta)$. For some default message $\vec{\mathbf{m}}^* \in \mathbb{G}_2^k$, compute a reference signature $\sigma^* =$

$\mathsf{SSIG.Sign}(\mathtt{crs}, \mathtt{sk}, \vec{\mathbf{m}}^*)$ as shown below. Let $\mathtt{vk} = (\{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^k, \sigma^*)$. Output $(\mathtt{vk}, \mathtt{sk})$.

- $\mathsf{SSIG.Sign}(\mathtt{crs}, \mathtt{sk}, \vec{\mathbf{m}})$: For $i = 1$ to k and randomly chosen $\zeta_i, \rho_i, \varphi_i \leftarrow \mathbb{Z}_p$, set

$$(\text{If } \mathbf{m}_i \neq 1): \quad \mathbf{s}'_i = \mathbf{g}_z^{\zeta_i} \mathbf{g}_r^{\rho_i} \mathbf{g}_i^{-1}, \quad \mathbf{t}'_i = \mathbf{m}_i, \quad \mathbf{v}'_i = \mathbf{h}_z^{\zeta_i} \mathbf{h}_u^{\varphi_i} \mathbf{h}_i^{-1}, \quad \mathbf{w}'_i = \mathbf{m}_i,$$

$$(\text{If } \mathbf{m}_i = 1): \quad \mathbf{s}'_i = \mathbf{g}_z^{\zeta_i} \mathbf{g}_r^{\rho_i}, \qquad \mathbf{t}'_i = \tilde{\mathbf{g}}, \qquad \mathbf{v}'_i = \mathbf{h}_z^{\zeta_i} \mathbf{h}_u^{\varphi_i}, \qquad \mathbf{w}'_i = \tilde{\mathbf{g}}.$$

and

$$\mathbf{z} = \prod_{i=1}^{k} \mathbf{t}_i'^{-\zeta_i}, \qquad \mathbf{r} = \tilde{\mathbf{g}}^{\alpha} \prod_{i=1}^{k} \mathbf{t}_i'^{-\rho_i}, \qquad \mathbf{u} = \tilde{\mathbf{g}}^{\beta} \prod_{i=1}^{k} \mathbf{w}_i'^{-\varphi_i}.$$

Then, compute

$$\{\mathbf{s}_i, \mathbf{t}_i\}_{i=1}^k \leftarrow \mathsf{RandSeq}(\{\mathbf{s}'_i, \mathbf{t}'_i\}_{i=1}^k) \text{ and } \{\mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k \leftarrow \mathsf{RandSeq}(\{\mathbf{v}'_i, \mathbf{w}'_i\}_{i=1}^k)$$

. Output $\sigma = (\mathbf{z}, \mathbf{r}, \mathbf{u}, \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k)$ as a signature.

- $\mathsf{SSIG.Vrf}(\mathtt{crs}, \mathtt{vk}, \vec{\mathbf{m}}, \sigma)$: Parse σ as $(\mathbf{z}, \mathbf{r}, \mathbf{u}, \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k)$. Output 1 if

$$\mathbf{A} = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i)\ \hat{e}(\mathbf{s}_i, \mathbf{t}_i), \quad \text{and} \tag{5.6}$$

$$\mathbf{B} = \hat{e}(\mathbf{h}_z, \mathbf{z})\ \hat{e}(\mathbf{h}_u, \mathbf{u}) \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i)\ \hat{e}(\mathbf{v}_i, \mathbf{w}_i) \tag{5.7}$$

hold for $\mathbf{A} = \prod_{i=0}^{k} \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i)$ and $\mathbf{B} = \prod_{i=0}^{k} \hat{e}(\mathbf{b}_i, \tilde{\mathbf{b}}_i)$. Output 0, otherwise.

- $\mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk})$: Parse $\mathtt{vk}$ into $(\{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^k, \sigma^*)$ and return 0 if it fails. Check if every element of σ^* is in the appropriate group $\mathbb{G}_1$ or $\mathbb{G}_2$, and verify that $1 = \mathsf{SSIG.Vrf}(\mathtt{crs}, \mathtt{vk}, \vec{\mathbf{m}}^*, \sigma^*)$. If any of the checks fails, output 0. Otherwise, output 1.

- $\mathsf{SSIG.Sim}(\mathtt{crs}, \mathtt{vk}, \vec{\mathbf{m}}, tk)$: Take σ^* from $\mathtt{vk}$ and parse it into $(\mathbf{z}, \mathbf{r}, \mathbf{u}, \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k)$. By using $tk = (\chi_1, \gamma_1, \delta_1, \ldots, \chi_k, \gamma_k, \delta_k)$, com-

pute $(\mathbf{z}', \mathbf{r}', \mathbf{u}')$ as

$$\mathbf{z}' = \mathbf{z} \cdot \prod_{i=1}^{k} (\mathbf{m}_i / \mathbf{m}_i^*)^{-\chi_i}, \quad \mathbf{r}' = \mathbf{r} \cdot \prod_{i=1}^{k} (\mathbf{m}_i / \mathbf{m}_i^*)^{-\gamma_i}, \text{ and } \quad \mathbf{u}' = \mathbf{u} \cdot \prod_{i=1}^{k} (\mathbf{m}_i / \mathbf{m}_i^*)^{-\delta_i}.$$

Output $\sigma = (\mathbf{z}', \mathbf{r}', \mathbf{u}', \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k)$ as a signature for $\vec{\mathbf{m}}$.

5.3.4 Security

The security of SSIG relies on k-SFP, a generalization of SFP that has k flexible pairings in each relation as formally defined below. In the case of $k = 1$, k-SFP becomes SFP.

Assumption 9 (Simultaneous k-Flexible Pairing Assumption (k-SFP))**.** *Let Λ be a common parameter and let $\mathbf{g}_z$, $\mathbf{h}_z$, $\mathbf{g}_r$, and $\mathbf{h}_u$ be random generators of $\mathbb{G}_1$. Let $\{(\mathbf{a}_i, \tilde{\mathbf{a}}_i), (\mathbf{b}_i, \tilde{\mathbf{b}}_i)\}_{i=1}^k$ be random elements in $(\mathbb{G}_1 \times \mathbb{G}_2)^{2k}$. For $j = 1, \ldots, q$, let R_j be a tuple $(\mathbf{z}, \mathbf{r}, \mathbf{u}, \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^k) \in {\mathbb{G}_2}^3 \times (\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2)^k$ that satisfies*

$$\prod_{i=1}^{k} \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i) = \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{s}_i, \mathbf{t}_i), \quad \textit{and} \tag{5.8}$$

$$\prod_{i=1}^{k} \hat{e}(\mathbf{b}_i, \tilde{\mathbf{b}}_i) = \hat{e}(\mathbf{h}_z, \mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u}) \prod_{i=1}^{k} \hat{e}(\mathbf{v}_i, \mathbf{w}_i). \tag{5.9}$$

Given Λ, $\mathbf{g}_z$, $\mathbf{h}_z$, $\mathbf{g}_r$, $\mathbf{h}_u$, $\{(\mathbf{a}_i, \tilde{\mathbf{a}}_i), (\mathbf{b}_i, \tilde{\mathbf{b}}_i)\}_{i=1}^k$, and uniformly chosen $R_1, \ldots, R_q$, it is hard to find $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star, \{\mathbf{s}_i^\star, \mathbf{t}_i^\star, \mathbf{v}_i^\star, \mathbf{w}_i^\star\}_{i=1}^k)$, that fulfill relations (5.8) and (5.9). A restriction is that $\mathbf{z}^\star \neq 1$ and $\mathbf{z}^\star \neq \mathbf{z} \in R_j$ for every R_j.

Theorem 18. *[3] For any generic algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ breaks k-SFP with ℓ group operations and pairings is bound by $\mathcal{O}(k^2 \cdot q^2 + \ell^2)/p$.*

The proof of Theorem 18 that justifies the assumption in the generic bilinear group model. Like, k-SFP implies SDP for any $k \geq 1$. Somewhat contradictory to the fact that k-SFP is a generalization of SFP, there does not seem to be a useful reduction between them for $k \geq 2$.

Theorem 19. *Signature scheme* SSIG *is correct and signature-simulatable. It is EUF-CMA with WI-simulation in the multi-user setting if* k-SFP *holds for* Λ.

Proof. CORRECTNESS. Let I (and I^*) denote the set of indexes where $\mathbf{m}_i \neq 1$ (and $\mathbf{m}_i = 1$, respectively) in SIG.Sign. Regarding the first relation in the verification predicates, we have:

$$\begin{aligned}
&\hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i)\ \hat{e}(\mathbf{s}_i, \mathbf{t}_i)\ = \\
&= \hat{e}\left(\mathbf{g}_z, \prod_{i=1}^{k} \mathbf{t}_i'^{-\zeta_i}\right)\ \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha} \prod_{i=1}^{k} \mathbf{t}_i'^{-\rho_i}\right)\ \cdot \\
&\quad \cdot \prod_{i \in I} \hat{e}\left(\mathbf{g}_i, \mathbf{m}_i\right)\ \hat{e}\left(\mathbf{g}_z^{\zeta_i} \mathbf{g}_r^{\rho_i} \mathbf{g}_i^{-1}, \mathbf{t}_i'\right)\ \prod_{i \in I^*} \hat{e}\left(\mathbf{g}_z^{\zeta_i} \mathbf{g}_r^{\rho_i}, \mathbf{t}_i'\right) \\
&= \hat{e}\left(\mathbf{g}_z, \prod_{i=1}^{k} \mathbf{t}_i'^{-\zeta_i}\right)\ \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha} \prod_{i=1}^{k} \mathbf{t}_i'^{-\rho_i}\right)\ \prod_{i=1}^{k} \hat{e}\left(\mathbf{g}_z^{\zeta_i} \mathbf{g}_r^{\rho_i}, \mathbf{t}_i'\right) \\
&= \hat{e}(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha}) = \mathbf{A}
\end{aligned}$$

The other relation can be verified in the same manner as $\mathbf{z} = \prod_{i=1}^{k} \mathbf{t}_i'^{-\zeta_i} = \prod_{i=1}^{k} \mathbf{w}_i'^{-\zeta_i}$. SIGNATURE-SIMULATABILITY. For every $\mathtt{vk} = (\{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^{k}, \sigma^*)$ such that $1 = \mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk})$, every elements in $\{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^{k}$ is in the correct group $\mathbb{G}_1$ and $\mathbb{G}_2$. Clearly there are (α, β) so that $\prod_{i=0}^{k} \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i) = \hat{e}(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha})$ and $\prod_{i=0}^{k} \hat{e}(\mathbf{b}_i, \tilde{\mathbf{b}}_i) = \hat{e}(\mathbf{h}_u, \tilde{\mathbf{g}}^{\beta})$ hold. Therefore such $\{(\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i)\}_{i=0}^{k}$ and (α, β) are a correct key pair. The rest is to show that SSIG.Sim correctly works to turn valid signature $\sigma^* = (\mathbf{z}, \mathbf{r}, \mathbf{u}, \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^{k})$ for $\vec{\mathbf{m}}^*$ into a signature $\sigma = (\mathbf{z}', \mathbf{r}', \mathbf{u}', \{\mathbf{s}_i, \mathbf{t}_i, \mathbf{v}_i, \mathbf{w}_i\}_{i=1}^{k})$

for message $\vec{\mathbf{m}}$. It holds that

$$\begin{aligned}
&\hat{e}(\mathbf{g}_z, \mathbf{z}')\, \hat{e}(\mathbf{g}_r, \mathbf{r}') \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i)\, \hat{e}(\mathbf{s}_i, \mathbf{t}_i) \\
&= \hat{e}(\mathbf{g}_z, \mathbf{z} \cdot \prod_{i=1}^{k} (\mathbf{m}_i/\mathbf{m}_i^*)^{-\chi_i})\, \hat{e}(\mathbf{g}_r, \mathbf{r} \cdot \prod_{i=1}^{k} (\mathbf{m}_i/\mathbf{m}_i^*)^{-\gamma_i}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_z^{\chi_i} \mathbf{g}_r^{\gamma_i}, \mathbf{m}_i)\, \hat{e}(\mathbf{s}_i, \mathbf{t}_i) \\
&= \hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}) \prod_{i=1}^{k} \hat{e}(\mathbf{g}_i, \mathbf{m}_i^*)\, \hat{e}(\mathbf{s}_i, \mathbf{t}_i) = \mathbf{A}.
\end{aligned}$$

The other relation $\hat{e}(\mathbf{h}_z, \mathbf{z}')\, \hat{e}(\mathbf{h}_u, \mathbf{u}') \prod_{i=1}^{k} \hat{e}(\mathbf{h}_i, \mathbf{m}_i)\, \hat{e}(\mathbf{v}_i, \mathbf{w}_i) = \mathbf{B}$ can be verified in the same way. Thus the output from $\mathsf{SSIG.Sim}$ is a valid signature for $\vec{\mathbf{m}}$.

EUF-CMA WITH WI-SIMULATION. Given an instance of k-SFP, we simulate the view of $\mathcal{A}$ in the attack environment as follows.

- (CRS generation) : Do the same as original $\mathsf{SSIG.Crs}$ by using given generators $(\mathbf{g}_r, \mathbf{h}_u, \mathbf{g}_z, \mathbf{h}_z)$ in the input instance. The commitment-key is assigned as a CRS, $\mathtt{crs} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^k)$, and the trapdoor is $tk = (\chi_1, \gamma_1, \delta_1, \ldots, \chi_k, \gamma_k, \delta_k)$.

