Tightly Secure Signatures and Public-Key Encryption

Dennis Hofheinz[1] and Tibor Jager[2]

[1]Karlsruhe Institute of Technology, Germany, dennis.hofheinz@kit.edu
[2]Ruhr-University Bochum, Germany, tibor.jager@rub.de

April 15, 2013

Abstract

We construct the first public-key encryption scheme whose chosen-ciphertext (i.e., IND-CCA) security can be proved under a standard assumption *and* does not degrade in either the number of users or the number of ciphertexts. In particular, our scheme can be safely deployed in unknown settings in which no a-priori bound on the number of encryptions and/or users is known.

As a central technical building block, we devise the first structure-preserving signature scheme with a tight security reduction. (This signature scheme may be of independent interest.) Combining this scheme with Groth-Sahai proofs yields a tightly simulation-sound non-interactive zero-knowledge proof system for group equations. If we use this proof system in the Naor-Yung double encryption scheme, we obtain a tightly IND-CCA secure public-key encryption scheme from the Decision Linear assumption.

We point out that our techniques are not specific to public-key encryption security. Rather, we view our signature scheme and proof system as general building blocks that can help to achieve a tight security reduction.

Keywords: Tight security proofs, structure-preserving signatures, public-key encryption, Groth-Sahai proofs.

Contents

1 Introduction

Security reductions. Many interesting cryptographic primitives (such as public-key encryption and signature schemes) cannot be proven secure with current techniques, as their security would imply $P \neq NP$. Instead, we usually provide a proof of security under a suitable (computational) assumption (such as the hardness of factoring large integers). Concretely, a *security reduction* shows that any successful adversary $\mathcal{A}$ on the scheme's security can be converted into a successful solver $\mathcal{B}$ of the underlying computational problem. Naturally, we would desire that $\mathcal{B}$'s success $\epsilon_{\mathcal{B}}$ is at least as large as $\mathcal{A}$'s success $\epsilon_{\mathcal{A}}$ in attacking the system. However, security reductions often suffer from a nontrivial multiplicative *security loss* L (such that only $\epsilon_{\mathcal{A}} \leq L \cdot \epsilon_{\mathcal{B}}$ can be guaranteed).

Cryptography in a multi-user setting. The issue of a nontrivial security loss becomes particularly problematic, e.g., in the case of a realistic public-key encryption (PKE) scenario with many users who encrypt and send many ciphertexts. Standard security notions for PKE schemes (such as IND-CCA security [50, 22]) only consider one user and one ciphertext. In particular, with very few exceptions ([10, 36]), most security proofs of encryption schemes only prove security in this simplified scenario. This can be justified with general results ([9, 10]) that show that one-user, one-ciphertext PKE security implies security in the much more realistic multi-user, multi-ciphertext case. However, this generic reduction suffers from a reduction loss of $L = n_U \cdot n_C$, where n_U is the number of users, and n_C is the number of ciphertexts per user.

That is, even if a PKE scheme reaches a certain level of security in the commonly considered one-user, one-ciphertext setting, its security level may be significantly lower in a realistic setting. (In fact, Bellare et al. [8] give a concrete example of such a scheme in the symmetric-key setting.) This is particularly problematic, since it may not be clear at deployment time for how many users and encryptions a PKE scheme will be used. We thus note that the analysis of cryptographic primitives in the multi-user setting is necessary to derive concrete security guarantees for realistic settings.

The difficulty in constructing tightly secure schemes. Let us say that a security reduction (in the multi-user setting) is *tight* if the corresponding reduction loss L is a (preferably small) constant. In particular, the security of a tightly secure scheme does not deteriorate in the number of users (or encryptions). For some security notions and constructions, tightly secure reductions can be constructed relatively painlessly. For instance, the random self-reducibility of the Decisional Diffie-Hellman problem allows to tightly prove the IND-CPA security of ElGamal encryption [23] even with many users and ciphertexts [10]. However, for other security notions, it seems inherently difficult to derive a tight security reduction.

For instance, there is no known PKE scheme with a tight (IND-CCA) security reduction from a standard assumption.[1] Diving into the technical details for a moment, one reason for this apparent difficulty is that an IND-CCA security reduction must be able to decrypt all of $\mathcal{A}$'s decryption queries, but must not be able to decrypt its own IND-CCA challenge. One way to resolve this dilemma is to partition the set of ciphertexts into those that can be decrypted, and those that

[1] Bellare, Boldyreva, and Micali [10] show that the security loss of Cramer-Shoup encryption [19] does not depend on the number of users; however, their reduction loss still grows linearly in the number of ciphertexts per user. The identity-based encryption schemes [28, 29] enjoy a tight reduction, and can be generically converted into tightly IND-CCA secure PKE schemes [15]; however, they rely on a non-standard multi-challenge assumption. A similar argument holds for the tightly IND-SO-CCA secure PKE scheme of Hofheinz [36].

cannot. (For instance, one can set up the proof simulation for $\mathcal{A}$ such that the reduction can decrypt all ciphertexts except for one single challenge ciphertext; examples of this approach are [12, 38, 39].) This proof technique can only argue about a small number of ciphertexts at a time, and a hybrid argument is required to show security in the multi-ciphertexts case. Such a hybrid argument results again in a reduction loss that is linear in the number of ciphertexts.

Another way to show IND-CCA security is to argue with the information the adversary has about the secret key. (Examples of this approach are [19, 20, 43].) Since the size of the secret key is limited, its entropy can only be used to argue about the security of a limited number of ciphertexts at a time. Again a hybrid argument (entailing a linear reduction loss) is required to argue about the security of many ciphertexts. One could hope that the described inherent hybrid arguments of partitioning and entropy-based strategies to show IND-CCA security can be circumvented using dual system (identity-based) encryption techniques [54, 45]. In a nutshell, dual system encryption provides a way to subtly and gradually randomize the distribution of challenge ciphertexts (and user secret keys in an IBE scheme) without explicitly partitioning the set of ciphertexts into decryptable and non-decryptable ones. However, while dual system techniques rely on re-randomizable computational problems (such as the Decision Linear problem), and thus in principle should not suffer from the described problems, all known dual systems schemes still have to use a hybrid argument and do not achieve a tight security reduction.

Note that one can construct IND-CCA-secure public-key encryption schemes with tight reduction in the random oracle model [5], for instance by applying the Fujisaki-Okamoto transform [26] to the tightly IND-CPA-secure schemes from [10].

Our contribution. In this paper, we present a general technique to construct tightly secure cryptographic primitives. As an example, we construct the first PKE scheme that is tightly IND-CCA secure under a simple assumption. Concretely, all the constructions in this paper build on the Decision Linear (DLIN) assumption.[2]

Our main technical building block is a *structure-preserving* signature scheme with a tight security reduction.[3] Loosely speaking, a structure-preserving signature scheme is one in which verification can be expressed as a sequence of group equations. In particular, structure-preserving schemes are amenable to Groth-Sahai (GS) proofs [35], which are efficient non-interactive proof systems for sets of equations over a group. Following a known paradigm [46, 34, 17], we then turn our signature scheme into a *simulation-sound* non-interactive zero-knowledge proof system for group equations.[4] Since our signature is tightly secure, so is the proof system.

This tightly secure and simulation-sound proof system offers the technical means to achieve tight security. We exemplify this by implementing the Naor-Yung paradigm [48, 46, 14] with our proof system to obtain a tightly IND-CCA secure PKE scheme.

[2]However, we expect that our constructions also naturally generalize to the — potentially weaker — K-Linear assumption and to suitable subgroup decision assumptions.

[3]We construct tightly secure structure-preserving signatures. (In fact, our schemes can sign their own public key; such signature schemes are commonly also referred to as automorphic.) While there exist tightly secure signature schemes (e.g., [27, 13, 18, 11, 40, 52]), and structure-preserving signature schemes (e.g., [25, 3, 17]), our scheme seems to be the first to achieve both properties. This combination of properties is crucial for our applications.

[4]By a simulation-sound zero-knowledge proof system, we mean one in which it is infeasible to generate valid proofs for false statements, even when already having observed *many* simulated proofs for possibly false statements.

Some technical details. Our signature scheme is tree-based and inspired by the scheme of Boneh, Mironov, and Shoup [13]. However, their scheme is not structure-preserving, as it uses the hash of group elements as an exponent. To avoid this kind of "domain translation," we construct a one-time signature scheme in which signatures and messages are vectors of group elements. (Since we want to implement a tree-based many-time signature scheme, we require, however, that messages can be longer than public verification keys.) To describe our (one-time) scheme, assume groups $\mathbb{G}, \mathbb{G}_T$ with pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Write $E : \mathbb{G}^3 \times \mathbb{G} \to \mathbb{G}_T^3$ for component-wise pairing. (That is, $E((u_1, u_2, u_3), v) = (e(u_1, v), e(u_2, v), e(u_3, v)$.) In a nutshell, a signature for a message $m = (m_i)_{i=1}^n \in \mathbb{G}^n$ is of the form $(s, t) \in \mathbb{G}^2$ and satisfies

$$\left(\prod_{i=1}^{n} E(U_i, m_i)\right) \cdot E(G, s) \cdot E(H, t) = E(X, z), \tag{1}$$

where the $G, H, X, U_1, \ldots, U_n \in \mathbb{G}^3$, and $z \in \mathbb{G}$ are part of the public verification key.[5] This means that m_i, s, t, z act as coefficients in a linear equation (in the $\mathbb{G}$-exponent) for the vectors U_i, G, H, X. Now if all the vectors U_i, G, H, X are DLIN-tuples of the form (g^x, h^y, k^{x+y}) (for fixed g, h, k), then any message can be signed when knowing all U_i, G, H, X-exponents. Now consider a setup in which one U_i (say, U_j) and X are non-DLIN-tuples, and the other U_i and G, H are DLIN-tuples. Then, only messages with a specific m_j-component can be signed. (This is easiest seen by thinking of DLIN-elements as linearly dependent vectors in the exponent; with this view, X and U_j are the only vectors outside the vector space generated by DLIN-tuples. We note that this idea of "unique signatures" already appears in the ROM-based signature scheme of Katz and Wang [41].) In the proof of (non-adaptive) one-time security, this m_j will be set up as the message signed for the adversary $\mathcal{A}$. Furthermore, if the adversary forges a message, we know that this message must have m_j in its j-th component. Since a forged signature must refer to a message M^* that is different from the message M signed for $\mathcal{A}$, there must be an j with $m_j^* \neq m_j$. A small hybrid argument over $j \in [n]$ thus shows security. (We stress that we employ a hybrid argument only over a small set $[n]$ that will not depend on the number of users or ciphertexts. Specifically, our scheme can be implemented with $n = 8$.) From this one-time secure scheme, we will construct a tree-based many-time secure scheme following ideas from [13]; in particular, we will re-use the U_i for many instances of the one-time scheme. (Such a re-use of public key parts has also been used and made explicit [7].) This will finally yield an adaptively secure structure-preserving signature scheme with a tight security reduction. The remaining steps that lead to a tightly simulation-sound proof system and tightly IND-CCA secure public-key encryption follow existing ideas [34, 14], so we will not outline them here. (Details follow inside.)