- (Key Generation Oracle $\mathcal{O}_{\mathsf{vk}}(\cdot)$) : Take $\{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=1}^k$ from the input instance. Choose $\zeta, \rho, \varphi \leftarrow \mathbb{Z}_p$ and $\tilde{\mathbf{g}} \leftarrow \mathbb{G}_2^*$. Then compute $\{\mathbf{a}'_i, \tilde{\mathbf{a}}'_i\}_{i=0}^k$ as the output of $\mathsf{RandSeq}\left((\mathbf{g}_z^{\zeta}\mathbf{g}_r^{\rho}, \tilde{\mathbf{g}}), (\mathbf{a}_1, \tilde{\mathbf{a}}_1), \ldots, (\mathbf{a}_k, \tilde{\mathbf{a}}_k)\right)$ and $\{\mathbf{b}'_i, \tilde{\mathbf{b}}'_i\}_{i=0}^k$ as the sequential randomization $\mathsf{RandSeq}\left((\mathbf{h}_z^{\zeta}\mathbf{h}_u^{\varphi}, \tilde{\mathbf{g}}), (\mathbf{b}_1, \tilde{\mathbf{b}}_1), \ldots, (\mathbf{b}_k, \tilde{\mathbf{b}}_k)\right)$. Then simulate a reference signature σ_0 as described below. The verification key is $\mathtt{vk} = (\{\mathbf{a}'_i, \tilde{\mathbf{a}}'_i, \mathbf{b}'_i, \tilde{\mathbf{b}}'_i\}_{i=0}^k, \sigma_0)$. Record $\mathtt{vk}$ to Q_K.

- (Signing Oracle $\mathcal{O}_{\mathsf{sign}}(\cdot)$) : Given message $\vec{\mathbf{m}}$ and $\mathtt{vk}$, return $\perp$ if $\mathtt{vk}$ is not in Q_K. Take a new tuple $R_j = (\mathbf{z}_j, \mathbf{r}_j, \mathbf{u}_j, \{\mathbf{s}_{ij}, \mathbf{t}_{ij}, \mathbf{v}_{ij}, \mathbf{w}_{ij}\}_{i=1}^k)$ from the given instance. Then compute

$$\mathbf{z}'_j = \mathbf{z}_j\, \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k} \mathbf{m}_i^{-\chi_i}, \quad \mathbf{r}'_j = \mathbf{r}_j\, \tilde{\mathbf{g}}^{\rho} \prod_{i=1}^{k} \mathbf{m}_i^{-\gamma_i}, \quad \mathbf{u}'_j = \mathbf{u}_j\, \tilde{\mathbf{g}}^{\varphi} \prod_{i=1}^{k} \mathbf{m}_i^{-\delta_i}. \tag{5.10}$$

by using (ζ, ρ, φ) used for generating $\mathtt{vk}$ in $\mathcal{O}_{\mathsf{vk}}(\cdot)$. The output signature is $\sigma_j = (\mathbf{z}'_j, \mathbf{r}'_j, \mathbf{u}'_j, \{\mathbf{s}_{ij}, \mathbf{t}_{ij}, \mathbf{v}_{ij}, \mathbf{w}_{ij}\}_{i=1}^k)$.

- (Simulation Oracle $\mathcal{O}_{\mathsf{wi}}(\cdot)$) : Given $\vec{\mathbf{m}}$ and $\mathtt{vk}$, return $\perp$ if $0 \leftarrow \mathsf{SSIG.Chk}(\mathtt{crs}, \mathtt{vk}_i)$ or $m \notin \mathcal{M}_{\mathtt{vk}_i}$. If $\mathtt{vk}$ is in Q_K, compute $\sigma \leftarrow \mathcal{O}_{\mathsf{sign}}(\cdot)(\vec{\mathbf{m}}, \mathtt{vk})$. Otherwise, compute $\sigma \leftarrow \mathsf{SSIG.Sim}(\mathtt{crs}, \mathtt{vk}, \vec{\mathbf{m}}, tk)$. Then compute $\pi \leftarrow \Pi\ .\mathsf{Prove}((\mathtt{crs}, \mathtt{vk}, m), \sigma)$ and return π.

When $\mathcal{A}$ outputs $(\vec{\mathbf{m}}^\dagger, \mathbf{z}^\dagger, \mathbf{r}^\dagger, \mathbf{u}^\dagger, \{\mathbf{s}_i^\dagger, \mathbf{v}_i^\dagger, \mathbf{t}_i^\dagger, \mathbf{w}_i^\dagger\}_{i=1}^k)$, compute

$$\mathbf{z}^\star = \left(\mathbf{z}^\dagger\right) \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^k \left(\mathbf{m}_i^\dagger\right)^{\chi_i}, \quad \mathbf{r}^\star = \left(\mathbf{r}^\dagger\right) \tilde{\mathbf{g}}^{-\rho} \prod_{i=1}^k \left(\mathbf{m}_i^\dagger\right)^{\gamma_i},$$
$$\text{and} \quad \mathbf{u}^\star = \left(\mathbf{u}^\dagger\right) \tilde{\mathbf{g}}^{-\varphi} \prod_{i=1}^k \left(\mathbf{m}_i^\dagger\right)^{\delta_i},$$

and set $\mathbf{s}_i^\star = \mathbf{s}_i^\dagger$, $\mathbf{t}_i^\star = \mathbf{t}_i^\dagger$, $\mathbf{v}_i^\star = \mathbf{v}_i^\dagger$, and $\mathbf{w}_i^\star = \mathbf{w}_i^\dagger$ for $i = 1, \ldots, k$. The reduction algorithm outputs a tuple $(\mathbf{z}^\star, \mathbf{r}^\star, \mathbf{u}^\star, \{\mathbf{s}_i^\star, \mathbf{t}_i^\star, \mathbf{v}_i^\star, \mathbf{w}_i^\star\}_{i=1}^k)$ and terminates.

It can be verified by inspection that the CRS, the verification-key and the signatures perfectly follow the legitimate distribution. When $\mathcal{A}$ is successful, for the outputs of the reduction algorithm, it holds that

$$\begin{aligned}
&\hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\ \hat{e}(\mathbf{g}_r, \mathbf{r}^\star)\ \prod_{i=1}^k \hat{e}(\mathbf{s}_i^\star, \mathbf{t}_i^\star) \\
&= \hat{e}\left(\mathbf{g}_z, \left(\mathbf{z}^\dagger\right) \tilde{\mathbf{g}}^{-\zeta} \prod_{i=1}^k \left(\mathbf{m}_i^\dagger\right)^{\chi_i}\right)\ \hat{e}\left(\mathbf{g}_r, \left(\mathbf{r}^\dagger\right) \tilde{\mathbf{g}}^{-\rho} \prod_{i=1}^k \left(\mathbf{m}_i^\dagger\right)^{\gamma_i}\right)\ \prod_{i=1}^k \hat{e}\left(\mathbf{s}_i^\dagger, \mathbf{t}_i^\dagger\right) \\
&= \hat{e}\left(\mathbf{g}_z^{-\zeta} \mathbf{g}_r^{-\rho}, \tilde{\mathbf{g}}\right)\ \hat{e}\left(\mathbf{g}_z, \mathbf{z}^\dagger\right)\ \hat{e}\left(\mathbf{g}_r, \mathbf{r}^\dagger\right)\ \prod_{i=1}^k \hat{e}\left(\mathbf{g}_i, \mathbf{m}_i^\dagger\right)\ \hat{e}\left(\mathbf{s}_i^\dagger, \mathbf{t}_i^\dagger\right) \\
&= \hat{e}(\mathbf{a}_0, \tilde{\mathbf{a}}_0)^{-1}\ \prod_{i=0}^k \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i) = \prod_{i=1}^k \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i).
\end{aligned}$$

One can also verify that $\hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\ \hat{e}(\mathbf{h}_u, \mathbf{u}^\star)\ \prod_{i=1}^k \hat{e}(\mathbf{v}_i^\star, \mathbf{w}_i^\star) = \prod_{i=1}^k \hat{e}(\mathbf{b}_i, \tilde{\mathbf{b}}_i)$ holds in the same way.

What remains is to show that $\mathbf{z}^\star$ is not in $\{1, \mathbf{z}_1, \ldots, \mathbf{z}_q\}$. Basically the argument is the same as the one in the proof of Theorem 16 in Section 5.1.2 by using the fact that the parameters $\zeta, \chi_1, \ldots, \chi_k$ are independent from $\mathcal{A}$'s view. For the same argument to hold, we have to show that when those values are also used for simulating $\mathcal{O}_{\mathsf{wi}}(\cdot)$, they remain information theoretically hidden even after π is seen by the adversary. For $\mathtt{vk}$ generated by $\mathcal{O}_{\mathsf{vk}}(\cdot)$, simulation is done just by calling the signing oracle, and, as we know, the signature does not reveal any information about the parameters. On the other hand, for $\mathtt{vk}$ that is not generated by $\mathcal{O}_{\mathsf{vk}}(\cdot)$, $\mathsf{SSIG.Chk}$ guarantees that there exists a corresponding signing key $\mathtt{sk}$. Since Π .Prove is witness indistinguishable and there exists randomness that is consistent to a valid signature that could have been generated by the signing algorithm using $\mathtt{sk}$, like in the previous case the parameters remain hidden. Thus the claim holds. □

5.4 Signing Mixed-Group Messages in the Asymmetric Setting

5.4.1 Overview

The signature schemes presented so far are capable of signing messages composed of group elements from one of the base groups. In the symmetric setting Λ_{sym}, when $\mathbb{G}_1 = \mathbb{G}_2$, it does not matter which base group it is. However, for the case when there is no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ (or $\mathbb{G}_2$ to $\mathbb{G}_1$), it is not immediately clear how to sign messages composed of elements from both $\mathbb{G}_1$ and $\mathbb{G}_2$. Such case can easily arise when using a structure-preserving signature scheme with Groth-Sahai proofs in the asymmetric setting. This setting is often of interest as it provides more efficient bit-size representation of the group elements and usually

requires simpler assumptions [62].

In this section, we construct a signature scheme with message space $\mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$. It is worth pointing out that splitting the message into two parts, one in $\mathbb{G}_1^{k_1}$ and another in $\mathbb{G}_2^{k_2}$, and signing each of them independently results in a trivial forgery after two signing queries.

5.4.2 Construction

Let $\mathsf{SIG2}$ be the constant-size signature scheme from Section 5.1, whose message space is $\mathbb{G}_2^{k_2}$. Let $\mathsf{SIG1}$ be a 'dual' scheme obtained by exchanging $\mathbb{G}_1$ and $\mathbb{G}_2$ in the same scheme. Let the message space of $\mathsf{SIG1}$ is $\mathbb{G}_1^{k_1+1}$. (Note that we use the same letters for variables in a signature. Accordingly, $\mathbf{z}, \mathbf{r}, \mathbf{u}, \mathbf{t}$, and $\mathbf{w}$ are in the same group as the input message while $\mathbf{s}$ and $\mathbf{v}$ are in the other group.) By using these signature scheme, we construct signature scheme XSIG whose message space be $\mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$ as follows. Let $(\vec{\mathbf{m}}, \vec{\tilde{\mathbf{m}}})$ be a message in $\mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$. For vector $\vec{\mathbf{m}} \in \mathbb{G}_1^{k_1}$ and single element $\mathbf{s} \in \mathbb{G}_1$, let $\vec{\mathbf{m}}||\mathbf{s}$ denote a vector in $\mathbb{G}_1^{k_1+1}$ obtained by appending $\mathbf{s}$ to the end of $\vec{\mathbf{m}}$. Let $\Lambda = \Lambda_{\mathsf{sxdh}}$ be given to the functions described below.

- $\mathsf{XSIG.Key}(1^\lambda)$: Run $(\mathtt{vk}_1, \mathtt{sk}_1) \leftarrow \mathsf{SIG1.Key}(1^\lambda)$ and $(\mathtt{vk}_2, \mathtt{sk}_2) \leftarrow \mathsf{SIG2.Key}(1^\lambda)$. Output $(\mathtt{vk}, \mathtt{sk}) = ((\mathtt{vk}_1, \mathtt{vk}_2), (\mathtt{sk}_1, \mathtt{sk}_2))$.

- $\mathsf{XSIG.Sign}(\mathtt{sk}, (\vec{\mathbf{m}}, \vec{\tilde{\mathbf{m}}}))$: Run $\sigma_2 = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w}) \leftarrow \mathsf{SIG2.Sign}(\mathtt{sk}_2, \vec{\tilde{\mathbf{m}}})$ and $\sigma_1 = (\mathbf{z}', \mathbf{r}', \mathbf{s}', \mathbf{t}', \mathbf{u}', \mathbf{v}', \mathbf{w}') \leftarrow \mathsf{SIG1.Sign}(sk_1, \vec{\mathbf{m}}||\mathbf{s})$. Output $\sigma = (\sigma_1, \sigma_2)$.

- $\mathsf{XSIG.Vrf}(\mathtt{vk}, (\vec{\mathbf{m}}, \vec{\tilde{\mathbf{m}}}), (\sigma_1, \sigma_2))$: Take $\mathbf{s} \in \mathbb{G}_1$ from σ_2. Run $b_2 = \mathsf{SIG2.Vrf}(\mathtt{vk}_2, \vec{\tilde{\mathbf{m}}}, \sigma_2)$ and $b_1 = \mathsf{SIG1.Vrf}(\mathtt{vk}_1, \vec{\mathbf{m}}||\mathbf{s}, \sigma_1)$. Output 1 if $b_2 = b_1 = 1$. Output 0, otherwise.

5.4.3 Security

Theorem 20. *If* SIG1 *and* SIG2 *are EUF-CMA, so is* XSIG.

Proof. Suppose that there is a successful adversary that launches chosen message attacks and outputs a valid forgery, $((\vec{\mathbf{m}}^\dagger, \vec{\tilde{\mathbf{m}}}^\dagger), (\sigma_1^\dagger, \sigma_2^\dagger))$. Consider $\mathbf{s}^\dagger$ included in $\sigma_2^\dagger$. Observe that $\sigma_1^\dagger$ is a signature for $\mathbf{m}^\dagger||\mathbf{s}^\dagger$. We then have 3 cases.

Type-I $\vec{\mathbf{m}}^\dagger||\mathbf{s}^\dagger$ has never been signed by the signing oracle. This case contradicts to the unforgeability of SIG1.

Type-II $\vec{\tilde{\mathbf{m}}}^\dagger$ has never been signed by the signing oracle. This case contradicts to the unforgeability of SIG2.

Type-III Both $\vec{\mathbf{m}}^\dagger||\mathbf{s}^\dagger$ and $\vec{\tilde{\mathbf{m}}}^\dagger$ have been signed by the signing oracle in separate queries. This case contradicts the DBP assumption, as we will show.