Further applications. We note that plugging our tightly simulation-sound proof system into the construction of [14] yields a PKE scheme that is tightly chosen-ciphertext secure even under *key-dependent* message attacks. Similarly, we expect that proofs of chosen-ciphertext security for *identity-based* encryption schemes can be made independent of the number of challenge ciphertexts. (However, here we do not expect to obtain independence from the number of users, i.e., identities.) Besides, structure-preserving signatures have found applications in several areas (e.g., [16, 25, 33]).

[5] We highlight that (1) actually consists of three pairing product equations. This can in part be justified by [3, Theorem 2], which states that already any secure structure-preserving two-time signature scheme must have at least two verification equations.

We expect that *tightly secure* structure-preserving signature schemes lead to tighter security proofs in these applications.

Differences to conference version. This paper constitutes the full version of [37]. Apart from full proofs and additional explanations, like the construction of a suitable IND-CPA secure encryption scheme which is suitable for our CCA-secure construction, this full version contains a description of an *adaptively* secure (EUF-CMA) structure-preserving signature scheme. See Section 3.4 for details.

Note on further related work. In [2, Appendix C] (the full version of [1]), Abe et al. describe a DLIN-based structure-preserving one-time signature scheme that is more efficient than ours and has subsequently also been proven compatible with our tree-based approach [4]. In particular, together with our work, their scheme yields a more efficient tightly IND-CCA-secure encryption scheme. (We were not aware of their one-time signature scheme when designing ours.)

2 Preliminaries

Notation. If A is a set, then $a \xleftarrow{\$} A$ denotes that a is distributed uniformly over A. If A is a probabilistic algorithm, then $a \xleftarrow{\$} A$ denotes that a is computed by A using fresh random coins. For $n \in \mathbb{N}$ we write $[n]$ to denote the set $[n] = \{1, \ldots, n\}$. For $j \in [n]$ we write $[n \setminus j]$ to denote the set $\{1, \ldots, n\} \setminus \{j\}$.

2.1 Digital Signatures

Syntax. Generally, we assume a parameter generation algorithm $\mathsf{Sig.Param}$ which takes as input the security parameter κ and generates public parameters $\Pi \xleftarrow{\$} \mathsf{Sig.Param}(\kappa)$. A digital signature scheme $\mathsf{Sig} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Vfy})$ consists of three algorithms. Key generation algorithm $\mathsf{Sig.Gen}$ generates, on input parameters Π, a keypair $(vk, sk) \xleftarrow{\$} \mathsf{Sig.Gen}(\Pi)$ consisting of a secret signing key sk and a public verification key vk. The signing algorithm $\mathsf{Sig.Sign}$ inputs a message and the secret signing key, and returns a signature $\sigma \xleftarrow{\$} \mathsf{Sig.Sign}(sk, m)$ of the message. The verification algorithm $\mathsf{Sig.Vfy}$ takes a verification key and a message with corresponding signature as input, and returns $b \leftarrow \mathsf{Sig.Vfy}(vk, m, \sigma)$ where $b \in \{0, 1\}$. We require the usual correctness properties.

Security. Let us recall the *existential unforgeability against chosen message attacks* (EUF-CMA) security experiment [32], played between a challenger and a forger $\mathcal{A}$.

1. The forger, on input public parameters Π, may ask a *non-adaptive* chosen-message query. To this end, it submits a list of messages $M^{(1)}, \ldots, M^{(q_0)}$ to the challenger.

2. The challenger runs $\mathsf{Sig.Gen}(\Pi)$ to generate a keypair (vk, sk). The forger receives vk and a signature $\sigma^{(i)}$ for each chosen message $M^{(i)}$, $i \in [q_0]$.

3. Now the forger may ask *adaptive* chosen-message queries. Each query consists of a message $M^{(i)}$, $i \in [q_0 + 1, q]$, and is answered by the challenger with a signature $\sigma^{(i)}$ under sk for message $M^{(i)}$.

4. Finally the forger outputs a message M^* and signature σ^*.

Definition 1. An adversary is *adaptive*, if it asks at least one adaptive chosen-message query. Otherwise it is *non-adaptive*. Let $\mathcal{A}$ be an adversary (adaptive or non-adaptive) that runs in time t, makes q chosen-message queries (in total), and outputs (M^*, σ^*). We say that $\mathcal{A}$ (ϵ, t, q)-breaks the EUF-CMA security of Sig if

$$\Pr[\mathsf{Sig.Vfy}(vk, M^*, \sigma^*) = 1 \wedge M^* \notin \{M^{(1)}, \ldots, M^{(q)}\}] \geq \epsilon.$$

We say that $\mathcal{A}$ (ϵ, t, q)-breaks the *strong* EUF-CMA security of Sig if

$$\Pr[\mathsf{Sig.Vfy}(vk, M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \notin \{(M^{(1)}, \sigma^{(1)}), \ldots, (M^{(q)}, \sigma^{(q)})\}] \geq \epsilon.$$

Accordingly, a signature scheme Sig is (ϵ, t, q)-*secure* against adaptive (non-adaptive) EUF-CMA attacks, if there exists no adaptive (non-adaptive) adversary that (ϵ, t, q)-breaks Sig.

2.2 Complexity Assumptions

In the following, let $\mathbb{G}$ be a group of prime order p. For a generator $g \in \mathbb{G}$ and arbitrary $h \in \mathbb{G}$, let $\log_g(h) \in \mathbb{Z}_p$ be the discrete logarithm of h (to base g), such that $g^{\log_g(h)} = h$.

Definition 2. Let $g, h \in \mathbb{G}$ be random generators of $\mathbb{G}$. We say that an adversary $\mathcal{A}$ (ϵ, t)-breaks the *Discrete Logarithm* (DLOG) assumption in $\mathbb{G}$, if $\mathcal{A}$ runs in time t and

$$\Pr[\mathcal{A}(g, h) = \log_g(h)] \geq \epsilon.$$

Furthermore, for generators $g, h, k \in \mathbb{G}$ let $\mathsf{DLIN}(g, h, k)$ denote the set

$$\mathsf{DLIN}(g, h, k) = \{(g^u, h^v, k^{u+v}) : u, v \in \mathbb{Z}_p\}$$

Let $G = (g, 1, k) \in \mathbb{G}^3$ and $H = (1, h, k) \in \mathbb{G}^3$. For two vectors $V = (v_1, v_2, v_3)$ and $W = (w_1, w_2, w_3)$ in $\mathbb{G}^3$ and $u \in \mathbb{Z}_p$ let $V \cdot W := (v_1 \cdot w_1, v_2 \cdot w_2, v_3 \cdot w_3)$ and let $V^u := (v_1^u, v_2^u, v_3^u)$. Then we can write the set $\mathsf{DLIN}(g, h, k)$ equivalently as

$$\mathsf{DLIN}(g, h, k) = \{U : U = G^u \cdot H^v, u, v \in \mathbb{Z}_p\}.$$

Definition 3. Let $g, h, k \xleftarrow{\$} \mathbb{G}$ be random generators of $\mathbb{G}$, and let $U \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$ and $V \xleftarrow{\$} \mathbb{G}^3$. We say that an adversary $\mathcal{B}$ (ϵ, t)-breaks the *Decision Linear* (DLIN) assumption in $\mathbb{G}$, if $\mathcal{B}$ runs in time t and

$$\Pr[\mathcal{B}(g, h, k, U) = 1] - \Pr[\mathcal{B}(g, h, k, V) = 1] \geq \epsilon.$$

3 Structure-Preserving Signatures

In the sequel let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and let g, h, k be random generators of $\mathbb{G}$. For a vector $V = (v_0, v_1, v_2) \in \mathbb{G}^3$ and a group element $w \in \mathbb{G}$, we write $E(V, w)$ to denote the vector

$$E(V, w) = (e(v_0, w), e(v_1, w), e(v_2, w)).$$

For two vectors $V = (v_0, v_1, v_2)$ and $W = (w_0, w_1, w_2)$ we denote with $V \cdot W$ the component-wise product

$$V \cdot W = (v_0 \cdot w_0, v_1 \cdot w_1, v_2 \cdot w_2).$$

For $w \in \mathbb{Z}_p$ we write V^w to denote $V^w = (v_0^w, v_1^w, v_2^w)$.

3.1 Structure-Preserving One-Time Signatures for Single Group Elements

Let $\mathsf{OTSig} = (\mathsf{OTSig.Gen}, \mathsf{OTSig.Sign}, \mathsf{OTSig.Vfy})$ be the following signature scheme.