Since the first two forgery cases are trivial, we focus on Type-III. We construct a reduction algorithm that simulates the environment for adversary $\mathcal{A}$ launching an adaptive chosen message attack on XSIG. The simulator only simulates SIG2 and honestly acts with respect to SIG1. We thus describe the simulation only with respect to SIG2. Given an instance of the DBP assumption, $(\Lambda, \mathbf{g}_z, \mathbf{g}_r)$, the simulator works as follows:

- (Key Generation): Choose random $\mathbf{h}_z$ and $\mathbf{h}_u$ from $\mathbb{G}_1^*$. Then, for $i = 1, \ldots, k_2$, set $\mathbf{g}_i = \mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}$ and $\mathbf{h}_i = \mathbf{h}_z^{\chi_i}\mathbf{h}_u^{\delta_i}$ for random χ_i, γ_i, and δ_i in $\mathbb{Z}_p$, and choose α, $\beta \leftarrow \mathbb{Z}_p$. Then compute $\{\mathbf{a}_i, \tilde{\mathbf{a}}_i\}_{i=0}^1 \leftarrow \mathsf{RandExtend}(\mathbf{g}_r, \tilde{\mathbf{g}}^\alpha)$ and $\{\mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^1 \leftarrow \mathsf{RandExtend}(\mathbf{h}_u, \tilde{\mathbf{g}}^\beta)$. Output $\mathtt{vk}_2 = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \{\mathbf{g}_i, \mathbf{h}_i\}_{i=1}^{k_2}, \{\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{b}_i, \tilde{\mathbf{b}}_i\}_{i=0}^1)$.

- (Signature Issuing): Given message $\vec{\tilde{\mathbf{m}}} \in \mathbb{G}_2^{k_2}$, choose ζ, ρ, τ, φ, $\omega \leftarrow \mathbb{Z}_p$ and set

$$\mathbf{z} = \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k_2} \tilde{\mathbf{m}}_i^{-\chi_i}, \quad \mathbf{r} = \tilde{\mathbf{g}}^{\zeta\rho/\tau+\alpha} \prod_{i=1}^{k_2} \tilde{\mathbf{m}}_i^{-\gamma_i}, \quad \mathbf{s} = \mathbf{g}_z^{\tau}\mathbf{g}_r^{\rho}, \quad \mathbf{t} = \tilde{\mathbf{g}}^{-\zeta/\tau}$$
$$\mathbf{u} = \tilde{\mathbf{g}}^{\zeta\varphi/\omega+\beta} \prod_{i=1}^{k_2} \tilde{\mathbf{m}}_i^{-\delta_i}, \quad \mathbf{v} = \mathbf{h}_z^{\omega}\mathbf{h}_u^{\varphi}, \quad \mathbf{w} = \tilde{\mathbf{g}}^{-\zeta/\omega}.$$

  Output $\sigma_2 = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$.

To see the correctness of the simulated signatures, observe that

$$\begin{aligned}
&\hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}, \mathbf{t}) \prod_{i=1}^{k_2} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{m}}_i) \\
&= \hat{e}\left(\mathbf{g}_z, \tilde{\mathbf{g}}^{\zeta} \prod_{i=1}^{k_2} \tilde{\mathbf{m}}_i^{-\chi_i}\right)\ \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^{\zeta\rho/\tau+\alpha} \prod_{i=1}^{k_2} \tilde{\mathbf{m}}_i^{-\gamma_i}\right)\ \hat{e}\left(\mathbf{s}, \mathbf{t}\right)\ \prod_{i=1}^{k_2} \hat{e}\left(\mathbf{g}_z^{\chi_i}\mathbf{g}_r^{\gamma_i}, \tilde{\mathbf{m}}_i\right) \\
&= \hat{e}\left(\mathbf{g}_z, \tilde{\mathbf{g}}^{\zeta}\right)\ \hat{e}\left(\mathbf{g}_r, \tilde{\mathbf{g}}^{\zeta\rho/\tau+\alpha}\right)\ \hat{e}\left(\mathbf{g}_z^{\tau}\mathbf{g}_r^{\rho}, \tilde{\mathbf{g}}^{-\zeta/\tau}\right) \\
&= \hat{e}(\mathbf{g}_r, \tilde{\mathbf{g}}^{\alpha}) = \prod_{i=0}^{1} \hat{e}(\mathbf{a}_i, \tilde{\mathbf{a}}_i)
\end{aligned}$$

holds. The other verification predicate holds in the same way. It is also not hard to inspect that the signatures follow a proper distribution due to the random coins in the simulation.

Let $\sigma^{\dagger} = (\sigma_1^{\dagger}, \sigma_2^{\dagger})$, where $\sigma_2^{\dagger} = (\mathbf{z}^{\dagger}, \mathbf{r}^{\dagger}, \mathbf{s}^{\dagger}, \mathbf{t}^{\dagger}, \mathbf{u}^{\dagger}, \mathbf{v}^{\dagger}, \mathbf{w}^{\dagger})$, be the forged signature for a message $(\vec{\mathbf{m}}^{\dagger}, \vec{\tilde{\mathbf{m}}}^{\dagger})$. By the forgery type constraints, there exists a signing query with message $(\vec{\mathbf{m}}^{\dagger}, \vec{\tilde{\mathbf{m}}})$ such that $\vec{\tilde{\mathbf{m}}} \neq \vec{\tilde{\mathbf{m}}}^{\dagger}$ and its signature $\sigma_2 = (\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ satisfies $\mathbf{s} = \mathbf{s}^{\dagger}$. Accordingly, we have

$$\hat{e}(\mathbf{g}_z, \mathbf{z}^{\dagger})\ \hat{e}(\mathbf{g}_r, \mathbf{r}^{\dagger})\ \hat{e}(\mathbf{s}^{\dagger}, \mathbf{t}^{\dagger})\ \prod_{i=1}^{k_2} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{m}}_i^{\dagger}) = \hat{e}(\mathbf{g}_z, \mathbf{z})\ \hat{e}(\mathbf{g}_r, \mathbf{r})\ \hat{e}(\mathbf{s}^{\dagger}, \mathbf{t})\ \prod_{i=1}^{k_2} \hat{e}(\mathbf{g}_i, \tilde{\mathbf{m}}_i). \tag{5.11}$$

Recall that $\mathbf{s}^{\dagger} = \mathbf{s} = \mathbf{g}_z^{\tau}\mathbf{g}_r^{\rho}$. By dividing the left-hand of the above equation by its

right-hand, we have

$$\begin{aligned}
1 &= \hat{e}\left(\mathbf{g}_z, \frac{\mathbf{z}^\dagger}{\mathbf{z}}\right)\ \hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}^\dagger}{\mathbf{r}}\right)\ \hat{e}\left(\mathbf{s}^\dagger, \frac{\mathbf{t}^\dagger}{\mathbf{t}}\right)\ \prod_{i=1}^{k_2} \hat{e}\left(\mathbf{g}_i, \frac{\tilde{\mathbf{m}}_i^\dagger}{\tilde{\mathbf{m}}_i}\right) \\
&= \hat{e}\left(\mathbf{g}_z, \frac{\mathbf{z}^\dagger}{\mathbf{z}} \prod_{i=1}^{k_2} \left(\frac{\tilde{\mathbf{m}}_i^\dagger}{\tilde{\mathbf{m}}_i}\right)^{\chi_i}\right)\ \hat{e}\left(\mathbf{g}_r, \frac{\mathbf{r}^\dagger}{\mathbf{r}} \prod_{i=1}^{k_2} \left(\frac{\tilde{\mathbf{m}}_i^\dagger}{\tilde{\mathbf{m}}_i}\right)^{\gamma_i}\right)\ \hat{e}\left(\mathbf{g}_z^\tau \mathbf{g}_r^\rho, \frac{\mathbf{t}^\dagger}{\mathbf{t}}\right) \\
&= \hat{e}(\mathbf{g}_z, \mathbf{z}^\star)\ \hat{e}(\mathbf{g}_r, \mathbf{r}^\star),
\end{aligned}$$

where $\mathbf{z}^\star = \frac{\mathbf{z}^\dagger}{\mathbf{z}}(\frac{\mathbf{t}^\dagger}{\mathbf{t}})^\tau \prod_i (\frac{\tilde{\mathbf{m}}_i^\dagger}{\tilde{\mathbf{m}}_i})^{\chi_i}$ and $\mathbf{r}^\star = \frac{\mathbf{r}^\dagger}{\mathbf{r}}(\frac{\mathbf{t}^\dagger}{\mathbf{t}})^\rho \prod_i (\frac{\tilde{\mathbf{m}}_i^\dagger}{\tilde{\mathbf{m}}_i})^{\gamma_i}$.

Since $\vec{\tilde{\mathbf{m}}}^\dagger \neq \vec{\tilde{\mathbf{m}}}$, there exists $i^\star$ such that $\tilde{\mathbf{m}}_{i^\star}^\dagger / \tilde{\mathbf{m}}_{i^\star} \neq 1$. Observe that $\chi_{i^\star}$ is independent of the view of the adversary. Hence the probability that $\mathbf{z}^\star = 1$ is negligible. The reduction algorithm outputs $(\mathbf{z}^\star, \mathbf{r}^\star)$ as a valid answer to the given instance of DBP. □

5.5 Strongly Unforgeable Signatures

The following generic construction of sEUF-CMA signature scheme is in [18]. Let SIG be a signature scheme and OTS be a one-time signature scheme. The construction requires that the message space of SIG covers the public-key space of OTS.

- FSIG1.Key(1^λ): Run $(\mathtt{vk}, \mathtt{sk}) \leftarrow$ SIG.Key(1^λ). Output $(\mathtt{vk}, \mathtt{sk})$.
- FSIG1.Sign($\mathtt{sk}, \vec{\mathbf{m}}$): $(\mathtt{vk}_o, \mathtt{sk}_o) \leftarrow$ OTS.Key(1^λ), $\sigma_1 \leftarrow$ SIG.Sign($\mathtt{sk}, \mathtt{vk}_o || \vec{\mathbf{m}}$), $\sigma_2 \leftarrow$ OTS.Sign($\mathtt{sk}_o, \sigma_1$). Output $\sigma = (\mathtt{vk}_o, \sigma_1, \sigma_2)$.
- FSIG1.Vrf($\mathtt{vk}, \vec{\mathbf{m}}, \sigma$): Parse σ into $(\mathtt{vk}_o, \sigma_1, \sigma_2)$. Compute $b_1 \leftarrow$ SIG.Vrf($\mathtt{vk}, \mathtt{vk}_o || \vec{\mathbf{m}}, \sigma_1$) and $b_2 \leftarrow$ OTS.Vrf($\mathtt{vk}_o, \sigma_1, \sigma_2$). Output 1 if $b_2 = b_1 = 1$. Output 0, otherwise.

As shown in [18], signature scheme FSIG1 is strongly EUF-CMA if SIG is EUF-CMA and OTS is sEUF-CMA against one-time chosen message attacks. In the one-

time chosen message attacks, the adversary is allowed to make at most one signing query. We refer to [18] for a proof.

By instantiating SIG and OTS by the ones in Section 5.1 and Section 4.2.1 with setting $\Lambda = \Lambda_{\mathsf{sym}}$, the resulting $\mathsf{FSIG1}$ outputs a signature of 32 group elements ($|\mathtt{vk}_o| = 22$, $|\sigma_1| = 7$, $|\sigma_2| = 3$) which is a constant in the size of $\vec{\mathbf{m}}$.

We can gain efficiency by using the same bases in SIG and OTS in the above instantiation. Let $\mathsf{FSIG2}$ denote this variant. Concretely, in $\mathsf{FSIG2}$, one-time signature OTS takes bases $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \mathbf{g}_1, \mathbf{h}_1, \ldots, \mathbf{g}_7, \mathbf{h}_7)$ from the verification key of SIG. Then $\mathtt{vk}_o$ only includes $\mathbf{a}$ and $\mathbf{b}$. As a result, a signature of $\mathsf{FSIG2}$ consists of 12 group elements.

The generic security argument for $\mathsf{FSIG1}$ no longer holds for $\mathsf{FSIG2}$ since SIG and OTS are not independent. We are still able to show that $\mathsf{FSIG2}$ is sEUF-CMA as follows.

Theorem 21. *Signature scheme $\mathsf{FSIG2}$ is sEUF-CMA if* SFP *holds for $\Lambda = \Lambda_{\mathsf{sym}}$.*

Proof. First observe that we cannot show a black-box reduction to the security of SIG and OTS by using their signing oracles since they share the bases. We instead construct reduction to their underlying assumptions. This is possible because, in both security proofs for SIG and OTS, bases $(\mathbf{g}_1, \mathbf{h}_1, \ldots, \mathbf{g}_7, \mathbf{h}_7)$ in the verificaion keys are set in the same manner with respect to $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u)$. Thus, while we simulate the signing oracle for SIG, we can also simulate signatures of OTS.

Let $\mathcal{O}_{\mathsf{sign}}(\cdot)$ be the signing oracle of $\mathsf{FSIG2}$. Suppose that an adversary outputs a valid forgery $(\mathtt{vk}_o^\dagger, \sigma_1^\dagger, \sigma_2^\dagger, \vec{\mathbf{m}}^\dagger)$. Let $Q_i = (\mathtt{vk}_o, \sigma_1, \sigma_2, \vec{\mathbf{m}})$ for $i = 1, \ldots, q$ be the record of interaction between the adversary and $\mathcal{O}_{\mathsf{sign}}(\cdot)$.

For a type of adversary that causes $(\mathtt{vk}_o^\dagger, \vec{\mathbf{m}}^\dagger) \neq (\mathtt{vk}_o, \vec{\mathbf{m}})$ for any Q_i, we construct a reduction to SFP by simulating SIG as shown in the proof of Theorem 16. We also

simulate OTS as shown in the proof of Theorem 12. Note that the simulation of OTS is possible since the way bases $\mathbf{g}_i$ and $\mathbf{h}_i$ are set in simulating SIG is exactly the same as that in simulating OTS. Thus we can successfully simulate FSIG2 by using these simulated SIG and OTS. It is important to see that exponents hidden in $\mathbf{g}_i$ and $\mathbf{h}_i$ remain independent of the view of the adversary even with the simulation of OTS. Thus a successful forgery results in a contradiction to SFP as shown in the proof of Theorem 16.

For the other type of adversary that causes $(\mathtt{vk}_o^\dagger, \vec{\mathbf{m}}^\dagger) = (\mathtt{vk}_o, \vec{\mathbf{m}})$ and $(\sigma_1^\dagger, \sigma_2^\dagger) \neq (\sigma_1, \sigma_2)$ for some $Q_{i^\star}$, we show a reduction to SDP by simulating OTS as shown in the proof of Theorem 12. Since simulation of SIG needs an instance of SFP, we generate a random instance of SFP from that of SDP as follows. Given an SDP instance $(\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u)$, set $\mathbf{a} = \mathbf{g}_r^\alpha$, $\mathbf{b} = \mathbf{h}_u^\beta$, and $\tilde{\mathbf{a}} = \tilde{\mathbf{b}} = \tilde{\mathbf{g}}$. Then for $j = 1, \ldots, q$, compute reference $R_j = (\mathbf{z}, \mathbf{r}, \mathbf{u}, \mathbf{s}, \mathbf{t}, \mathbf{v}, \mathbf{w})$ by choosing $\zeta \leftarrow \mathbb{Z}_p$ and setting $\mathbf{z} = \tilde{\mathbf{g}}^{-\zeta}$, $\mathbf{r}' = \tilde{\mathbf{g}}^\alpha$, $\mathbf{s}' = \mathbf{g}_z$, $\mathbf{t}' = \tilde{\mathbf{g}}^{-\zeta}$, $\mathbf{u}' = \tilde{\mathbf{g}}^\beta$, $\mathbf{v}' = \mathbf{h}_z$, $\mathbf{w}' = \tilde{\mathbf{g}}^\zeta$ and applying $(\mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w}) \leftarrow \mathsf{SigRand}(\mathbf{r}', \mathbf{s}', \mathbf{t}', \mathbf{u}', \mathbf{v}', \mathbf{w}')$. The rest of the simulation for SIG is the same as that in the proof of Theorem 16. As well as the previous case, the simulation of SIG retains independent of the exponents hidden in $\mathbf{g}_i$ and $\mathbf{h}_i$. Thus a successful forgery contradicts to SDP as shown in the proof of Theorem 12. Finally, applying Theorem 3 to reduce SDP to SFP completes the proof. □

5.6 Applications

We present constructions for round-optimal blind signatures following the framework of [55], the efficient instantiation of which has been an open problem since Crypto'06; and efficient fully secure group signatures supporting concurrent join procedure, with previous constructions being in the random oracle model, secure under weaker model,

not supporting concurrent join procedure, or being inefficient. Our signature schemes not only embody known modular protocol designs, but also achieve an excellent efficiency. These are good examples that enlighten again the usefulness of modular protocol design and significance of developing efficient *structure-preserving* building blocks.

5.6.1 Round-Optimal Blind Signatures

We present an efficient instantiation of Fischlin's round-optimal blind signature scheme [55]. In fact, we use the modification of [73, 4] for which the generic construction uses a non-interactive witness indistinguishable (NIWI) proof system and a simulatable signature scheme. This gives the first efficient round-optimal non-committing blind signature scheme adaptively secure in the universally composability (UC) framework [39].

The structure of the framework is the following. A user commits to a message m with opening $\mathfrak{d}$ and sends the commitment $\mathfrak{c}$ to the signer. The signer signs commitment $\mathfrak{c}$ and returns the signature σ to the user. Then the user computes a NIWI argument π with a witness $(\mathfrak{c}, \mathfrak{d}, \sigma)$ for the fact that he knows a commitment $\mathfrak{c}$ of the message m, he knows the correct opening $\mathfrak{d}$, and he has a valid signature on $\mathfrak{c}$ with respect to a verification key $\mathtt{vk}$ of the signer. The security follows from the generic framework in [4].

To instantiate this generic scheme, we use the GS proof system, the simulatable signature scheme SSIG from Section 5.3 for $k = 1$ (i.e. for signing only a single group element), and the commitment scheme $\mathsf{TC4}$ in Section 4.1.4. In fact, any commitment scheme suffices for our purpose as long as commitment key, commitments, and openings to be group elements and the verification is by pairing product equations.

The choice of $\mathsf{TC4}$ is due to the efficiency; it has the smallest commitment size.

Let $\Lambda \in \{\Lambda_{\mathsf{sym}}, \Lambda_{\mathsf{sxdh}}\}$ be the common parameter. Let $(\mathtt{ck}, \mathtt{tk}) \leftarrow \mathsf{TC4.Key}(1^\lambda)$, $(\mathtt{crs}', \mathtt{tk}') \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$, and $\mathtt{crs}$ be the common reference string for the GS argument system. Concretely, those are $\mathtt{ck} = \tilde{\mathbf{f}} \in \mathbb{G}_2$, $\mathtt{vk} = (\mathbf{g}_z, \mathbf{h}_z, \mathbf{g}_r, \mathbf{h}_u, \mathbf{g}_1, \mathbf{h}_1) \in \mathbb{G}_1^6$, and $\mathtt{crs}$ is set up in the way the simulated CRS is created according to Section 2.4. The CRS for the blind signature scheme is $\Sigma = (\Lambda, \mathtt{ck}, \mathtt{crs}', \mathtt{crs})$. A signer runs $(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathsf{SSIG.Key}(\mathtt{crs}')$ where $\mathtt{vk} = (\mathbf{A}, \mathbf{B}, \sigma^*)$ and publish $\mathtt{vk}$ as his verification key. The blind signature issuing protocols are as follows:

- On input $m \in \mathbb{Z}_p$, a user computes $(\mathfrak{c}, \mathfrak{d}) \leftarrow \mathsf{TC4.Com}(\mathtt{ck}, m)$ where $(\mathfrak{c}, \mathfrak{d}) = (\tilde{\mathbf{g}}^m \tilde{\mathbf{f}}^\delta, \mathbf{g}^\delta) \in \mathbb{G}_2 \times \mathbb{G}_1$. Then the user sends $\mathfrak{c}$ to the signer.

- The signer computes $(\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w}) \leftarrow \mathsf{SSIG.Sign}(\mathtt{sk}, \mathfrak{c})$ and sends back the signature σ to the user.

- The user computes $(\mathbf{r}', \mathbf{s}', \mathbf{t}', \mathbf{u}', \mathbf{v}', \mathbf{w}') \leftarrow \mathsf{SigRand}(\mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ as in Section 5.1.3, and gives a GS argument π with a witness $(\mathfrak{c}, \mathfrak{d}, \mathbf{z}, \mathbf{r}', \mathbf{u}')$ for the pairing product equations

$$\hat{e}(\mathbf{g}, \mathfrak{c})\, \hat{e}(\mathfrak{d}, \tilde{\mathbf{f}}^{-1}) = \hat{e}(\mathbf{g}, \tilde{\mathbf{g}}^m), \tag{5.12}$$

$$\hat{e}(\mathbf{g}_z, \mathbf{z})\, \hat{e}(\mathbf{g}_r, \mathbf{r}')\, \hat{e}(\mathbf{g}_1, \mathfrak{c}) = \mathbf{A} \cdot \hat{e}(\mathbf{s}', \mathbf{t}')^{-1}, \tag{5.13}$$

$$\hat{e}(\mathbf{h}_z, \mathbf{z})\, \hat{e}(\mathbf{h}_u, \mathbf{u}')\, \hat{e}(\mathbf{h}_1, \mathfrak{c}) = \mathbf{B} \cdot \hat{e}(\mathbf{v}', \mathbf{w}')^{-1}. \tag{5.14}$$

  Then output a signature $\sigma_{\mathsf{BS}} = (\mathbf{s}', \mathbf{t}', \mathbf{v}', \mathbf{w}', \pi)$ for m.

Given $(\sigma_{\mathsf{BS}}, m)$, a verifier accepts $\sigma_{\mathsf{BS}} = (\mathbf{s}', \mathbf{t}', \mathbf{v}', \mathbf{w}', \pi)$ if π is a correct GS-proof with respect to relations (5.12), (5.13), and (5.14).

In the construction, the use of $\mathsf{SigRand}$ is for better efficiency and does not affect to the framework due to the nature of perfect randomness. The resulting blind

Scheme	#rounds	Communication	Sig.Size	Sec.Model	Assump.
Oka06[86]	4	$3^{[N^2]}, 4^{[1]}, 10^{[p]}$	$4^{[1]}, 1^{[p]}$	SA	2SDH,DCR
KZ08[79]	6	$9^{[N^2]}, 7^{[1]}, 7^{[p]}$	$4^{[1]}, 1^{[p]}$	UC	2SDH,DCR
Fuc10[56]	2	$22^{[1]}$	$30^{[1]}$	SA	DAHSDH,HDL,DLIN
This Work	2	$8^{[1]}$	$28^{[1]}$	UC	SFP, DLIN

Table 5.1: Summary of efficiency of concurrently secure efficient blind signatures in the standard model. Columns for "Communication" and "Signature Size" count the number of elements, indicating the groups they belong to ($[N^2]$, [1], and $[p]$, respectively, for $\mathbb{Z}_{N^2}$, $\mathbb{G}_1$ and $\mathbb{Z}_p$). UC: Universally Composable Security with Adaptive Corruption [40, 4]. SA: Stand-Alone Security. 2SDH: 2-Variable Strong Diffie-Hellman Assumption [86]. DCR: Decision Composite Residuosity [88]. DAHSDH,HDL: See [56]

signature consists of 4 group elements, 5 GS commitments to group elements, and proof elements for 3 pairing product equations. Note that when $\Lambda = \Lambda_{\mathsf{sym}}$, we could swap the elements in the second pairing of equation (5.12) and get all three equations to be one-sided pairing products. Thus, the size of final blind signature is 28 group elements for $\Lambda = \Lambda_{\mathsf{sym}}$ using GS-proof system with DLIN setting. It can be reduced to 26 group elements (precisely 8 in $\mathbb{G}_1$ and 18 in $\mathbb{G}_2$) for $\Lambda = \Lambda_{\mathsf{sxdh}}$ using GS-proof system in the Λ_{sxdh} setting. The communication complexity is quite low. Only 8 group elements are exchanged in total, and achieves optimal 2 moves. These figures could be a good efficiency standard for "crafted" constructions to compare.

By replacing SSIG with SIG from Section 5.1, one could also instantiate the very original Fischlin's scheme that is secure against static adversaries. This, however, requires NIZK proofs and hence becomes less efficient; NIZK requires that we replace $\mathbf{A}$ and $\mathbf{B}$ with their pairing product representations as originally described for SIG in Section 5.1. We also remark that the construction can be extended to a *partially-blind*

scheme [2] as SSIG (and SIG) can sign multiple group elements at once.

Table 5.1 summaries efficiency of some known blind signature schemes. There are other schemes that achieve concurrent security without random oracles, e.g., [34, 78, 73, 79]. [86] is a representative from those without GS-proofs. Sizes for [86] vary in parameter setting and include some approximation. Numbers for [79] translates numbers in $\mathbb{Z}_{N^3}$ and $\mathbb{Z}_N$ into that of $\mathbb{Z}_{N^2}$ with appropriate factors . (Precisely, $9^{[N^2]}$ is a translation of $1^{[N^3]} + 6^{[N^2]} + 3^{[N]}$.) Our instantiation is very strong in communication while the schemes in [86, 79] with classical blind-then-unblind structure have an advantage in the signature size.

5.6.2 Group Signatures with Concurrent Join

This section highlights a useful property of our signature schemes that the message space is compatible with the verification key space. In particular, we present the most efficient instantiation of a group signature scheme that allows efficient concurrent join protocol [77].

In the symmetric setting $\Lambda = \Lambda_{\mathsf{sym}}$, the message space of USIG1 from Section 5.2 includes the verification key space. This allows Alice to sign Bob's key and Bob can sign Charlie's key and so on. Such a chaining can be hidden by applying NIZK. A signature scheme which is capable of signing its own verification key is introduced as "automorphic" in [56]. It is proven to have some interesting high-level applications such as proxy signatures.

Conceptually, a group signature scheme is a special case of such anonymous delegation system with only one hop of delegation. As sketched in [38] and embodied in [77], the above single-round certification protocol between Alice and Bob brings some favorable properties in the construction of efficient group signature schemes.

The security requirements for a group signature scheme are: *correctness*, which guarantees that the algorithms succeed for honestly behaving users; *anonymity*, which ensures that the signature does not reveal the signer's identity; and *traceability*, which guarantees that all signatures can be traced to a real signer (with *non-frameability* ensuring that that signer participated in the computation of the signature). The second property appears in two flavors in the literature: CPA-anonymity and CCA-anonymity, with the latter providing the adversary with access to an opening (tracing) oracle before and after receiving the challenge signature, whereas for the former the adversary has no such oracle access. For formal definitions and more detailed discussions of the two models refer to [15, 22, 17].

In the following, we revisit the general idea of [38, 77, 67] with CPA-anonymity [22] by using terminology of proof of knowledge. The construction extends to CCA-anonymity by following the generic construction in [67]. Let $\mathsf{SIG0}$ and $\mathsf{SIG1}$ be signature schemes, and Π be a witness indistinguishable proof of knowledge system. A group signature, GSIG, consists of 5 algorithms $\{\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Open}\}$ such that:

- $\mathsf{GSIG.Setup}$ is a setup algorithm that takes security parameter 1^λ and runs $(\mathtt{vk}_\mathrm{c}, \mathtt{sk}_\mathrm{c}) \leftarrow \mathsf{SIG0.Key}(1^\lambda)$ and $(\mathtt{crs}, \mathtt{sk}_\mathrm{o}) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$. Group verification-key is $\mathtt{vk}_\mathrm{g} = (\mathtt{vk}_\mathrm{c}, \mathtt{crs})$. The certification-key $\mathtt{sk}_\mathrm{c}$ is given privately to the issuer and the opening-key $\mathtt{sk}_\mathrm{o}$ is given privately to the opener.

- $\mathsf{GSIG.Join}$ is an interactive protocol between a group member and the issuer. The group member generates his own key-pair $(\mathtt{vk}_\mathrm{u}, \mathtt{sk}_\mathrm{u}) \leftarrow \mathsf{SIG1.Key}(1^\lambda)$ and send $\mathtt{vk}_\mathrm{u}$ to the issuer. The issuer signs on $\mathtt{vk}_\mathrm{u}$ by $\sigma_\mathrm{c} \leftarrow \mathsf{SIG0.Sign}(\mathtt{sk}_\mathrm{c}, \mathtt{vk}_\mathrm{u})$ and send the certificate σ_c to the member.