$\mathsf{OTSig.Gen}(g, h, k)$**:** Given random generators $g, h, k \in \mathbb{G}$, choose a random generator $z \xleftarrow{\$} \mathbb{G}$ and integers $u, v, x, y \xleftarrow{\$} \mathbb{Z}_p$. Set $U := (g^u, h^v, k^{u+v})$ and $X := (g^x, h^y, k^{x+y})$. Set $vk := (g, h, k, U, X, z)$ and $sk := (u, v, x, y)$ and return (vk, sk).

$\mathsf{OTSig.Sign}(sk, m)$**:** Given a message $m \in \mathbb{G}$ and a secret key $sk = (u, v, x, y)$, compute $s := z^x m^{-u}$ and $t := z^y m^{-v}$ and return $\sigma = (s, t)$.

$\mathsf{OTSig.Vfy}(vk, m, \sigma)$**:** Given a public key $vk = (g, h, k, U, X, z)$, message m, and signature $\sigma = (s, t)$, let $G := (g, 1, k)$ and $H = (1, h, k)$. Return 1 if equation

$$E(U, m) \cdot E(G, s) \cdot E(H, t) = E(X, z)$$

holds. Otherwise return 0.

It is a straightforward calculation to verify the correctness of this scheme. Before we prove security, let us state a technical lemma which will simplify the proof.

Lemma 1. *Let $g, h, k \in \mathbb{G}$ be generators, $G := (g, 1, k)$ and $H := (1, h, k)$. There exists an algorithm $\mathcal{T}$ which takes as input $G, H, U \in \mathbb{G}^3$ and $m \in \mathbb{G}$, and outputs $X \in \mathbb{G}^3$ and $z, s, t \in \mathbb{G}$ such that*

(i) If $U \in \mathsf{DLIN}(g, h, k)$, then X is distributed uniformly over $\mathsf{DLIN}(g, h, k)$ and z is uniform over $\mathbb{G}$.

(ii) (s, t) satisfy the equation

$$E(U, m) \cdot E(G, s) \cdot E(H, t) = E(X, z). \tag{2}$$

(iii) If $U \notin \mathsf{DLIN}(g, h, k)$ then (2) has a unique *solution (m, s, t).*

(iv) If $U \in \mathsf{DLIN}(g, h, k)$ then for each m there exist unique *(s, t) such that (m, s, t) satisfies (2).*

The running time of $\mathcal{T}$ is dominated by 7 exponentiations in $\mathbb{G}$.

The proof is given at the end of this section.

Theorem 1. *Suppose there exists a non-adaptive adversary $\mathcal{A}$ that $(\epsilon, t, 1)$-breaks the EUF-CMA security of OTSig. Then there exists an adversary $\mathcal{B}$ that (ϵ', t')-breaks the DLIN assumption in $\mathbb{G}$, where t' is roughly the runtime of the EUF-CMA experiment with $\mathcal{A}$, and $\epsilon' \geq \epsilon - 1/p$.*

Proof. We construct adversary $\mathcal{B}$ as follows. $\mathcal{B}$ receives as input a DLIN challenge (g, h, k, U), where either $U \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$ or $U \xleftarrow{\$} \mathbb{G}^3$. It starts $\mathcal{A}$, and receives a chosen-message $m \in \mathbb{G}$. Then $\mathcal{B}$ runs algorithm $\mathcal{T}$ from Lemma 1 to obtain $(X, z, s, t) \xleftarrow{\$} \mathcal{T}(G, H, U, m)$. It defines $vk := (g, h, k, U, X, z)$ and returns vk and signature (s, t) to $\mathcal{A}$.

If $\mathcal{A}$ outputs $(m', (s', t'))$ such that

$$E(U, m') \cdot E(G, s') \cdot E(H, t') = E(X, z)$$

is satisfied and $m' \neq m$, then $\mathcal{B}$ outputs 1. Otherwise it outputs 0.

Analysis. If $U \in \mathsf{DLIN}(g,h,k)$, then by Property (i) of Lemma 1 vk is a correctly distributed public key. By Property (ii) (s,t) is a valid signature for m under vk, and by Property (iv) (s,t) is unique, thus correctly distributed. Therefore in this case $\mathcal{A}$ outputs a forgery (m',σ') with probability ϵ by assumption. Thus, if $U \xleftarrow{\$} \mathsf{DLIN}(g,h,k)$ then $\mathcal{B}$ outputs 1 with probability ϵ.

If $U \xleftarrow{\$} \mathbb{G}^3$ then we have $U \notin \mathsf{DLIN}(g,h,k)$ with probability $1-1/p$. Moreover, if indeed $U \notin \mathsf{DLIN}(g,h,k)$ then by Property (iii) of Lemma 1 the solution (m,s,t) is unique, in which case $\mathcal{B}$ outputs 1 with probability 0. Thus, if $U \xleftarrow{\$} \mathbb{G}^3$ then $\mathcal{B}$ outputs 1 with probability at most $1/p$. □

Proof of Lemma 1. Algorithm $\mathcal{T}$ chooses random integers $\hat{z},\hat{s},\hat{t} \xleftarrow{\$} \mathbb{Z}_p$, and proceeds as follows.

- If $m \neq 1$, then $\mathcal{T}$ sets $z = m^{\hat{z}}$, $s := m^{\hat{s}}$, $t := m^{\hat{t}}$, and $X := (U \cdot G^{\hat{s}} \cdot H^{\hat{t}})^{1/\hat{z}}$.
- If $m = 1$, then $\mathcal{T}$ sets $z = g^{\hat{z}}$, $s := g^{\hat{s}}$, $t := g^{\hat{t}}$, and $X := (G^{\hat{s}} \cdot H^{\hat{t}})^{1/\hat{z}}$.

Note that in each case z is distributed uniformly over $\mathbb{G}$. If $m \neq 1$, then X is uniform over $\mathsf{DLIN}(g,h,k)$ if $U \in \mathsf{DLIN}(g,h,k)$. If $m = 1$, then X is uniform over $\mathsf{DLIN}(g,h,k)$, regardless of U. This yields (i). A straightforward calculation shows that in each case the values (s,t) are defined such that the equation in (ii) is satisfied. Computing

- $m^{\hat{z}}$, $m^{\hat{s}}$, $m^{\hat{t}}$, $G^{\hat{s}} = (g^{\hat{s}},1,k^{\hat{s}})$, and $H^{\hat{t}} = (1,h^{\hat{t}},k^{\hat{t}})$ (in case $m \neq 1$), or
- $g^{\hat{z}}$, $g^{\hat{s}}$, $g^{\hat{t}}$, $G^{\hat{s}} = (g^{\hat{s}},1,k^{\hat{s}})$, and $H^{\hat{t}} = (1,h^{\hat{t}},k^{\hat{t}})$ (in case $m = 1$)

takes seven exponentiations in $\mathbb{G}$.

In order to prove (iii), let us write $U = (g^u,h^v,k^w)$ and $X = (x_0,x_1,x_2)$. Then we can view Equation 2 as a system of equations

$$\begin{aligned} e(g^u,m)\cdot e(g,s)\cdot e(1,t) &= e(x_0,z)\\ e(h^v,m)\cdot e(1,s)\cdot e(h,t) &= e(x_1,z)\\ e(k^w,m)\cdot e(k,s)\cdot e(k,t) &= e(x_2,z) \end{aligned}$$

Taking discrete logarithms to base g, the above system of equations is equivalent to

$$\begin{pmatrix} u & 1 & 0\\ v\log h & 0 & \log h\\ w\log k & \log k & \log k \end{pmatrix}\begin{pmatrix}\log m\\ \log s\\ \log t\end{pmatrix} = \begin{pmatrix}\log x_0 \log z\\ \log x_1 \log z\\ \log x_2 \log z\end{pmatrix}$$

The determinant of the 3×3 matrix is equal to

$$w\log h\log k - u\log h\log k - v\log h\log k,$$

which equals 0 if and only if $w = u+v$ (note that h and k are generators, therefore we have $\log h \neq 0 \neq \log k$). Since $U \notin \mathsf{DLIN}(g,h,k)$ implies $w \neq u+v$, the solution (m,s,t) to the system of equations is unique, which proves (iii). Finally, if we fix $\log m$ then (iv) follows from the fact that $\log h \neq 0$.

3.2 Structure-Preserving One-Time Signatures for Vectors of Group Elements

In this section we extend the message space of OTSig from the previous section to vectors $M = (m_1, \ldots, m_n)$ of n elements of $\mathbb{G}$. The scheme is very similar, except that now the public key and the verification equation contain n elements $U_1, \ldots, U_n$ instead of a single element U.

Scheme $\mathsf{OTSig}^n = (\mathsf{OTSig.Gen}^n, \mathsf{OTSig.Sign}^n, \mathsf{OTSig.Vfy}^n)$ works as follows.

$\mathsf{OTSig.Gen}^n(g, h, k)$**:** Given a generators $g, h, k \in \mathbb{G}$, choose a random generator $z \xleftarrow{\$} \mathbb{G}$ and $2(n+1)$ integers $u_1, v_1, \ldots, u_n, v_n, x, y \xleftarrow{\$} \mathbb{Z}_p$. Set $U_i := (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for $i \in [n]$ and $X := (g^x, h^y, k^{x+y})$. Set $vk := (g, h, k, U_1, \ldots, U_n, X, z)$ and $sk := (u_1, v_1, \ldots, u_n, v_n, x, y)$ and return (vk, sk).

$\mathsf{OTSig.Sign}^n(sk, M)$**:** Given a message vector $M = (m_1, \ldots, m_n) \in \mathbb{G}^n$ and secret key sk, compute $s := z^x \prod_{i=1}^n m_i^{-u_i}$ and $t := z^y \prod_{i=1}^n m_i^{-v_i}$ and return $\sigma = (s, t)$.

$\mathsf{OTSig.Vfy}^n(vk, M, \sigma)$**:** Given a public key vk, message vector $M = (m_1, \ldots, m_n)$, and signature $\sigma = (s, t)$, let $G := (g, 1, k)$ and $H = (1, h, k)$. Return 1 if equation

$$\prod_{i=1}^{n} E(U_i, m_i) \cdot E(G, s) \cdot E(H, t) = E(X, z)$$

holds. Otherwise return 0.

Again it is a simple calculation to verify the correctness of this scheme.

The following lemma generalizes Lemma 1, and will be useful to prove the above theorem as well as for the security proof of our tree-based signature scheme in the following section.

Lemma 2. *Let $g, h, k \in \mathbb{G}$ be generators, $G := (g, 1, k)$ and $H := (1, h, k)$. There exists an algorithm $\mathcal{T}_n$ which takes as input an index $j \in [n]$, $G, H, U_j \in \mathbb{G}^3$, $(u_i, v_i) \in \mathbb{Z}_p$ for $i \in [n \setminus j]$ and $M = (m_1, \ldots, m_n) \in \mathbb{G}^n$, and outputs $X \in \mathbb{G}^3$ and $z, s, t \in \mathbb{G}$ such that*

(i) If $U_j \in \mathsf{DLIN}(g, h, k)$, then X is distributed uniformly over $\mathsf{DLIN}(g, h, k)$ and z is uniform over $\mathbb{G}$.

(ii) (s, t) satisfy the equation

$$\prod_{i=1}^{n} E(U_i, m_i) \cdot E(G, s) \cdot E(H, t) = E(X, z). \tag{3}$$

where $U_i := (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for $i \in [n \setminus j]$.

(iii) If $U_j \notin \mathsf{DLIN}(g, h, k)$ then for each $((m_1^*, \ldots, m_n^*), s^*, t^*)$ satisfying (3) we have $m_j^* = m_j$.

(iv) If $U_j \in \mathsf{DLIN}(g, h, k)$ then for each $M = (m_1, \ldots, m_n) \in \mathbb{G}^n$ there exist unique *(s, t) such that (M, s, t) satisfy (3).*

The running time of $\mathcal{T}_n$ is dominated by one execution of Algorithm $\mathcal{T}$ from Lemma 1 plus $2(n-1)$ exponentiations in $\mathbb{G}$.

The proof is a simple extension of the proof of Lemma 1, and deferred to the end of this section.

Theorem 2. *Suppose there exists a non-adaptive adversary $\mathcal{A}$ that $(\epsilon, t, 1)$-breaks the EUF-CMA security of OTSig^n. Then there exists an adversary $\mathcal{B}$ that (ϵ', t')-breaks the DLIN assumption in $\mathbb{G}$, where t' is roughly with runtime of the EUF-CMA experiment with $\mathcal{A}$, and $\epsilon' \geq \epsilon/n - 1/p$.*

Proof. We construct adversary $\mathcal{B}$ as follows. $\mathcal{B}$ receives as input a DLIN challenge (g, h, k, U), where either $U \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$ or $U \xleftarrow{\$} \mathbb{G}^3$. It picks a random index $j \in [n]$, starts $\mathcal{A}$, and receives a chosen-message vecor $M = (m_1, \ldots, m_n) \in \mathbb{G}$. Then it chooses $u_i, v_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $U_i = (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for each $i \in [n \setminus j]$, and runs algorithm $\mathcal{T}_n$ from Lemma 2 on input $(j, G, H, U_j, (u_i, v_i)_{i\in[n]\setminus\{j\}}, M)$ to obtain $(X, z, s, t) \xleftarrow{\$} \mathcal{T}_n$. Then it sets $vk = (g, h, k, U_1, \ldots, U_n, X, z)$ and returns vk and signature $\sigma = (s, t)$ for M to $\mathcal{A}$.

If $\mathcal{A}$ outputs (M^*, s^*, t^*) , $M^* = (m_1^*, \ldots, m_n^*)$, such that $m_j^* \neq m_j$ and equation

$$\prod_{i=1}^{n} E(U_i, m_i^*) \cdot E(G, s^*) \cdot E(H, t^*) = E(X, z).$$

is satisfied, then $\mathcal{B}$ outputs 1. Otherwise it outputs 0.

Analysis. If $U \in \mathsf{DLIN}(g, h, k)$, then by Property (i) of Lemma 2 vk is a correctly distributed public key. By Property (ii) (s, t) is a valid signature for m under vk, and by Property (iv) (s, t) is unique, thus correctly distributed. Therefore in this case $\mathcal{A}$ outputs a forgery (M^*, s^*, t^*) with probability ϵ by assumption. Since $M \neq M^*$ we have $m_j \neq m_j^*$ with probability at least $1/n$. Thus, if $U \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$ then $\mathcal{B}$ outputs 1 with probability ϵ/n.

If $U \xleftarrow{\$} \mathbb{G}^3$ then we have $U \in \mathsf{DLIN}(g, h, k)$ with probability $1/p$. Moreover, if $U \notin \mathsf{DLIN}(g, h, k)$ then by Property (iii) of Lemma 2 there exists no M^* that satisfies the verification equation with $m_j^* \neq m_j$. Thus, if $U \xleftarrow{\$} \mathbb{G}^3$ then $\mathcal{B}$ outputs 1 with probability $1/p$.

□

Proof of Lemma 2. Algorithm $\mathcal{T}_n$ runs Algorithm $\mathcal{T}$ from Lemma 1 to compute $(X, z, s_j, t_j) \xleftarrow{\$} \mathcal{T}(G, H, U, U_j, m_j)$, which yields a solution to the equation

$$E(U_j, m_j) \cdot E(G, s_j) \cdot E(H, t_j) = E(X, z).$$

Clearly Property (i) of $\mathcal{T}_n$ follows from Property (i) of $\mathcal{T}$.

Then $\mathcal{T}_n$ sets $s := s_j \cdot \prod_{i\in[n\setminus j]} m_i^{-u_i}$ and $t := t_j \cdot \prod_{i\in[n\setminus j]} m_i^{-v_i}$. Note that

$$\begin{aligned}\prod_{i=1}^{n} E(U_i, m_i) \cdot E(G, s) \cdot E(H, t) &= \prod_{i=1}^{n} E(U_i, m_i) \cdot E(G, s_j) \cdot E(H, t_j) \cdot \prod_{i\in[n\setminus j]} E(U_i, m_i)^{-1} \\ &= E(U_j, m_j) \cdot E(G, s_j) \cdot E(H, t_j) = E(X, z),\end{aligned}$$

which proves Property (ii) of $\mathcal{T}_n$.

To prove Property (iii), suppose (for contradiction) that there exists $((m_1^*, \ldots, m_n^*), s^*, t^*)$ with $m_j^* \neq m_j$ that satisfies Equation (3). Let

$$s_j^* := s^* \cdot \prod_{i\in[n\setminus j]} m_i^{u_i} \quad \text{and} \quad t_j^* := t^* \cdot \prod_{i\in[n\setminus j]} m_i^{v_i}.$$

Then, as above, we have

$$\prod_{i=1}^{n} E(U_i, m_i^*) \cdot E(G, s^*) \cdot E(H, t^*) = \prod_{i=1}^{n} E(U_i, m_i^*) \cdot E(G, s_j^*) \cdot E(H, t_j^*) \cdot \prod_{i \in [n \setminus j]} E(U_i, m_i')^{-1}$$
$$= E(U_j, m_j^*) \cdot E(G, s_j^*) \cdot E(H, t_j^*) = E(X, z).$$

By Property (iii) of Algorithm $\mathcal{T}$ the latter is impossible, unless $m_j^* = m_j$, contradiction.

Property (iv) of Algorithm $\mathcal{T}_n$ follows from Property (iv) of Algorithm $\mathcal{T}$, and the observation that each U_i, $i \in [n \setminus j]$, can be written *uniquely* as $U_i = G^{u_i} \cdot H^{v_i}$.

3.3 Structure-Preserving Signatures Secure Against Non-adaptive Adversaries

Now we construct a signature scheme based on a binary tree of depth d. The scheme has message space $\mathbb{G}^8$, allows us to issue up to 2^d signatures (where d may be large enough such that 2^d is virtually unbounded, e.g. $d = 80$), and is provably secure against non-adaptive adversaries under the DLIN assumption.

Basic Idea. The construction is based on binary Merkle trees [47], instantiated such that all nodes except for the root can be generated "on the fly." In particular, not the complete tree must be stored (which would clearly be infeasible for large d). Each node of the tree consists of a key pair (vk, sk) of our one-time signature scheme from Section 3.2. The two children of this node are authenticated by a signature over their respective public keys that verifies under vk. The key-pairs corresponding to tree leaves are used to sign actual messages.

Recall that a public key consists of a vector $(g, h, k, U_1, \ldots, U_n, X, z)$, where n is the number of group elements to be signed. In order to obtain a tight security reduction, we re-use the public-key components $(g, h, k, U_1, \ldots, U_n)$ for all nodes of the tree. Only the (X, z)-components are unique for each node. The tight reduction is inspired by the proof of the tree-based signature scheme of Boneh et al. [13]. Let us give some more details on an informal level.

- The tree is parametrized by $(g, h, k) \in \mathbb{G}^3$ and $U_1, \ldots, U_8 \in \mathsf{DLIN}(g, h, k)$, where for each $i \in [8]$ we have $U_i = (g^{u_i}, h^{v_i}, k^{u_i + v_i})$ for random $u_i, v_i \xleftarrow{\$} \mathbb{Z}_p$. (It will later become clear that we will sign vectors of group elements, where each consists of 8 group elements. This is the reason why we choose $n = 8$ here).

- Each tree node N is identified by a four-tuple of group elements $N = (X, z) \in \mathbb{G}^4$, where $z \xleftarrow{\$} \mathbb{G}$ is random and $X = (g^x, h^y, k^{x+y})$ for random $x, y \xleftarrow{\$} \mathbb{Z}_p$.