- $\mathsf{GSIG.Sign}$ is a signing algorithm run by a group member to sign message m.

It consists of signing on message m by $\sigma_{\mathrm{u}} \leftarrow \mathsf{SIG1.Sign}(\mathtt{sk}_{\mathrm{u}}, m)$ and generating a NIWI proof of knowledge $\pi \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, \mathrm{y}, \mathrm{x})$ which proves that $\mathsf{SIG0.Vrf}(\mathtt{vk}_{\mathrm{c}}, \mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}) = 1$ and $\mathsf{SIG1.Vrf}(\mathtt{vk}_{\mathrm{u}}, m, \sigma_{\mathrm{u}}) = 1$ with respect to a witness $\mathrm{x} = (\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}})$ for the statement $\mathrm{y} = (\mathtt{vk}_{\mathrm{c}}, m)$. The final output is π which is a group signature.

- $\mathsf{GSIG.Vrf}$ takes $(\mathtt{vk}_{\mathrm{g}}, m, \pi)$ as input and verifies correctness of π with respect to $\mathrm{y} = (\mathtt{vk}_{\mathrm{c}}, m)$ for the common reference string $\mathtt{crs}$.

- $\mathsf{GSIG.Open}$ is an opening algorithm run by the opener who has opening-key $\mathtt{sk}_{\mathrm{o}}$. Given π and $\mathtt{sk}_{\mathrm{o}}$ as input, the algorithm runs the knowledge extractor of the NIWI proof system and extracts witness $(\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}})$. The exposed verification key $\mathtt{vk}_{\mathrm{u}}$ identifies the group member who actually created π. This algorithm will be associated with another algorithm that publicly verifies the correctness of the opening.

Theorem 22. *Group signature scheme GSIG is CPA-anonymous, traceable, and non-frameable.*

We refer to [22] and [17] for formal definitions of the security notions stated in the theorem. Intuitively, CPA-anonymity is that the adversary cannot distinguish group signatures from two members. As IND-CPA security, the adversary is not given oracle access to the opener. Traceability guarantees that once a group signature is opened, it identifies a group member who once completed $\mathsf{GSIG.Join}$. Non-frameability means that no one but a group member can issue a valid group signature that points to the member if opened.

Proof. CPA-anonymity follows directly from the (computational) WI property [71] of the proof system Π . For traceability, suppose that there is a valid signature π on

message m. Due to the knowledge soundness of Π , the opener can extract $(\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}})$ from π and $(\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}})$ fulfills $1 = \mathsf{SIG0.Vrf}(\mathtt{vk}_{\mathrm{c}}, \mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}})$. If $\mathtt{vk}_{\mathrm{u}}$ does not point any group member registered through $\mathsf{GSIG.Join}$, σ_{c} is a valid forgery for $\mathsf{SIG0}$, which contradicts to the EUF-CMA property of $\mathsf{SIG0}$. Thus $\mathtt{vk}_{\mathrm{u}}$ allows tracing. For non-frameability, suppose that the opener extracts $(\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}})$ from a group signature on message m. If $1 = \mathsf{SIG1.Vrf}(\mathtt{vk}_{\mathrm{u}}, m, \sigma_{\mathrm{u}})$ holds but the owner of $\mathtt{vk}_{\mathrm{u}}$ have never signed on m, it is a valid forgery against $\mathsf{SIG1}$, contradicting the EUF-CMA property of $\mathsf{SIG1}$. □

As mentioned in [77], the above framework has been known without efficient instantiation in the standard model. Using our main signature scheme SIG as $\mathsf{SIG0}$ and GS-proof system as Π , we can instantiate the construction efficiently. We assess the efficiency in the setting $\Lambda = \Lambda_{\mathsf{sym}}$ as follows. Let $\mathsf{SIG1}$ be a signature scheme whose verification key $\mathtt{vk}_{\mathrm{u}}$ and signature σ_{u} consist of α and β group elements, respectively. Let γ be the number of group elements needed to prove relation $1 = \mathsf{SIG1.Vrf}(\mathtt{vk}_{\mathrm{u}}, m, \sigma_{\mathrm{u}})$ including GS-commitments for $\mathtt{vk}_{\mathrm{u}}$ and σ_{u}. Regardless of size $\mathtt{vk}_{\mathrm{u}}$ to be certified, our $\mathsf{SIG0}$ outputs σ_{c} of size 7. Since 4 out of the 7 elements in σ_{c} can be perfectly randomized and given in the clear as we have done in Section 5.6.1, we need only 3 GS-commitments in proving relation $1 = \mathsf{SIG0.Vrf}(\mathtt{vk}_{\mathrm{c}}, \mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}})$, which consists of two one-sided pairing product equations and costs 6 elements in a proof. (Commitments of $\mathtt{vk}_{\mathrm{u}}$ is already included in γ.) In total, this gives a group signature of size $19 + \gamma$. Alternatively, one can instantiate $\mathsf{SIG0}$ with the signature scheme in [44], that has $9\alpha + 4$ elements in σ_{c} and $3\alpha + 3$ one-sided and 3α double-sided pairing product equations in $\mathsf{SIG0.Vrf}$. In that case, the size of a group signature is $63\alpha + 21 + \gamma$.

If we instantiate $\mathsf{SIG1}$ with the full EUF-CMA Boneh-Boyen signature scheme from [21], $\mathtt{vk}_{\mathrm{u}}$ consists of $\alpha = 4$ group elements (including the bases). A signature consists of one group element and one scalar value but the scalar value is totally

Scheme	Concurrent Join	Non-Frameability	Signature Size	Assumptions
BW07[29]	yes	no	$6^{[N]}$	SD, HSDH
Gro07[67]	no	yes	$28^{[1]}$	SDH, q-U, DLIN
GSIG([44]+BB[21])	yes	yes	$297^{[1]}, 1^{[p]}$	DLIN,SDH,HSDH
GSIG(SIG+BB[21])	yes	yes	$43^{[1]}, 1^{[p]}$	SFP, SDH

Table 5.2: Summary of efficiency and properties of group signature schemes with CPA-anonymity in the standard model. The signature size counts the number of elements and indicating the groups they belong to ([1], $[N]$, and $[p]$ respectively for $\mathbb{G}_1$, $\mathbb{Z}_N$, and $\mathbb{Z}_p$). SD: Subgroup Decision Assumption [25]. q-U: See [67].

random and independent of the verification-key. So we have $4 + 1$ GS-commitments in proving $1 = \mathsf{SIG1.Vrf}(\mathtt{vk}_{\mathrm{u}}, m, \sigma_{\mathrm{u}})$. The verification predicate consists of a double-sided pairing product equation, which yields 9 group elements in a proof. In total, we have $\gamma = 24$ and a group signature consists of 43 group elements and 1 scalar value. With [44] for SIG0, the signature size will be 297 group elements and 1 scalar value. These figures can be slightly decreased by using the GS proofs in the SXDH setting.

Table 5.2 summarizes some efficient group signature schemes that provide CPA-anonymity in the standard model with non-interactive assumptions. ([77] allows concurrent join but the security is argued in the random oracle model [16]. A scheme in [9] is non-frameable but only allows sequential join. It bases on strong interactive assumptions.) [29] (and also [28]) provides efficiency with reasonable assumptions but are frameable. The scheme in [67] is non-frameable but does not allow concurrent join as their Join protocol includes 6 rounds of interaction. Also, the traceability of [67] demands a strong dedicated assumption on top of the security of the building blocks. Our construction GSIG(SIG+BB[21]) yields a signature that includes 15 more group

elements than that of [67]. This is the price for achieving concurrent join property and allowing very simple and modular security argument without dedicated assumptions.

One of the advantages of using our SIG for SIG0 is that it allows inserting a warranty in the clear to σ_c so that the signing policy given to a group member is explicit. Due to the constant-size property of SIG, this useful extension can be done without impacting to the size of the group signature (except for the warranty itself) at all.

Finally, point out that our scheme is easily extended for CCA-anonymity using the approach from [67]. This done with the use of a strong one-time signature scheme and a IND-CCA encryption with labels (or selective-tag CCA secure tag-based public-key encryption scheme [80]).

- GSIG.Setup is a setup algorithm that takes security parameter 1^λ and runs $(\mathtt{pk}, \mathtt{sk}_{\mathrm{enc}}) \leftarrow \mathsf{ENC.Key}(1^\lambda)$, $(\mathtt{vk}_c, \mathtt{sk}_c) \leftarrow \mathsf{SIG0.Key}(1^\lambda)$, and $(\mathtt{crs}, \mathtt{sk}_o) \leftarrow \Pi.\mathsf{Crs}(1^\lambda)$. Group verification-key is $\mathtt{vk}_g = (\mathtt{pk}, \mathtt{vk}_c, \mathtt{crs})$. The certification-key $\mathtt{sk}_c$ is given privately to the issuer and the opening-key $\mathtt{sk}_o$ is given privately to the opener.

- GSIG.Join is an interactive protocol between a group member and the issuer. The group member generates his own key-pair $(\mathtt{vk}_u, \mathtt{sk}_u) \leftarrow \mathsf{SIG1.Key}(1^\lambda)$ and send $\mathtt{vk}_u$ to the issuer. The issuer signs on $\mathtt{vk}_u$ by $\sigma_c \leftarrow \mathsf{SIG0.Sign}(\mathtt{sk}_c, \mathtt{vk}_u)$ and send the certificate σ_c to the member.

- GSIG.Sign is a signing algorithm run by a group member to sign message m. It consists of generating one-time key pair $(\mathtt{vk}_{\mathrm{ots}}, \mathtt{sk}_{\mathrm{ots}}) \leftarrow \mathsf{OTS.Key}(1^\lambda)$, signing the one-time verification key $\mathtt{vk}_{\mathrm{ots}}$ by $\sigma_u \leftarrow \mathsf{SIG1.Sign}(\mathtt{sk}_u, \mathtt{vk}_{\mathrm{ots}})$, encrypting σ_u under the public key $\mathtt{pk}$ with a label $\mathtt{vk}_{\mathrm{ots}}$ by $\mathfrak{c} \leftarrow \mathsf{ENC.Enc}^{\mathtt{vk}_{\mathrm{ots}}}(\mathtt{pk}, \sigma_u; r)$, and generating a NIWI proof of knowledge $\pi \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, \mathrm{y}', \mathrm{x}')$ which

proves that $\mathsf{SIG0.Vrf}(\mathtt{vk}_{\mathrm{c}}, \mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}) = 1$ and $\mathsf{SIG1.Vrf}(\mathtt{vk}_{\mathrm{u}}, \mathtt{vk}_{\mathrm{ots}}, \sigma_{\mathrm{u}}) = 1$, and a NIZK proof $\psi \leftarrow \Pi.\mathsf{Prove}(\mathtt{crs}, \mathrm{y}'', \mathrm{x}'')$ that $\mathsf{ENC}^{\mathtt{vk}_{\mathrm{ots}}}(\mathtt{pk}, \sigma_{\mathrm{u}}; r) = \mathfrak{c}$. The joint witness and statement for the proofs are $\mathrm{x} = (\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}}, r)$ and $\mathrm{y} = (\mathtt{pk}, \mathtt{vk}_{\mathrm{c}}, m, \mathtt{vk}_{\mathrm{ots}}, \mathfrak{c})$. Finally, sign everything with the one-time key: $\sigma_{\mathrm{ots}} \leftarrow \mathsf{OTS.Sign}(\mathtt{sk}_{\mathrm{ots}}, (\pi, \psi, \mathfrak{c}, \mathtt{vk}_{\mathrm{ots}}, m))$. Output as the group signature.

- $\mathsf{GSIG.Vrf}$ takes $(\mathtt{vk}_{\mathrm{g}}, m, (\pi, \psi, \mathfrak{c}, \mathtt{vk}_{\mathrm{ots}}, \sigma_{\mathrm{ots}}))$ as input and verifies correctness of the proofs with respect to the right statements as well as $\mathsf{OTS.Vrf}(\mathtt{vk}_{\mathrm{ots}}, (\pi, \psi, \mathfrak{c}, \mathtt{vk}_{\mathrm{ots}}, m), \sigma_{\mathrm{ots}}) = 1$.

- $\mathsf{GSIG.Open}$ is an opening algorithm run by the opener who has the opening-key $\mathtt{sk}_{\mathrm{o}}$. Given a group signature $(\pi, \psi, \mathfrak{c}, \mathtt{vk}_{\mathrm{ots}}, \sigma_{\mathrm{ots}})$, the algorithm runs the knowledge extractor of the NIWI proof system for the proof π and extracts witness $(\mathtt{vk}_{\mathrm{u}}, \sigma_{\mathrm{c}}, \sigma_{\mathrm{u}})$. The exposed verification key $\mathtt{vk}_{\mathrm{u}}$ identifies the group member who actually created π.

This strengthening costs extra 15 group elements in a signature. Accordingly, we have a CCA-anonymous group signature scheme with concurrent join whose signature consists of 58 group elements and one scalar value.

Chapter 6

Structure-Preserving Encryption

We construct a structure-preserving IND-CCA encryption scheme, secure under DLIN in the Λ_{sym} setting. For simplicity, we describe the scheme when encrypting a message that is a single group element in $\mathbb{G}$, but it is easily extended to encrypt a vector of group elements. The scheme shares some similarities with the encryption of Cramer and Shoup [46, 48] and the Linear Cramer-Shoup encryption described by Schacham [99]. Those scheme are usually implemented using cryptographic hash functions, but could also be realized in a hash-free manner by treating a part of the ciphertext as a sequence of bits [46, 48]. However, neither version preserves the underlying algebraic structure, hence the schemes are not structure-preserving.

Then, we use our construction to build a two-party protocol for joint computation of ciphertext. Such protocol previously could not be realized efficiently using the existing encryption schemes. We describe the proof of security through the ideal-world/real-world paradigm, similarly to [36], which can easily be extended to a proof in the more general simulation-based models by Canetti [39] and Küsters [81].

6.1 Structure-Preserving Encryption

Let $\Lambda := (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, \mathbf{g})$. We construct a structure-preserving encryption $\mathsf{ENC} = (\mathsf{ENC.Key}, \mathsf{ENC.Enc}, \mathsf{ENC.Dec})$ which supports labels. For simplicity, we assume that a label $\boldsymbol{\ell}$ is a single group element, but the scheme extends trivially for the case of label being a vector of group elements. Also, labels from the space $\{0,1\}^*$ could be hashed to one or several group elements, but in that case they have to be part of the statement rather than the witness for any NIZK proof.

- $\mathsf{ENC.Key}(1^\lambda)$: Choose random group generators $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \leftarrow \mathbb{G}^*$. For randomly chosen $\vec{x} \leftarrow \mathbb{Z}_p^3$ set $\mathbf{h}_1 = \mathbf{g}_1^{x_1}\mathbf{g}_3^{x_3}$, $\mathbf{h}_2 = \mathbf{g}_2^{x_2}\mathbf{g}_3^{x_3}$. Then, select $\vec{y}_0, \ldots, \vec{y}_5 \leftarrow \mathbb{Z}_p^3$, and compute $\mathbf{f}_{i,1} = \mathbf{g}_1^{y_{i,1}}\mathbf{g}_3^{y_{i,3}}$, $\mathbf{f}_{i,2} = \mathbf{g}_2^{y_{i,2}}\mathbf{g}_3^{y_{i,3}}$, for $i = 0, \ldots, 5$. Output $\mathtt{pk} = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{h}_1, \mathbf{h}_2, \{\mathbf{f}_{i,1}, \mathbf{f}_{i,2}\}_{i=0}^5)$ and $\mathtt{sk} = (\vec{x}, \{\vec{y}\}_{i=0}^5)$.

- $\mathsf{ENC.Enc}^{\boldsymbol{\ell}}(\mathtt{pk}, \mathbf{m})$: Choose random $r, s \leftarrow \mathbb{Z}_p$ and set

$$\mathbf{u}_1 = \mathbf{g}_1^r,\ \mathbf{u}_2 = \mathbf{g}_2^s,\ \mathbf{u}_3 = \mathbf{g}_3^{r+s}, \quad \mathbf{c} = \mathbf{m} \cdot \mathbf{h}_1^r\mathbf{h}_2^s,$$
$$\mathbf{V} = \prod_{i=0}^{3} \hat{e}(\mathbf{f}_{i,1}^r\mathbf{f}_{i,2}^s, \mathbf{u}_i) \cdot \hat{e}(\mathbf{f}_{4,1}^r\mathbf{f}_{4,2}^s, \mathbf{c}) \cdot \hat{e}(\mathbf{f}_{5,1}^r\mathbf{f}_{5,2}^s, \boldsymbol{\ell}),$$

  where $\mathbf{u}_0 = \mathbf{g}$. Output $\mathfrak{c} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{c}, \mathbf{V})$.

- $\mathsf{ENC.Dec}^{\boldsymbol{\ell}}(\mathtt{sk}, \mathfrak{c})$: Parse $\mathfrak{c}$ as $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{c}, \mathbf{V})$. Then check whether

$$\mathbf{V} \stackrel{?}{=} \prod_{i=0}^{3} \hat{e}(\mathbf{u}_1^{y_{i,1}}\mathbf{u}_2^{y_{i,2}}\mathbf{u}_3^{y_{i,3}}, \mathbf{u}_i) \cdot \hat{e}(\mathbf{u}_1^{y_{4,1}}\mathbf{u}_2^{y_{4,2}}\mathbf{u}_3^{y_{4,3}}, \mathbf{c}) \cdot \hat{e}(\mathbf{u}_1^{y_{5,1}}\mathbf{u}_2^{y_{5,2}}\mathbf{u}_3^{y_{5,3}}, \boldsymbol{\ell}),$$

  where $\mathbf{u}_0 = \mathbf{g}$. If unsuccessful, reject the ciphertext as invalid.

  Otherwise, output $\mathbf{m} = \mathbf{c} \cdot (\mathbf{u}_1^{x_1}\mathbf{u}_2^{x_2}\mathbf{u}_3^{x_3})^{-1}$.

Note that using the one-side randomization $\mathsf{RandOneSide}$ from Section 2.5, $\mathbf{V} \in \mathbb{G}_T$ can be replaced by six random group elements $\mathbf{v}_0, \ldots, \mathbf{v}_5 \in \mathbb{G}$ for which the following

equation holds: $\mathbf{V} = \prod_{i=0}^{3} \hat{e}(\mathbf{v}_i, \mathbf{u}_i) \cdot \hat{e}(\mathbf{v}_4, \mathbf{c}) \cdot \hat{e}(\mathbf{v}_5, \boldsymbol{\ell})$. This way, the ciphertext consists only of elements from the base group.

To observe correctness of decryption, note that

$$\begin{aligned} \mathbf{c} \cdot (\mathbf{u}_1^{x_1}\mathbf{u}_2^{x_2}\mathbf{u}_3^{x_3})^{-1} &= \mathbf{m} \cdot \mathbf{h}_1^r \mathbf{h}_2^s \cdot \left((\mathbf{g}_1^r)^{x_1}(\mathbf{g}_2^s)^{x_2}(\mathbf{g}_3^{r+s})^{x_3}\right)^{-1} \\ &= \mathbf{m} \cdot (\mathbf{g}_1^{x_1}\mathbf{g}_3^{x_3})^r(\mathbf{g}_2^{x_2}\mathbf{g}_3^{x_3})^s \cdot \left((\mathbf{g}_1^r)^{x_1}(\mathbf{g}_2^s)^{x_2}(\mathbf{g}_3^{r+s})^{x_3}\right)^{-1} = \mathbf{m}. \end{aligned}$$

The correctness of the validity element $\mathbf{V}$ can be verified similarly. Next we show the IND-CCA security of the scheme.

Theorem 23. *If DLIN holds for Λ_{sym}, the encryption scheme is IND-CCA.*

Proof. We proceed in a sequence of games by modifying $\mathrm{Exp}_{\textit{IND-CCA}}$ in each of them. We start with a game where the experiment is defined as in Definition 4. In that experiment the challenge ciphertext is an encryption of $\mathbf{m}_b$, for a randomly chosen bit b, where $\mathbf{m}_0, \mathbf{m}_1$ are the messages produced by $\mathcal{A}$. In the last game, the experiment produces a challenge ciphertext which is an encryption of a message chosen uniformly at random from the message space. Then, we show that all those games are computationally indistinguishable.

Game 0**:** The experiment is defined according to $\mathrm{Exp}_{\textit{IND-CCA}}$.

Game 1**:** The experiment is modified so that the challenge ciphertext is computed using the "decryption procedure", i.e., $\mathbf{c} = \mathbf{m} \cdot \mathbf{u}_1^{x_1}\mathbf{u}_2^{x_2}\mathbf{u}_3^{x_3}$ and $\mathbf{V}$ is computed the way it is verified. The change is only syntactical, so the two games are identical from $\mathcal{A}$'s perspective.

Game 2**:** The randomness vector $\vec{\mathbf{u}} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ of the challenge ciphertext is computed as non-DLIN tuple, i.e., $\mathbf{u}_1 = \mathbf{g}_1^r$, $\mathbf{u}_2 = \mathbf{g}_2^s$, $\mathbf{u}_3 = \mathbf{g}_3^t$ where $r, s, t \leftarrow \mathbb{Z}_p$ and $r + s \neq t$. The experiments in Game 1 and Game 2 are indistinguishable by

DLIN. *Note that all decryption queries with ciphertext which has a randomness vector a DLIN tuple yield a unique plaintext and do not reveal any information about the secret key.*

Game 3**:** All decryption queries with ciphertext $\mathfrak{c}$ which has a non-DLIN randomness vector $\vec{\mathbf{u}}$ are rejected. Consider the two possible cases:

- $(\vec{\mathbf{u}}^\star, \mathbf{c}^\star, \boldsymbol{\ell}^\star) = (\vec{\mathbf{u}}, \mathbf{c}, \boldsymbol{\ell})$. Such decryption query is rejected because it is either the challenge ciphertext (when $\mathbf{V} = \mathbf{V}^\star$) or the verification predicate fails trivially (when $\mathbf{V} \neq \mathbf{V}^\star$). This case is the same in the previous game.
- otherwise, when $(\vec{\mathbf{u}}^\star, \mathbf{c}^\star, \boldsymbol{\ell}^\star) \neq (\vec{\mathbf{u}}, \mathbf{c}, \boldsymbol{\ell})$. By Lemma 7, such decryption query is always rejected in Game 2 except for a negligible probability, whereas in Game 3 it is always rejected.

As the number of decryption queries is polynomial the experiments in Game 2 and Game 3 are indistinguishable except with a negligible probability.

Game 4: The challenge ciphertext encrypts a random message from the message space. Game 3 and Game 4 are (information theoretically) indistinguishable by Lemma 6.

In the last game, the challenge bit b is independent from the ciphertext, so the experiment returns 1 with probability $\frac{1}{2}$. By the indistinguishability of the consecutive games, $\mathcal{A}$ wins $\mathrm{Exp}_{\textit{IND-CCA}}$ with probability $\frac{1}{2} + \mathsf{negl}(\lambda)$. □

In the next two lemmas, let $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \leftarrow \mathbb{G}^*$ and $\mathbf{u}_1 = \mathbf{g}_1^r$, $\mathbf{u}_2 = \mathbf{g}_2^s$, $\mathbf{u}_3 = \mathbf{g}_3^t$, where r, s, t are randomly chosen from $\mathbb{Z}_p$ such that $r + s \neq t$. And for convenience, let us denote with z_1, z_2, z_3 the discrete logarithms of $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ with base $\mathbf{g}$.

Lemma 6. *For randomly chosen* $\vec{x} \leftarrow \mathbb{Z}_p^3$, *let* $\mathbf{h}_1 = \mathbf{g}_1^{x_1}\mathbf{g}_3^{x_3}$, $\mathbf{h}_2 = \mathbf{g}_2^{x_2}\mathbf{g}_3^{x_3}$, *and* $\pi = \mathbf{u}_1^{x_1}\mathbf{u}_2^{x_2}\mathbf{u}_3^{x_3}$. *Then for a randomly chosen* $\psi \leftarrow \mathbb{G}$ *it is true that* $(\mathbf{h}_1, \mathbf{h}_2, \pi) \equiv (\mathbf{h}_1, \mathbf{h}_2, \psi)$, *where "*$\equiv$*" denotes distributional equivalence.*

Proof. Note that $\mathbf{h}_1 = \mathbf{g}^{x_1 z_1 + x_3 z_3}$ and $\mathbf{h}_2 = \mathbf{g}^{x_2 z_2 + x_3 z_3}$. Then, for the tuple $(\mathbf{h}_1, \mathbf{h}_2, \pi)$ the following equation holds:

$$\begin{pmatrix} z_1 & 0 & z_3 \\ 0 & z_2 & z_3 \\ rz_1 & sz_2 & tz_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \mathrm{dlog}_{\mathbf{g}}(\mathbf{h}_1) \\ \mathrm{dlog}_{\mathbf{g}}(\mathbf{h}_2) \\ \mathrm{dlog}_{\mathbf{g}}(\pi) \end{pmatrix}$$

Denote the leftmost matrix by M. It has a determinant $\det(M) = z_1 z_2 z_3 (t - r - s)$ which is not equal to 0 due to the choice of the parameters. Therefore the matrix is invertible and for any $\pi \in \mathbb{G}$, and fixed $\mathbf{h}_1, \mathbf{h}_2$, there exist a unique $\vec{x}$ which yields the tuple $(\mathbf{h}_1, \mathbf{h}_2, \pi)$. □

Lemma 7. *Let* $\vec{\mathbf{u}}^\star = (\mathbf{u}_1^\star, \mathbf{u}_2^\star, \mathbf{u}_2^\star)$ *be any tuple such that* $\mathbf{u}_1^\star = \mathbf{g}_1^{r^\star}$, $\mathbf{u}_2^\star = \mathbf{g}_2^{s^\star}$, *and* $\mathbf{u}_3^\star = \mathbf{g}_3^{t^\star}$, *for* $r^\star + s^\star \neq t^\star$. *And for randomly chosen* $\vec{y}_0, \vec{y}_1, \ldots, \vec{y}_5 \leftarrow \mathbb{Z}_p^3$, *let* $\mathbf{f}_{i,1} = \mathbf{g}_1^{y_{i,1}}\mathbf{g}_3^{y_{i,3}}$, $\mathbf{f}_{i,2} = \mathbf{g}_2^{y_{i,2}}\mathbf{g}_3^{y_{i,3}}$, *where* $i = 0, \ldots, 5$. *For* $\vec{\mathbf{m}}$ *and* $\vec{\mathbf{m}}^\star$ *in* $\mathbb{G}^5$, *let* π *and* $\pi^\star$ *denote*

$$\pi = \prod_{i=0}^{5} \hat{e}(\mathbf{u}_1^{y_{i,1}}\mathbf{u}_2^{y_{i,2}}\mathbf{u}_3^{y_{i,3}}, \mathbf{m}_i) \quad \textit{and} \quad \pi^\star = \prod_{i=0}^{5} \hat{e}((\mathbf{u}_1^\star)^{y_{i,1}}(\mathbf{u}_2^\star)^{y_{i,2}}(\mathbf{u}_3^\star)^{y_{i,3}}, \mathbf{m}_i^\star),$$

where $\mathbf{m}_0 = \mathbf{m}_0^\star = \mathbf{g}$. *Then, for any* $\vec{\mathbf{m}}$ *and* $\vec{\mathbf{m}}^\star$, $\vec{\mathbf{m}} \neq \vec{\mathbf{m}}^\star$, *the following distributional equivalence hold:* $(\{\mathbf{f}_{i,1}\mathbf{f}_{i,2}\}_{i=0}^5, \pi, \pi^\star) \equiv (\{\mathbf{f}_{i,1}\mathbf{f}_{i,2}\}_{i=0}^5, \pi, \psi)$, *where* $\psi \leftarrow \mathbb{G}_T$.