- To each node $N = (X, z)$ of the tree we assign the public key $vk = (g, h, k, U_1, \ldots, U_8, X, z)$ with corresponding secret key $sk_N = (u_1, v_1, \ldots, u_8, v_8, x, y)$. Note that this is a valid key pair for the one-time signature scheme from Section 3.2, instantiated such that vectors of 8 group elements can be signed. Note also that each node is identified by $(X, z) \in \mathbb{G}^4$, so that we can sign two child nodes with each public key.

- The tree is constructed — on the fly — as follows. Let $N_L = (X_L, z_L)$ and $N_R = (X_R, z_R)$ be the two children of node $N = (X, z)$. Then a signature of the message $M = (N_L, N_R) = (X_L, z_L, X_R, z_R) \in \mathbb{G}^8$ under secret key sk_N authenticates N_L and N_R as children of N. (This is why we chose $n = 8$).

- This gives the following signature scheme, which can be used to sign 8-tuples of elements of $\mathbb{G}$:
  - The public key of the signature scheme consists of $(g, h, k, U_1, \ldots, U_8)$ and the root node $N_0 = (X_0, z_0)$, the secret key consists of the discrete logarithms $(u_1, v_1, \ldots, u_8, v_8, x_0, y_0)$.
  - In order to sign a message $M = M_d \in \mathbb{G}^8$, select a leaf node N_d which has not been used before. Let $N_{d-1}, \ldots, N_0$ denote the path from N_d to the root N_0, and for all N_i ($i \in \{1, \ldots, d-1\}$), let N_i^{co} denote the sibling of N_i. Let

$$M_{i-1} := \begin{cases} (N_i, N_i^{\mathsf{co}}), & \text{if } N_i^{\mathsf{co}} \text{ is the right-hand sibling of } N_i, \\ (N_i^{\mathsf{co}}, N_i), & \text{if } N_i^{\mathsf{co}} \text{ is the left-hand sibling of } N_i. \end{cases} \tag{4}$$

    A signature for M_d consists of all pairs $M_{d-1}, \ldots, M_0$ and signatures $(\sigma_d, \ldots, \sigma_0)$ such that each signature σ_i authenticates M_i as child of node N_{i-1}.

  See Figure 1 in Appendix A.1 for an illustration.

We note that, strictly speaking, the described scheme is not structure-preserving. (The reason is the case distinction (4).) We will show in Section 4.2 how to implement our scheme in a structure-preserving way.

Full Scheme Description. The tree-based scheme $\mathsf{TSig} = (\mathsf{TSig.Gen}, \mathsf{TSig.Sign}, \mathsf{TSig.Vfy})$ is defined as follows.

$\mathsf{TSig.Gen}(g, h, k)$**:** Given generators $g, h, k \in \mathbb{G}$, choose integers $u_1, v_1, \ldots, u_8, v_8, x_0, y_0 \xleftarrow{\$} \mathbb{Z}_p$ and a random generator $z_0 \xleftarrow{\$} \mathbb{G}$. Set $U_i := (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for $i \in [n]$ and $X_0 := (g^{x_0}, h^{y_0}, k^{x_0+y_0})$. Set

$$vk = (g, h, k, U_1, \ldots, U_8, X_0, z_0) \quad \text{and} \quad sk = (u_1, v_1, \ldots, u_8, v_8, x_0, y_0)$$

and return (vk, sk). This defines the root of the tree as $N_0 = (X_0, z_0)$.

$\mathsf{TSig.Sign}(sk, M)$**:** To sign a message $M = M_d = (m_1, \ldots, m_8) \in \mathbb{G}^n$,

1. generate the leftmost unused leaf $N_d = (X_d, z_d)$ of the tree by choosing $z_d \xleftarrow{\$} \mathbb{G}$, $x_d, y_d \xleftarrow{\$} \mathbb{Z}_p$, and setting $X_d = (g^{x_d}, h^{y_d}, k^{x_d+y_d})$. This defines the key pair associated to N_d as $vk_d = (g, h, k, U_1, \ldots, U_8, X_d, z_d)$ and $sk_d = (u_1, v_1, \ldots, u_8, v_8, x_d, z_d)$.
2. Then compute all nodes N_i of the tree from N_d up to the root that have not been visited before by choosing $x_i, y_i \xleftarrow{\$} \mathbb{Z}_p$ and $z_i \xleftarrow{\$} \mathbb{G}$ and setting $X_i = (g^{x_i}, h^{y_i}, k^{x_i+y_i})$ and $N_i = (X_i, z_i)$. This defines the keys associated to N_i as $vk_i = (g, h, k, U_1, \ldots, U_8, X_i, z_i)$ and $sk_i = (u_1, v_1, \ldots, u_8, v_8, x_i, z_i)$.

   For each node N_i the sibling N_i^{co} is generated the same way, if it has not been visited before.

   Each node N_i and its sibling N_i^{co} and their respective key pairs (vk_i, sk_i) are stored as long as they are an ancestor or the sibling of an ancestor of the current signing leaf.
3. The message M_d is authenticated by a signature σ_d over M_d under secret key sk_d

4. For each pair of siblings N_i, N_i^{co}, $i \in 1, \ldots, d$, let $M_{i-1} \in \mathbb{G}^8$ denote the tuple

$$M_{i-1} := \begin{cases} (N_i, N_i^{\mathsf{co}}), & \text{if } N_i^{\mathsf{co}} \text{ is the right-hand sibling of } N_i, \\ (N_i^{\mathsf{co}}, N_i), & \text{if } N_i^{\mathsf{co}} \text{ is the left-hand sibling of } N_i. \end{cases}$$

   The pair of siblings M_{i-1} is authenticated with respect to ancestor N_{i-1} by a signature σ_{i-1} under key sk_{i-1} of node N_{i-1}.

5. The resulting signature over message M_d is consists of $\sigma = (M_{d-1}, \ldots, M_0, \sigma_d, \ldots, \sigma_0) \in \mathbb{G}^{10d+2}$.

$\mathsf{TSig.Vfy}(vk, M_d, \sigma)$**:** A given signature σ of message vector M_d verified by checking that σ_i is a valid one-time signature over M_i for all $i \in [d]$.

More precisely, note that each node $M_i = (X_L, z_L, X_R, z_r)$ induces two verification keys, namely $vk_{i,L} = (g, h, k, U_1, \ldots, U_8, X_L, z_L)$ and $vk_{i,R} = (g, h, k, U_1, \ldots, U_8, X_R, z_R)$. Therefore, for each $i \in [d]$ the verification algorithm tests whether σ_i is a valid one-time signature over M_i under either key $vk_{i,L}$ or $vk_{i,R}$. If this is true for all i, then it outputs 1, otherwise 0.

We note that using the technique of Goldreich [30] the above scheme can be made stateless by using a pseudo-random function to derive the randomness $x, y \in \mathbb{Z}_p$ and $z \in \mathbb{G}$ of each node $X = ((g^x, h^y, k^{x+y}), z)$ and to determine the leaf used to sign a given message.

In this section we prove that this scheme is secure against non-adaptive attacks, which suffices for our main application. In addition, we describe in Section 3.4 how to combine the non-adaptively secure scheme with the algorithm from Lemma 1 to obtain an adaptively-secure scheme with tight reduction.

Theorem 3. *Suppose there exists a non-adaptive adversary $\mathcal{A}$ that (ϵ, t, q)-breaks the EUF-CMA security of TSig. Then there exists an adversary $\mathcal{B}$ that (ϵ', t')-breaks the DLIN assumption in $\mathbb{G}$, where t' is roughly the runtime of the EUF-CMA experiment with $\mathcal{A}$, and $\epsilon' = \epsilon/8$.*

Note that the success probability ϵ' of the DLIN-breaker $\mathcal{B}$ is independent of the number q of chosen-message queries issued by $\mathcal{A}$.

Proof. Let us fix some notation. In the sequel we will say that note N_j is a *direct ancestor* of leaf N_d, if N_j lies on the path from N_d to the root N_0. We say that N_j^{co} is an *indirect ancestor* of leaf N_d, if it is a sibling of a direct ancestor of N_d.

We will denote with $N_{d,1}, \ldots, N_{d,q}$ the q leaves of the tree that are used to sign the chosen-messages of $\mathcal{A}$, where $N_{d,i}$ is used to sign message $m^{(i)}$. Note that these nodes are the q "leftmost" nodes of the tree, since they are used from left to right (this is the stateful case, which is simpler to analyze; in the stateless variant the proof idea will be the same, but the description will be more complicated). We denote with $\mathcal{N}_{\mathsf{leaves}} := \{N_{d,1}, \ldots, N_{d,q}\}$ the set of all these leaves, with $\mathcal{N}_{\mathsf{dir}}$ denote the set of all *direct* ancestors of nodes in $\mathcal{N}_{\mathsf{leaves}}$, and with $\mathcal{N}_{\mathsf{out}}$ the set of all indirect ancestors of nodes in $\mathcal{N}_{\mathsf{leaves}}$ which are *not* a direct ancestor of any $N_i \in \mathcal{N}_{\mathsf{leaves}}$. See Figure 2 in Appendix A.2 for an illustration.

Let $\mathcal{N} := \mathcal{N}_{\mathsf{leaves}} \cup \mathcal{N}_{\mathsf{dir}} \cup \mathcal{N}_{\mathsf{out}}$. Note also that the set of all tree nodes which the adversary ever sees throughout the security experiment is identical to $\mathcal{N}$.

We proceed in a sequence of games. Let X_i denote the event that $\mathcal{A}$ wins in Game i.

Game 0. This is the original (non-adaptive) security experiment. Recall that the adversary has to select a list of messages $M^{(1)}, \ldots, M^{(q)} \in \mathbb{G}^8$ to be signed at the beginning of the game, before seeing the public key. By assumption we have $\Pr[X_0] = \epsilon$.