Proof. Similarly to the proof of the previous lemma, let us define all variables which depend on $\{\vec{y}_0\}$ as result of a constant matrix M multiplied by the vector $(\vec{y}_0 || \vec{y}_1 || \ldots || \vec{y}_5)$. For convenience, denote with $w_i = \mathrm{dlog}_{\mathbf{g}}(\mathbf{m}_i)$ and $w_i^\star = \mathrm{dlog}_{\mathbf{g}}(\mathbf{m}_i^\star)$,

for $i = 1, \ldots, 5$. Then, for a matrix

$$M = \begin{pmatrix} z_1 & 0 & z_3 & - & - & - & \ldots & - & - & - \\ 0 & z_2 & z_3 & - & - & - & \ldots & - & - & - \\ - & - & - & z_1 & 0 & z_3 & \ldots & - & - & - \\ - & - & - & 0 & z_2 & z_3 & \ldots & - & - & - \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ - & - & - & - & - & - & \ldots & z_1 & 0 & z_3 \\ - & - & - & - & - & - & \ldots & 0 & z_2 & z_3 \\ rz_1 & sz_2 & tz_3 & w_1 rz_1 & w_1 sz_2 & w_1 tz_3 & \ldots & w_5 rz_1 & w_5 sz_2 & w_5 tz_3 \\ r^\star z_1 & s^\star z_2 & t^\star z_3 & w_1^\star r^\star z_1 & w_1^\star s^\star z_2 & w_1^\star t^\star z_3 & \ldots & w_5^\star r^\star z_1 & w_5^\star s^\star z_2 & w_5^\star t^\star z_3 \end{pmatrix},$$

we have

$$M \cdot \begin{pmatrix} | \\ \vec{y}_0 \\ | \\ \vdots \\ | \\ \vec{y}_5 \\ | \end{pmatrix} = \begin{pmatrix} \mathrm{dlog}_{\mathbf{g}}(\mathbf{f}_{0,1}) \\ \mathrm{dlog}_{\mathbf{g}}(\mathbf{f}_{0,2}) \\ \vdots \\ \mathrm{dlog}_{\mathbf{g}}(\mathbf{f}_{5,1}) \\ \mathrm{dlog}_{\mathbf{g}}(\mathbf{f}_{5,2}) \\ \mathrm{dlog}_{\hat{e}(\mathbf{g},\mathbf{g})}(\pi) \\ \mathrm{dlog}_{\hat{e}(\mathbf{g},\mathbf{g})}(\pi^\star) \end{pmatrix}.$$

We would like to argue that the rows of the matrix M are linearly independent. As there exist i such that $\mathbf{m}_i \neq \mathbf{m}_i^\star$, then if we choose the sub-matrix M' consisting of the intersection of the last two rows and rows 1, 2, $2i+1$, $2i+2$ with columns

$1,\ 2,\ 3,\ 3i+1,\ 3i+2,\ 3i+3$, we get:

$$M' = \begin{pmatrix} z_1 & 0 & z_3 & 0 & 0 & 0 \\ 0 & z_2 & z_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_1 & 0 & z_3 \\ 0 & 0 & 0 & 0 & z_2 & z_3 \\ rz_1 & sz_2 & tz_3 & w_i r z_1 & w_i s z_2 & w_i t z_3 \\ r^\star z_1 & s^\star z_2 & t^\star z_3 & w_i^\star r^\star z_1 & w_i^\star s^\star z_2 & w_i^\star t^\star z_3 \end{pmatrix}.$$

If the rows of M are not linearly independent, so are the rows of M'. However, M' has a determinant $\det(M') = \pm z_1^2 z_2^2 z_3^2 (w - w^\star)(t - r - s)(t^\star - r^\star - s^\star)$ which is not equal to 0 due to choice of the parameters. Therefore, the rows of M are linearly independent. □

6.2 Application: Joint Computation of Ciphertext

We consider a two-party protocol, each party in possession of some secret value, for a joint computation of a ciphertext of a function of the two secrets, encrypted under a third-party public key `pk`. We study the case where only the first party learns the ciphertext whereas the second has no output. We assume that the public inputs, i.e, the function, the third-party public key, and commitments of the secrets, are agreed upon prior to the execution of the protocol.

The idea of the protocol is that the first party computes a partial and blinded encryption of their secret and sends it to the other party. Also, the first party proves that the computation is carried correctly. The second party takes the values from the first flow of the protocol and using its secret and some randomness computes a blinded full encryption of the agreed function of the two secret keys. Then, the second party sends those values and proves that they are computed correctly. Finally, the

$\mathcal{P}_1$'s input:

$\underline{x_1,\ w_1,\ \mathbf{a}^{x_2}\mathbf{b}^{w_2}, \mathtt{pk}}$

$\bar{\mathbf{u}}'_1 = \mathbf{g}^{t_1} \cdot \mathbf{g}_1^{r_1},$

$\bar{\mathbf{u}}'_2 = \mathbf{g}^{t_2} \cdot \mathbf{g}_2^{s_1},$

$\bar{\mathbf{u}}'_3 = \mathbf{g}^{t_3} \cdot \mathbf{g}_3^{r_1+s_1},$

$\bar{\mathbf{u}}'_4 = \mathbf{g}^{t_4} \cdot \mathbf{a}^{x_1} \cdot \mathbf{h}_1^{r_1}\mathbf{h}_2^{s_1},$

$\bar{\mathbf{V}}'_1 = \prod_{i=1}^{4} \hat{e}(\mathbf{f}_{i,1}, \mathbf{g}^{t_i}) \cdot \hat{e}(\mathbf{g}_1, \mathbf{g}^{t_5}),$

$\bar{\mathbf{V}}'_2 = \prod_{i=1}^{4} \hat{e}(\mathbf{f}_{i,2}, \mathbf{g}^{t_i}) \cdot \hat{e}(\mathbf{g}_2, \mathbf{g}^{t_6}),$

$\underline{\bar{\mathbf{u}}'_1,\ \bar{\mathbf{u}}'_2,\ \bar{\mathbf{u}}'_3,\ \bar{\mathbf{u}}'_4,\ \bar{\mathbf{V}}'_1,\ \bar{\mathbf{V}}'_2} \rightarrow$

$\underline{\text{proof of correct comp.}\pi_1} \rightarrow$

$\mathcal{P}_2$'s input:

$\underline{x_2,\ w_2,\ \mathbf{a}^{x_1}\mathbf{b}^{w_1}, \mathtt{pk}}$

$\bar{\mathbf{u}}_0 = \mathbf{g},$

$\bar{\mathbf{u}}_1 = \bar{\mathbf{u}}'_1 \cdot \mathbf{g}_1^{r_2},$

$\bar{\mathbf{u}}_2 = \bar{\mathbf{u}}'_2 \cdot \mathbf{g}_2^{s_2},$

$\bar{\mathbf{u}}_3 = \bar{\mathbf{u}}'_3 \cdot \mathbf{g}_3^{r_2+s_2},$

$\bar{\mathbf{u}}_4 = \bar{\mathbf{u}}'_4 \cdot \mathbf{a}^{x_2} \cdot \mathbf{h}_1^{r_2}\mathbf{h}_2^{s_2},$

$\bar{\mathbf{V}} = \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}, \bar{\mathbf{u}}_i)/\bar{\mathbf{V}}'_1\right)^{r_2} \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,2}, \bar{\mathbf{u}}_i)/\bar{\mathbf{V}}'_2\right)^{s_2},$

$\underline{\bar{\mathbf{u}}_1,\ \bar{\mathbf{u}}_2,\ \bar{\mathbf{u}}_3,\ \bar{\mathbf{u}}_4,\ \bar{\mathbf{V}}} \leftarrow$

$\underline{\text{proof of correct comp.}\pi_2} \leftarrow$

$\mathbf{u}_0 = \mathbf{g},$

$\mathbf{u}_1 = \bar{\mathbf{u}}_1/\mathbf{g}^{t_1} = \mathbf{g}_1^{r}$

$\mathbf{u}_2 = \bar{\mathbf{u}}_2/\mathbf{g}^{t_2} = \mathbf{g}_2^{s}$

$\mathbf{u}_3 = \bar{\mathbf{u}}_3/\mathbf{g}^{t_3} = \mathbf{g}_3^{r+s}$

$\mathbf{u}_4 = \bar{\mathbf{u}}_4/\mathbf{g}^{t_4} = \mathbf{a}^{x_1+x_2} \cdot \mathbf{h}_1^{r}\mathbf{h}_2^{s}$

$\mathbf{V} = \bar{\mathbf{V}}\ \hat{e}(\mathbf{u}_1\mathbf{g}_1^{-r_1}, \mathbf{g}^{t_5})\ \hat{e}(\mathbf{u}_2\mathbf{g}_2^{-s_1}, \mathbf{g}^{t_6})$
$\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}^{r_1}\mathbf{f}_{i,2}^{s_1}, \mathbf{u}_i)$

Figure 6.1: Joint ciphertext computation.

first party unblinds the ciphertext and updates the consistency element to obtain a valid encryption of the function of the two secrets under jointly chosen randomness. The function can be a constant to the power of any polynomial of the two secrets; for simplicity, we consider the function $\mathbf{a}^{x_1+x_2}$ where $\mathbf{a}$ is a fixed group element and x_1, x_2 are the two secrets.

Recall the structure of the ciphertext of the public-key encryption scheme described in Section 6.1: for a public key $\mathtt{pk} = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{h}_1, \mathbf{h}_2, \{\mathbf{f}_{i,1}, \mathbf{f}_{i,2}\}_{i=0}^4)$ and randomly chosen $r, s \leftarrow \mathbb{Z}_p$, the ciphertext is computed as

$$(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4, \mathbf{V}) = \left(\mathbf{g}_1^r,\ \mathbf{g}_2^s,\ \mathbf{g}_3^{r+s},\quad \mathbf{m} \cdot \mathbf{h}_1^r \mathbf{h}_2^s,\quad \prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}^r \mathbf{f}_{i,2}^s, \mathbf{u}_i) \right), \text{where } \mathbf{u}_0 = \mathbf{g}.$$

Recall that the encryption supports labels, and our protocol can easily be modified to support labels, but we omit that in favor of a clearer presentation.

Note that the protocol on Figure 6.1 computes a valid ciphertext because $\mathbf{u}_1 = \mathbf{g}_1^r$ for $r = r_1 + r_2$, $\mathbf{u}_2 = \mathbf{g}_2^s$ for $s = s_1 + s_2$, $\mathbf{u}_3 = \mathbf{g}_3^{r+s}$, $\mathbf{u}_4 = \mathbf{m} \cdot \mathbf{h}_1^r \mathbf{h}_2^s$ for $\mathbf{m} = \mathbf{a}^{x_1+x_2}$, and $\mathbf{V} = \prod_{i=0}^4 \hat{e}(\mathbf{f}_{i,1}^r \mathbf{f}_{i,2}^s, \mathbf{u}_i)$. To see $\mathbf{V}$ is indeed computed this way, note that:

$$\bar{\mathbf{V}} = \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}, \bar{\mathbf{u}}_i) / \bar{\mathbf{V}}_1' \right)^{r_2} \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,2}, \bar{\mathbf{u}}_i) / \bar{\mathbf{V}}_2' \right)^{s_2} = \frac{\prod_{i=0}^4 \hat{e}(\mathbf{f}_{i,1}^{r_2} \mathbf{f}_{i,2}^{s_1}, \mathbf{u}_i)}{\hat{e}(\mathbf{g}_1, \mathbf{g}^{t_5})^{r_2}\ \hat{e}(\mathbf{g}_2, \mathbf{g}^{t_6})^{s_2}}$$

and

$$\bar{\mathbf{V}}\ \hat{e}\left(\frac{\mathbf{u}_1}{\mathbf{g}_1^{r_1}}, \mathbf{g}^{t_5} \right)\ \hat{e}\left(\frac{\mathbf{u}_2}{\mathbf{g}_2^{s_1}}, \mathbf{g}^{t_6} \right) = \bar{\mathbf{V}}\ \hat{e}(\mathbf{g}_1^{r_2}, \mathbf{g}^{t_5})\ \hat{e}(\mathbf{g}_2^{s_2}, \mathbf{g}^{t_6}) = \prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}^{r_2} \mathbf{f}_{i,2}^{s_2}, \mathbf{u}_i).$$

The security of the protocol is shown using the real-world/ideal-world paradigm. In the ideal experiment, the joint computation is performed by a trusted party to which the two parties simply give their secrets, and the trusted party carries the computation and returns the corresponding outputs. In the real experiment, the two parties execute the described protocol. The security requirement is that for any misbehaving party $\mathcal{A}$ in the real experiment, there is a corresponding misbehaving

party $\mathcal{B}$ in the ideal experiment, for which no efficient distinguisher can tell between the two experiments. But before proceeding with the security proof, let us see how π_1 and π_2 are implemented

As we will see later, some elements of the witness for π_1 and π_2 have to be extractable in order security to hold. However, the witness consists only of exponents, and if one wants to uses GS proofs, each exponent would have to be committed bit-by-by (and proof equations adjusted) in order to be able to extract it. However, this would be costly. Given that the protocol is interactive by nature, we favor the efficient ZK proofs of knowledge using "sigma-protocols".