Game 1. Now we change the way the tree is set up. In the sequel let $\mathcal{T}_8$ denote the algorithm $\mathcal{T}_n$ from Lemma 2 instantiated with $n = 8$. Moreover, to abbreviate notation let us write $\mathcal{T}_8(j, m)$ shorthand for $\mathcal{T}_8(j, G, H, U_j, (u_i, v_i)_{i\in[8\setminus j]}, m)$.

The challenger in this game chooses a random dummy message $\tilde{M} \xleftarrow{\$} \mathbb{G}^8$ and an index $j \xleftarrow{\$} [8]$ at the beginning of the game. It constructs the tree from bottom up, by proceeding as follows. Given the list of chosen-messages $M^{(1)}, \ldots, M^{(q)}$, each node $N_d^{(i)} \in \mathcal{N}_{\mathsf{leaves}}$ is determined by computing

$$(X_d^{(i)}, z_d^{(i)}, s_d^{(i)}, t_d^{(i)}) \xleftarrow{\$} \mathcal{T}_8(j, M^{(i)})$$

and setting $N_d^{(i)} := (X_d^{(i)}, z_d^{(i)})$. Note that $(s^{(i)}, t^{(i)})$ is a valid signature that authenticates $M^{(i)}$ as child of $N_d^{(i)}$.

Each node $N \in \mathcal{N}_{\mathsf{out}}$ is computed by running

$$(X, z, s, t) \xleftarrow{\$} \mathcal{T}_8(j, \tilde{M}).$$

and setting $N := (X, z)$. The tuple (s, t), which is a valid signature that authenticates the dummy message $\tilde{M}$ as a child of N, is discarded, since the challenger does not need it.

Each ancestor $N \in \mathcal{N}_{\mathsf{dir}}$ of two nodes (N_L, N_R) is computed, from the leaves up to the root, by running

$$(X, z, s, t) \xleftarrow{\$} \mathcal{T}_8(j, (N_L, N_R)).$$

Again (s, t) is a valid signature authenticating (N_L, N_R) as children of N.

Note that $U_j \in \mathsf{DLIN}(g, h, k)$, thus, due to the properties of algorithm $\mathcal{T}_n$ from Lemma 2, Game 1 is perfectly indistinguishable from Game 0 for the adversary, which implies $\Pr[X_1] = \Pr[X_0]$.

Game 2. Now we are ready to construct our DLIN distinguisher $\mathcal{B}$. $\mathcal{B}$ receives as input a DLIN challenge (g, h, k, U), and proceeds identical to the challenger in Game 1, except that it defines U_j as $U_j := U$. Note that $\mathcal{B}$ is able to set-up the tree and compute all signatures as before, by running Algorithm $\mathcal{T}_8(j, G, H, U_j, (u_i, v_i)_{i\in[8\setminus j]}, m)$.

If $\mathcal{A}$ outputs a forgery (M_d^*, σ^*), consisting of a signature $\sigma = (M_{d-1}^*, \ldots, M_0^*, \sigma_d^*, \ldots, \sigma_0^*)$ and message $M_d^* \in \mathbb{G}^8$, let $N_d^*, \ldots, N_1^*, N_0$ denote the path from M_d^* to the root N_0 given by signature σ.

$\mathcal{B}$ determines $\delta \in \{0, \ldots, d\}$ with δ the largest index such that $N_\delta^* \in \mathcal{N}$. Recall that N_δ^* was previously computed by $\mathcal{B}$ by running $(X, z, s, t) \xleftarrow{\$} \mathcal{T}_8(j, M')$ for some input $M' = (m'_1, \ldots, m'_8)$ and setting $N_\delta^* := (X, z)$. So in particular equation

$$\prod_{i=1}^{8} E(U_i, m'_i) \cdot E(G, s) \cdot E(H, t) = E(X, z)$$

is satisfied. Moreover, since $\sigma_\delta^* = (s_\delta^*, t_\delta^*)$ is a valid signature over message $M_\delta^* = (m_{\delta,1}^*, \ldots, m_{\delta,8}^*)$, equation

$$\prod_{i=1}^{8} E(U_i, m_{\delta,i}^*) \cdot E(G, s_\delta^*) \cdot E(H, t_\delta^*) = E(X, z) \tag{5}$$

is satisfied, too. Note also that we must have $M^*_\delta \neq M$, as otherwise this is not a valid forgery or δ is not the largest index such that $N^*_\delta \in \mathcal{N}$. $\mathcal{B}$ outputs 1 if $m^*_{\delta,j} \neq m'_j$ holds (and 0 else).

Analysis. If $U \in \mathsf{DLIN}(g,h,k)$, then by Property (i) and (ii) of Lemma 2, Game 2 is perfectly indistinguishable from Game 1. Therefore in this case $\mathcal{A}$ outputs a forgery with probability ϵ by assumption. Since $M_\delta \neq M'$ we have $m^*_{\delta,j} \neq m'_j$ with probability at least $1/8$. Thus, if $U \xleftarrow{\$} \mathsf{DLIN}(g,h,k)$ then $\mathcal{B}$ outputs 1 with probability $\epsilon/8$.

If $U \xleftarrow{\$} \mathbb{G}^3$ then we have $U \in \mathsf{DLIN}(g,h,k)$ with probability $1/p$. Moreover, if $U \notin \mathsf{DLIN}(g,h,k)$ then by Property (iii) of Lemma 2 there exists no M^*_δ such that Equation (5) is satisfied and $m^*_{\delta,j} \neq m'_j$. Thus, if $U \xleftarrow{\$} \mathbb{G}^3$ then $\mathcal{B}$ outputs 1 with probability $1/p$. □

3.4 Structure-Preserving Signatures Secure Against Adaptive Adversaries

It is well-known that one can combine a non-adaptively secure signature scheme with a non-adaptively secure one-time signature scheme to obtain an adaptively secure signature scheme [24]. Thus, by combining our one-time signature scheme from Section 3.1 (or the one-time signature scheme for vectors from Section 3.2) with our tree-based signature scheme from Section 3.3 we obtain an adaptively secure structure-preserving signature scheme.

The generic reduction from [24] loses a factor of q, where q is the number of signatures issued. We would like to note that we can also obtain an adaptively secure scheme with *tight* security reduction. Consider the following signature scheme $\mathsf{Sig} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Vfy})$.

$\mathsf{Sig.Gen}(g,h,k)$**:** Run $(vk_T, sk_T) \xleftarrow{\$} \mathsf{TSig.Gen}(g,h,k)$ and choose $U \xleftarrow{\$} \mathsf{DLIN}(g,h,k)$. Set $vk := (vk_T, U)$ and $sk := sk_T$.

$\mathsf{Sig.Sign}(sk,m)$**:** Let $G := (g,1,k)$ and $H := (1,h,k)$. Compute $(X,z,s,t) \xleftarrow{\$} \mathcal{T}(G,H,U,m)$, where $\mathcal{T}$ is the algorithm from Lemma 1. Set $M := (X,z,g,g,g,g) \in \mathbb{G}^8$ (the four g elements are a simple padding to obtain the correct message size for TSig) and compute $\sigma_T \xleftarrow{\$} \mathsf{TSig.Sign}(sk_T, M)$. The resulting signature is $\sigma = (\sigma_T, X, z, s, t)$.

$\mathsf{Sig.Vfy}(vk,m,\sigma)$**:** Parse $\sigma = (\sigma_T, X, z, s, t)$. Set $M := (X,z,g,g,g,g)$ and output 1 if

$$\mathsf{TSig.Vfy}(vk_T, M, \sigma_T) = 1 \quad \text{and} \quad E(U,m) \cdot E(G,s) \cdot E(H,t) = E(X,z).$$

Otherwise output 0.

The security proof is rather simple and along the lines of [24], therefore we only sketch it. For $i \in [q]$ let $m^{(i)}$ denote the i-th message chosen by the adversary, and let $\sigma^{(i)} = (\sigma_T^{(i)}, X^{(i)}, z^{(i)}, s^{(i)}, t^{(i)})$ denote the corresponding signature. In the security proof we consider two types of adversaries.

Type I. A Type-I adversary outputs a message-signature pair $(m, (\sigma_T, X, z, s, t))$ such that there exists $\sigma^{(i)}$ with $(X^{(i)}, z^{(i)}) = (X, z)$. Since this is impossible if $U \notin \mathsf{DLIN}(g,h,k)$, due to Property (iii) of Lemma 1, this allows us to distinguish whether $U \in \mathsf{DLIN}(g,h,k)$ or not.

Type II. A Type-II adversary outputs a message-signature pair $(m, (\sigma_T, X, z, s, t))$ such that there exists no $\sigma^{(i)}$ with $(X^{(i)}, z^{(i)}) = (X, z)$. In this case the adversary has to compute a new signature for TSig. In order to reduce to the non-adaptive security of TSig, we generate

$U := G^u \cdot H^v$ and all $X^{(i)} = G^{x_i} \cdot H^{y_i}$ for random integers $u, v, x_i, y_i \xleftarrow{\$} \mathbb{Z}_p$, $i \in [q]$, at the beginning of the game, and use the integers $u, v, (x_i, y_i)_{i\in[q]}$ as a trapdoor to sign messages, similar to the signing algorithm of OTSig.

Besides the one-time signature based construction of [24] there is also a construction based on chameleon hashes [42], which offers a tight reduction. We are not aware of any suitable chameleon hash functions which are *structure-preserving*, but we would like to note that our use of algorithm $\mathcal{T}$ in the above construction is very similar — except that it does not meet the syntactic definition of a chameleon hash.

4 Tightly Simulation-Sound NIZK Proofs for Pairing Product Equations

In this section we use the signature scheme from Section 3.3 to construct a NIZK proof for satisfiability of pairing product equations whose security reduces tightly to the DLIN assumption. "Tight" means here that the success probability of the reduction is independent of the number of simulated proofs the adversary sees. The construction is a special case of Groth-Sahai (GS) proofs [35], and uses a trick from [34, Section 4] to express the disjunction of two sets of pairing product equations as one set.

4.1 Non-Interactive Zero-Knowledge Proofs

Let R be a binary relation and let $\mathcal{L} := \{x : \exists w \text{ s.t. } R(x, w) = 1\}$ be the language defined by R. A non-interactive zero-knowledge proof system $\mathsf{NIZK} = (\mathsf{NIZK.Gen}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vfy})$ for $\mathcal{L}$ consists of three algorithms. The common reference string generation algorithm $crs \xleftarrow{\$} \mathsf{NIZK.Gen}(\kappa)$ takes as input a security parameter κ and outputs a common reference string crs. Algorithm $\pi \xleftarrow{\$} \mathsf{NIZK.Prove}(crs, x, w)$ takes as input crs, statement x, and a witness w that $x \in \mathcal{L}$, and outputs a proof π. The verification algorithm $\mathsf{NIZK.Vfy}(crs, \pi, x) \in \{0, 1\}$ takes as input proof π and statement x. We say that $\mathsf{NIZK.Vfy}$ *accepts* if $\mathsf{NIZK.Vfy}(crs, \pi, x) = 1$. We say that $\mathsf{NIZK.Vfy}$ *rejects* if $\mathsf{NIZK.Vfy}(crs, \pi, x) = 0$.

NIZK is $(\epsilon_{\mathsf{ZK}}, \epsilon_{\mathsf{snd}}, \epsilon_{\mathsf{simsnd}}, t, Q)$-secure, if the following holds.

Perfect completeness. For each $(x, w) \in R$, each parameter κ, and each $crs \xleftarrow{\$} \mathsf{NIZK.Gen}(\kappa)$ holds that

$$\Pr[\mathsf{NIZK.Vfy}(crs, \pi, x) = 1 : \pi \xleftarrow{\$} \mathsf{NIZK.Prove}(crs, x, w)] = 1.$$

Soundness. For all adversaries $\mathcal{A}$ running in time t holds that

$$\Pr[\mathcal{A}(crs) = (x, \pi) : x \notin \mathcal{L} \wedge \mathsf{NIZK.Vfy}(crs, \pi, x) = 1] \leq \epsilon_{\mathsf{snd}}$$

Zero knowledge. There exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, such that $(crs, td) \xleftarrow{\$} \mathcal{S}_0(\kappa)$ generates a common reference string and trapdoor information td, and $\pi \xleftarrow{\$} \mathcal{S}_1(crs_{\mathsf{sim}}, td, x)$ generates a simulated proof π for statement x (where not necessarily $x \in \mathcal{L}$).

Let $crs_{\mathsf{real}} \xleftarrow{\$} \mathsf{NIZK.Gen}(\kappa)$ and let $\mathcal{O}_{\mathsf{real}}$ denote an oracle that takes as input $(x, w) \in R$ and returns $\mathsf{NIZK.Prove}(crs_{\mathsf{real}}, x, w)$. Let $(crs_{\mathsf{sim}}, td) \xleftarrow{\$} \mathcal{S}_0(\kappa)$ and let $\mathcal{O}_{\mathsf{sim}}$ return $\mathcal{S}_1(crs_{\mathsf{sim}}, td, x)$ on input $(x, w) \in R$.

We require that

$$\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{real}}}(crs_{\mathsf{real}}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{sim}}}(crs_{\mathsf{sim}}) = 1] \leq \epsilon_{\mathsf{ZK}}$$

for all $\mathcal{A}$ running in time at most t that issue at most Q oracle queries.

Simulation soundness. For $crs \xleftarrow{\$} \mathcal{S}_0(\kappa)$ and for all adversaries $\mathcal{A}$ running in time t that may query $\mathcal{S}_1$ at most Q times for simulated proofs $\pi_1, \ldots, \pi_Q$ of arbitrary statements $x_1, \ldots, x_Q$ (where possibly $x_i \notin \mathcal{L}$ for some or all $i \in [Q]$) holds that

$$\Pr[\mathcal{A}^{\mathcal{S}_1}(crs) = (x, \pi) : x \notin \mathcal{L} \wedge (x, \pi) \neq (x_i, \pi_i) \forall i \in [Q] \wedge \mathsf{NIZK.Vfy}(crs, \pi, x) = 1] \leq \epsilon_{\mathsf{simsnd}}$$

We will also use a variant of NIZK proof systems as a technical building block. Namely, a **(perfectly) non-interactive witness-indistinguishable (NIWI) proof system** is defined like a NIZK proof system above, with the following difference. Instead of the zero-knowledge and simulation-soundness properties, we require (perfect) witness-indistinguishability: for all *crs* in the image of NIZK.Gen, and all $(x, w_1), (x, w_2) \in R$ (for the same x), we require that the distributions induced by $\mathsf{NIZK.Prove}(crs, x, w_1)$ and $\mathsf{NIZK.Prove}(crs, x, w_2)$ are identical.

4.2 Building Blocks

Pairing Product Equations. Following [34, 35], a pairing product equation (PPE) s of length ℓ over $\mathbb{G}$ is an equation of the form

$$\prod_{j=1}^{\ell} e(Q_{j,0}, Q_{j,1}) = 1 \quad \text{with} \quad Q_{j,b} = A_{j,b} \cdot \prod_{i=1}^{\nu} X_i^{\alpha_{j,b,i}} \tag{6}$$

where the $A_i \in \mathbb{G}$ and $\alpha_{j,b,i} \in \mathbb{Z}_p$ are constants, and the $X_i \in \mathbb{G}$ are variables. We say that a vector $\vec{x} = (x_1, \ldots, x_\nu) \in \mathbb{G}^\nu$ *satisfies* the equation, if Equation 6 holds when setting $X_i = x_i$. A set S of pairing product equations is *satisfiable*, if there exists a vector $\vec{x}$ that satisfies all equations $s \in S$ simultaneously. In the following, we will consider sets of satisfiable PPEs as languages for NIZK proof systems.

Disjunctions of Pairing Product Equations. Groth [34, Section 4.8] shows how to express the disjunction of several sets of PPEs through one set. Concretely, given n sets $S_1, \ldots, S_n$ of PPEs, he constructs a set $S := OR(S_1, \ldots, S_n)$ of PPEs such that

- every solution $\vec{x}$ that satisfies S allows to efficiently derive a solution $\vec{x}_i$ of at least one S_i,
- every solution $\vec{x}_i$ of some S_i allows to efficiently derive a solution $\vec{x}$ of S,
- if S_i has ν_i variables X_i and consists of equations of total length ℓ_i, then S has total length $2\ell + 1$ for $\ell = \sum_{i=1}^{n} \ell_i$, and $(\sum_{i=1}^{n} \nu_i) + n + \ell$ variables.

NIWI Proofs for a Set of Pairing Product Equations. Groth and Sahai [35] present an efficient non-interactive witness-indistinguishable proof system for arbitrary sets of PPEs. Their system features a CRS *crs* that can be chosen either to be *hiding* or to be *binding*. If *crs* is hiding, then the resulting proofs are perfectly witness-indistinguishable. If *crs* is binding, the resulting proofs enjoy perfect soundness, and become extractable: a special trapdoor to *crs* allows to extract

a witness $\vec{x}$ from a valid proof. Hiding and binding CRSs are computationally indistinguishable under the DLIN assumption in the underlying group $\mathbb{G}$. When implemented over a DLIN-group (as will be the case in our setting), their system has the following efficiency properties (cf. Figure 2 in [35]):

- the CRS contains 6 $\mathbb{G}$-elements,
- each used variable X_i results in 3 $\mathbb{G}$-elements in the proof,
- each PPE incurs 9 $\mathbb{G}$-elements in the proof.

TSig-Verification as a Set of Pairing Product Equations. The verification algorithm of our weakly-secure DLIN-based signature scheme TSig from Section 3 can be expressed as a set of PPEs. Concretely, assume a verification key $vk = (g, h, k, U_1, \ldots, U_8, X_0, z_0)$ for TSig, and a message $M = (m_{d,1}, \ldots, m_{d,8}) \in \mathbb{G}^8$. Recall that a TSig-signature $\Sigma \in \mathbb{G}^{10d+2}$ determines OTSig-verification-keys vk_i, messages M_i, and signatures $\Sigma^{(i)}$ such that Σ is valid iff $\Sigma^{(i)}$ is a valid OTSig-signature of $M_i = (m_{i,j})_{j=1}^8$ under vk_{i-1} for all $i \in [d]$. Hence, verification amounts to checking a set $S_{vk,M}^{\mathsf{TSig}} = \{OR(S_{L,i}, S_{R,i})\}_{i\in[d]}$, where $S_{D,i}$ (for $i \in [d]$ and $D \in \{L, R\}$) is given by

$$S_{D,i} = \left\{ \left(\prod_{j=1}^{8} E(U_i, m_{i,j}) \right) \cdot E(G, s_i) \cdot E(H, t_i) = E(X_{D,i}, z_{D,i}) \right\}$$

of PPEs, where $G, H, U_1, \ldots, U_8 \in \mathbb{G}^3$ and the $m_{d,j} \in \mathbb{G}$ are constants, and the $m_{i,j} \in \mathbb{G}$ (for $i \in [d-1]$), $X_{L,i}, X_{R,i} \in \mathbb{G}^3$, and $z_{L,i}, z_{R,i}, s_i, t_i \in \mathbb{G}$ (for $i \in [d]$) are variables.

Tightly Secure One-Time Signatures. As a final preparation, we require a means to secure proofs from tampering. Typically, this is done via a one-time signature scheme (as, e.g., in [46]). For our purposes, however, we require *tightly secure* (but not necessarily structure-preserving) one-time signatures. To enable a tight reduction, we will consider signature schemes with an algorithm $\mathsf{TOTS.Param}$ that outputs common system parameters $pars_{\mathsf{tots}}$.