The proofs of correct computation are carried using the notation and implementation from [30] which extend trivially to the bilinear setting when the bases are publicly known group elements. The proofs π_1 and π_2 are computed as follows:

$$
\begin{aligned}
\pi_1 = \exists\, & t_1, \ldots, t_4 \nmid x_1, r_1, s_1, t_5, t_6 : \\
& \bar{\mathbf{u}}'_1 = \mathbf{g}^{t_1} \cdot \mathbf{g}_1^{r_1} \ \wedge\ \bar{\mathbf{u}}'_2 = \mathbf{g}^{t_2} \cdot \mathbf{g}_2^{s_1} \ \wedge\ \bar{\mathbf{u}}'_3 = \mathbf{g}^{t_3} \cdot \mathbf{g}_3^{r_1+s_1} \ \wedge \\
& \bar{\mathbf{u}}'_4 = \mathbf{g}^{t_4} \cdot \mathbf{a}^{x_1} \cdot \mathbf{h}_1^{r_1}\mathbf{h}_2^{s_1} \ \wedge \\
& \bar{\mathbf{V}}'_1 = \prod_{i=1}^{4} \hat{e}(\mathbf{f}_{i,1}, \mathbf{g}^{t_i}) \cdot \hat{e}(\mathbf{g}_1, \mathbf{g}^{t_5}) \ \wedge\ \bar{\mathbf{V}}'_2 = \prod_{i=1}^{4} \hat{e}(\mathbf{f}_{i,2}, \mathbf{g}^{t_i}) \cdot \hat{e}(\mathbf{g}_2, \mathbf{g}^{t_6})
\end{aligned}
$$

and

$$
\begin{aligned}
\pi_2 = \exists\, & r_2, s_2 \nmid x_2 : \\
& \bar{\mathbf{u}}_1 = \bar{\mathbf{u}}'_1 \cdot \mathbf{g}_1^{r_2} \ \wedge\ \bar{\mathbf{u}}_2 = \bar{\mathbf{u}}'_2 \cdot \mathbf{g}_2^{s_2} \ \wedge\ \bar{\mathbf{u}}_3 = \bar{\mathbf{u}}'_3 \cdot \mathbf{g}_3^{r_2+s_2} \ \wedge \\
& \bar{\mathbf{u}}_4 = \bar{\mathbf{u}}'_4 \cdot \mathbf{a}^{x_2} \cdot \mathbf{h}_1^{r_2}\mathbf{h}_2^{s_2} \ \wedge \\
& \bar{\mathbf{V}} = \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,1}, \bar{\mathbf{u}}_i) / \bar{\mathbf{V}}'_1 \right)^{r_2} \left(\prod_{i=0}^{4} \hat{e}(\mathbf{f}_{i,2}, \bar{\mathbf{u}}_i) / \bar{\mathbf{V}}'_2 \right)^{s_2},
\end{aligned}
$$

where $\bar{\mathbf{u}}_0 = \mathbf{g}$.

To prove security of the above described protocol, we show that for any adversary

$\mathcal{A}$ attacking the real protocol, there exist an adversary $\mathcal{B}$ which attacks the ideal one. As usual, $\mathcal{B}$ runs the real protocol with an instance of $\mathcal{A}$ internally and simulates the protocol execution. The two cases to be considered for the security proof are when $\mathcal{P}_1$ is corrupted and $\mathcal{P}_2$ is honest, and vice versa.

For the case when $\mathcal{P}_1$ is corrupted by $\mathcal{A}$, in the first step $\mathcal{B}$ receives $\bar{\mathbf{u}}'_1,\ \bar{\mathbf{u}}'_2,\ \bar{\mathbf{u}}'_3,\ \bar{\mathbf{u}}'_4,\ \bar{\mathbf{V}}'_1,\ \bar{\mathbf{V}}'_2$ as well as π_1. Using the property of the proof system, $\mathcal{B}$ extracts x_1, r_1, s_1, t_5, t_6, and computes $\mathbf{g}^{t_1}, \ldots, \mathbf{g}^{t_4}$ from the elements received from $\mathcal{A}$. Then, $\mathcal{B}$ hands over x_1 to the trusted party and receives back $(\mathbf{u}^\star_1, \mathbf{u}^\star_2, \mathbf{u}^\star_3, \mathbf{u}^\star_4, \mathbf{V}^\star)$ which is the ciphertext to be computed at the end by $\mathcal{P}_1$. $\mathcal{B}$ computes:

$$\bar{\mathbf{u}}_1 = \mathbf{u}^\star_1 \cdot \mathbf{g}^{t_1},\ \bar{\mathbf{u}}_2 = \mathbf{u}^\star_2 \cdot \mathbf{g}^{t_2},\ \bar{\mathbf{u}}_3 = \mathbf{u}^\star_3 \cdot \mathbf{g}^{t_3},\ \bar{\mathbf{u}}_4 = \mathbf{u}^\star_4 \cdot \mathbf{g}^{t_4}, \text{ and}$$
$$\bar{\mathbf{V}} = \mathbf{V}^\star \cdot \left(\hat{e}\left(\frac{\mathbf{u}^\star_1}{\mathbf{g}_1^{r_1}}, \mathbf{g}^{t_5}\right)\ \hat{e}\left(\frac{\mathbf{u}^\star_2}{\mathbf{g}_2^{s_1}}, \mathbf{g}^{t_6}\right) \prod_{i=0}^{4} \hat{e}\left(\mathbf{f}_{i,1}^{r_1}\mathbf{f}_{i,2}^{s_1}, \mathbf{u}^\star_i\right) \right)^{-1},$$

and sends those to $\mathcal{P}_1$. The proof π_2 is simulated using the ZK property of the proof system.

In the case when $\mathcal{P}_2$ is corrupt, $\mathcal{B}$ chooses random $\bar{\mathbf{u}}'_1, \bar{\mathbf{u}}'_2, \bar{\mathbf{u}}'_3, \bar{\mathbf{u}}'_4 \leftarrow \mathbb{G}$ and $\bar{\mathbf{V}}' \leftarrow \mathbb{G}_T$, and delivers those to together with simulated proof π_1. In the next step, $\mathcal{B}$ receives from $\mathcal{P}_2$ the values $\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_3, \bar{\mathbf{u}}_4, \bar{\mathbf{V}}$ as well as a proof π_2 from which $\mathcal{B}$ extracts x_2. Finally, $\mathcal{B}$ submits x_2 to the trusted party.

So, using interactive ZK proofs of knowledge, borrowing techniques from [30], which require the strong RSA assumption [11] and the DCR assumption [88] to hold for the appropriate additional groups, the two-party protocol described in Figure 6.1 is secure if DLIN holds for $\mathbb{G}$.

Bibliography

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology - CRYPTO 2010*, LNCS, pages 209–237. Springer-Verlag, 2010.

[2] M. Abe and E. Fujisaki. How to date blind signatures. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96*, volume 1163 of *LNCS*, pages 244–251. Springer-Verlag, 1996.

[3] M. Abe, K. Haralambiev, and M. Ohkubo. Signing on group elements for modular protocol designs. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org`.

[4] M. Abe and M. Ohkubo. A framework for universally composable non-committing blind signatures. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 435–450. Springer-Verlag, 2009.

[5] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[6] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[7] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[8] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, pages 591–606, 1998.

[9] G. Ateniese, J. Camenisch, S. hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org`.

[10] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer-Verlag, 2000.

[11] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 480–494. Springer-Verlag, 1997.

[12] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO*, pages 108–125, 2009.

[13] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Non-interactive anonymous credentials. In R. Canetti, editor, *Theory of Cryptography,*

Fifth Theory of Cryptography Conference, TCC 2008, volume 4948 of *LNCS*. Springer-Verlag, 2008. Also available on IACR ePrint Archive, 2007/384.

[14] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In *Public Key Cryptography*, pages 85–99, 2003.

[15] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptology - EUROCRPYT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer-Verlag, 2003.

[16] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.

[17] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–154. Springer-Verlag, 2005. Full version available at IACR e-print 2004/077.

[18] M. Bellare and S. Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In *Proceedings of the 10th International Conference on Theory and Practice of Public-Key Cryptography - PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer-Verlag, 2007.

[19] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the* 20*th annual ACM Symposium on the Theory of Computing*, pages 103–112. ACM, 1988.

[20] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption. In *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004.

[21] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — Eurocrypt '04*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.

[22] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.

[23] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – Crypto 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[24] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer-Verlag, 2003.

[25] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *Theory of Cryptography Conference– TCC'2005*, volume 3378 of *LNCS*, pages 325–341. Springer-Verlag, 2005.

[26] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

[27] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In C. Boyd, editor, *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, 2001.

[28] X. Boyen and B. Waters. Compact group signatures without random oracles. In *Advances in Cryptology — Eurocrypt '06*, volume 4004 of *LNCS*, pages 427–444. Springer-Verlag, 2006. Full version available from IACR ePrint Archive 2005/381.

[29] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography—PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer-Verlag, 2007. Available at `http://www.cs.stanford.edu/~xb/pkc07/`.

[30] J. Camenisch, N. Casati, T. Groß, and V. Shoup. Credential authenticated identification and key exchange. In *CRYPTO*, pages 255–276, 2010.

[31] J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer-Verlag, 2009.

[32] J. Camenisch, K. Haralambiev, M. Kohlweiss, and J. Lapor. Accountability without subliminal influence. unpublished manuscript, 2011.

[33] J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography, PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer-Verlag, 2009.

[34] J. Camenisch, M. Koprowski, and B. Warinschi. Efficient blind signatures without random oracles. In C. Blundo and S. Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, volume 3352 of *LNCS*, pages 134–148. Springer-Verlag, 2005.

[35] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag, 2004.

[36] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.

[37] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144, 2003.

[38] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 410–424. Springer-Verlag, 1997.

[39] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the* 42*nd IEEE Annual Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[40] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Technical Report 2000/067, IACR e-print Archive, 2005. 2nd version updated on 13 Dec 2005.

[41] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the* 30*th annual ACM Symposium on Theory of Computing*, pages 209–218, 1998.

[42] R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing chosen-ciphertext security. Technical Report 2003/174, IACR ePrint archive, 2003. Conference version appeared in CRYPTO 2003.

[43] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the* 34*th annual ACM Symposium on Theory of Computing*, pages 494–503. ACM, 2002.

[44] J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 179–196. Springer-Verlag, 2009.

[45] D. Chaum and E. V. Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, 1991.

[46] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[47] R. Cramer and V. Shoup. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[48] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[49] I. Damg**s***a*rd and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer-Verlag, 2002.

[50] I. Damg**s***a*rd and J. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *LNCS*, pages 581–596. Springer-Verlag, 2002.

[51] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. Cryptology ePrint Archive, Report 2010/154, 2010. `http://eprint.iacr.org/`.

[52] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[53] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the 23rd annual ACM Symposium on the Theory of Computing*, pages 542–552, New York City, 1991.

[54] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st IEEE Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE, 1990.

[55] M. Fischlin. Round-optimal composable blind signatures in the common reference model. In C. Dwork, editor, *Advances in Cryptology — CRYPTO '06*, volume 4117 of *LNCS*, pages 60–77. Springer-Verlag, 2006.

[56] G. Fuchsbauer. Automorphic signatures in bilinear groups. IACR ePrint Archive 2009/320. Revised March 17, 2010., 2010.

[57] G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Transferable anonymous constant-size fair e-cash. IACR ePrint Archive 2009/146. Also to appear in CANS 2009., 2009.

[58] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30. Springer-Verlag, 1997.

[59] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Technical Report 2006/165, IACR ePrint archive, 2006.

[60] S. D. Galbraith and V. Rotger. Easy decision-diffie-hellman groups. *LMS Journal of Computation and Mathematics*, 7:2004, 2004.

[61] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[62] E. Ghadafi, N. P. Smart, and B. Warinschi. Groth-sahai proofs revisited. In *Public Key Cryptography*, pages 177–192, 2010.

[63] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[64] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[65] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In J. Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer-Verlag, 2008. Preliminary version: IACR ePrint Archive 2008/163.

[66] J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer-Verlag, 2006.

[67] J. Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology – Asiacrypt'07*, volume 4833 of *LNCS*, pages 164–180. Springer-Verlag, 2007.

[68] J. Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, January 2009. Update version available from the author's homepage.

[69] J. Groth. Homomorphic trapdoor commitments to group elements. Unpublished Manuscript, 2010.

[70] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer-Verlag, 2006.

[71] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology — Eurocrypt '08*, volume 4965 of *LNCS*, pages 415–432. Springer-Verlag, 2008. Full version available: IACR ePrint Archive 2007/155.

[72] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Günther, editor, *Advances in Cryptology — EUROCRYPT '88*, volume 330 of *LNCS*, pages 123–128. Springer-Verlag, 1988.

[73] C. Hazay, J. Katz, C. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *Theory of Cryptography Conference, TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer-Verlag, 2007.

[74] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer-Verlag, 2007.

[75] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[76] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *LNCS*, pages 198–214. Springer-Verlag, 2005.

[77] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In R. Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 198–214. Springer-Verlag, 2005.

[78] A. Kiayias and H. Zhou. Concurrent blind signatures without random oracles. In *SCN 2006*, volume 4116 of *LNCS*, pages 49–62. Springer-Verlag, 2006.

[79] A. Kiayias and H. Zhou. Equivocal blind signatures and adaptive uc-security. In R. Canetti, editor, *Theory of Cryptography Conference, TCC 2008*, volume 4948 of *LNCS*, pages 340–355. Springer-Verlag, 2008.

[80] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *Theory of Cryptography Conference – TCC'06*, volume 3876 of *LNCS*, pages 581–600. Springer-Verlag, 2006.

[81] R. Kusters. Simulation-based security with inexhaustible interactive turing machines. In *Computer Security Foundations Workshop, 2006. 19th IEEE*, pages 12–320. IEEE, 2006.

[82] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography, SAC'99*, volume 1758 of *LNCS*, pages 184–199. Springer-Verlag, 2000.

[83] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[84] M. Naor and M. Yung. Universal one-way hash functions and their crypographic applications. In *Proceedings of the* 21*st annual ACM Symposium on the Theory of Computing*, pages 33–43, 1989.

[85] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the* 22*nd annual ACM Symposium on the Theory of Computing*, pages 427–437, 1990.

[86] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In S. Halevi and T. Rabin, editors, *Theory of Cryptography Conference, TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer-Verlag, 2006. Full version avaialble on ePrint archive.

[87] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, 1998.

[88] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

[89] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 563–572. IEEE Computer Society, 2005.

[90] R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005)*, pages 533–542. ACM, 2005.

[91] T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *LNCS*, pages 522–526. Springer-Verlag, 1991.

[92] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer-Verlag, 1992.

[93] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, 1992.

[94] M. Rückert and D. Schröder. Security of verifiably encrypted signatures and a construction without random oracles. IACR ePrint Archive 2009/027, 2009.

[95] A. Sahai. Non-malleable non-interactive zero-knowledge and chosen-ciphertext security. In *Proceedings of the* 40*th IEEE Annual Symposium on Foundations of Computer Science*, pages 543–553, 1999.

[96] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer-Verlag, 2001.

[97] A. D. Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *FOCS*, pages 427–436, 1992.

[98] M. Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.

[99] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007.

[100] H. Shacham and B. Waters. Efficient ring signatures without random oracles. In *Public Key Cryptography*, pages 166–180, 2007.

[101] E. R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17(4):277–296, 2004.