Definition 4. The security experiment for strong *n-fold* one-time EUF-CMA security is identical to the strong one-time EUF-CMA experiment (see Section 2), except that the adversary $\mathcal{A}$ gets the scheme's public parameters and n verification keys pk_i ($i \in [n]$) as input. $\mathcal{A}$ may request (up to) one signature for each pk_i, and may finally output a forged signature under exactly one pk_i. We say that a signature scheme TOTS is strongly n-fold one-time (ϵ, t, q)-secure if there is no $\mathcal{A}$ that (ϵ, t, q)-breaks the strong EUF-CMA security of TOTS.

We now construct a signature scheme TOTS whose n-fold one-time EUF-CMA security experiment reduces to the discrete logarithm problem in $\mathbb{G}$. The corresponding reduction loses only a factor of 2, independently of n.

$\mathsf{TOTS.Param}(\kappa)$**:** The common parameters $pars_{\mathsf{tots}}$ are a two generators g, h_0 and a collision-resistant hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$.

$\mathsf{TOTS.Gen}(pars_{\mathsf{tots}}, pars_{\mathsf{tots}})$**:** Uniformly choose exponents $\omega_1, s_1 \in \mathbb{Z}_p$ and output $vk_{\mathsf{tots}} := (h_1, c_1) := (g^{\omega_1}, g^{s_1})$ and $sk_{\mathsf{tots}} := (\omega_1, s_1)$.

$\mathsf{TOTS.Sign}(pars_{\mathsf{tots}}, sk_{\mathsf{tots}}, m)$**:** Uniformly choose $r_0 \in \mathbb{Z}_p$ and compute $c_0 := g^{\mathsf{H}(m)}h_0^{r_0}$ and $r_1 = (s_1 - \mathsf{H}(c_0))/\omega_1 \bmod p$, such that $c_1 = g^{\mathsf{H}(c_0)}h_1^{r_1}$. Output $\sigma = (r_0, r_1)$.

$\mathsf{TOTS.Vfy}(pars_{\mathsf{tots}}, vk_{\mathsf{tots}}, m, \sigma)$**:** Parse $\sigma =: (r_0, r_1)$, and set $c_0 := g^{\mathsf{H}(m)}h_0^{r_0}$. If $c_1 = g^{\mathsf{H}(c_0)}h_1^{r_1}$, output 1, else 0.

Note that TOTS essentially consists of a two-fold application of Pedersen commitments [49], interpreted as one-time signatures.

Lemma 3. *Let $n \in \mathbb{N}$. Then scheme TOTS above is strongly n-fold one-time EUF-CMA secure assuming H is collision-resistant and the discrete logarithm assumption in $\mathbb{G}$ holds. Concretely, $\epsilon_{\mathsf{n\text{-}cma}} \leq 2\epsilon_{\mathsf{dlog}} + \epsilon_{\mathsf{crhf}}$ for the advantage $\epsilon_{\mathsf{n\text{-}cma}}$ of an arbitrary n-fold one-time EUF-CMA adversary $\mathcal{A}$, and the advantages ϵ_{dlog} of a corresponding DLOG-solver $\mathcal{B}$ and ϵ_{crhf} of a H-collision-finder $\mathcal{C}$.*

The proof is fairly standard, therefore we only sketch it.

Proof sketch. Let m^* and $\sigma^* = (r_0^*, r_1^*)$ denote forged message and signature; let $vk^*_{\mathsf{tots}} = (h_1^*, c_1^*)$ be the corresponding verification key. Let m be the message signed for $\mathcal{A}$ under vk^*_{tots}, and let $\sigma = (r_0, r_1)$ be the corresponding signature. Write $c_0^* = g^{\mathsf{H}(m^*)}h^{r_0^*}$ and $c_0 = g^{\mathsf{H}(m)}h^{r_0}$. We can assume $(m^*, \sigma^*) \neq (m, \sigma)$ and $\mathsf{TOTS.Vfy}(pars_{\mathsf{tots}}, vk^*_{\mathsf{tots}}, m^*, \sigma^*) = 1$. Distinguish the following cases:

H-collisions: If $m^* \neq m$ and $\mathsf{H}(m^*) = \mathsf{H}(m)$, or if $c_0^* \neq c_0$ and $\mathsf{H}(c_0^*) = \mathsf{H}(c_0)$, then $\mathcal{A}$ has found a H-collision. The probability for a H-collision can be bounded by ϵ_{dlog} using a straightforward reduction. In the following we silently assume that no H-collision took place.

$c_0^* = c_0$: We can assume $m^* \neq m$, since $m^* = m$ would imply $r_0^* = r_0$ and $r_1^* = r_1$. Hence, we have found two different decompositions $g^{\mathsf{H}(m^*)}h_0^{r_0^*} = c_0 = g^{\mathsf{H}(m)}h_0^{r_0}$ of c_0, which allows to deduce $\log_g(h_0)$. A straightforward reduction to the DLOG assumption bounds this event with ϵ_{dlog}.

$c_0^* \neq c_0$: In this case, we have found two different decompositions $g^{\mathsf{H}(c_0^*)}h_1^{r_1^*} = c_1 = g^{\mathsf{H}(m)}h_1^{r_1}$ of c_1, from which we can derive $\log_g(h_1)$. A reduction $\mathcal{B}$ to the DLOG problem can proceed as follows, given input (g, h). First, $\mathcal{B}$ sets up $h_0 = g^{\omega_0}$ for known ω_0; all verification keys are generated as $(h_1, c_1) = (h^\alpha, g^{\mathsf{H}(c_0)}h^{\alpha r_1})$ for $c_0 = g^{s_0} \in \mathbb{G}$ and uniform $\alpha, r_1, s \in \mathbb{Z}_p$ (chosen fresh for each key of course). This allows to generate signatures by setting $r_0 := (s_0 - \mathsf{H}(m))/\omega_0 \bmod p$ and $\sigma = (r_0, r_1)$. Finally, *any* forged signature with $c_0^* \neq c_0$ can then be used to derive $\log_g(h_1) = \log_g(h^\alpha)$ for some known α, from which $\log_g(h)$ follows. Thus, $c_0^* \neq c_0$ can occur with probability at most ϵ_{dlog}.

Taking things together yields the claim. Note that the derived bound for $\epsilon_{\mathsf{n\text{-}cma}}$ does not depend on n. □

4.3 Our Simulation-Sound NIZK Proof System

We are now ready to describe our proof system for a set S of PPEs. Intuitively, we will prove, using GS NIWI proofs, that *either* S is satisfiable, *or* that we know a TSig-signature for a "suitably unique" value (or both). The "suitably unique" value will be a verification key for a strongly (and tightly) secure one-time signature scheme. Simulated proofs prove the "or" branch of the statement, using TSig's signing key. Simulation-soundness (and also soundness) follows from the existential

unforgeability of TSig, and from the soundness of GS proofs. A bit more formally, consider the following non-interactive proof system:

$\mathsf{NIZK.Gen}(\mathbb{G})$ outputs a CRS $crs = (crs_{\mathsf{GS}}, vk, pars_{\mathsf{tots}})$, where (a) crs_{GS} is a binding CRS for DLIN-based GS proofs over $\mathbb{G}$, (b) vk is a verification key for TSig, (c) $pars_{\mathsf{tots}}$ are parameters for a strongly n-fold EUF-CMA secure one-time signature scheme TOTS, whose security reduction does not depend on n.

$\mathsf{NIZK.Prove}(crs, S, \vec{x})$ takes as input a CRS $crs = (crs_{\mathsf{GS}}, vk)$, a set S of PPEs, and a satisfying assignment $\vec{x} = (x_1, \ldots, x_\nu) \in \mathbb{Z}_p^\nu$. Then, $\mathsf{NIZK.Prove}$ samples a TOTS keypair $(vk_{\mathsf{tots}}, sk_{\mathsf{tots}})$ and outputs the tuple $\pi = (\pi_{\mathsf{GS}}, vk_{\mathsf{tots}}, \sigma_{\mathsf{tots}})$. Here, π_{GS} is a GS proof (using CRS crs_{GS}) for the set $OR(S, S^{\mathsf{TSig}}_{vk, vk_{\mathsf{tots}}})$ of PPEs (as described above), and σ_{tots} is a TOTS-signature under vk_{tots} of (S, π_{GS}).

$\mathsf{NIZK.Vfy}(crs, S, \pi)$ takes a CRS $crs = (crs_{\mathsf{GS}}, vk)$, a set S of PPEs, and a proof π as above, verifies σ_{tots}, and then checks π as a GS proof for the set $OR(S, S^{\mathsf{TSig}}_{vk, vk_{\mathsf{tots}}})$ of PPEs.

Theorem 4. *The proof system* NIZK *just described is* $(\epsilon_{\mathsf{ZK}}, \epsilon_{\mathsf{snd}}, \epsilon_{\mathsf{simsnd}}, t, Q)$*-secure, where* $\epsilon_{\mathsf{ZK}} \leq 2|\epsilon_{\mathsf{GS}}|$ *and* $\epsilon_{\mathsf{snd}}, \epsilon_{\mathsf{simsnd}} \leq \epsilon_{\mathsf{tots}} + \epsilon_{\mathsf{tsig}}$ *for the advantages of suitable adversaries on the indistinguishability of hiding and binding GS CRSs, the strong* Q*-fold one-time EUF-CMA security of* TOTS*, and the weak EUF-CMA security of* TSig*. All constructed adversaries have roughly the same runtime as the zero-knowledge, soundness, resp. simulation-soundness experiments (with adversaries of runtime* t*).*

Proof sketch. Before we go through the properties necessary for a simulation-sound NIZK proof system, we describe a simulator $\mathcal{S}$. Initially, $\mathcal{S}$ samples a CRS $crs = (crs_{\mathsf{GS}}, vk, pk_{\mathsf{CH}}, G, H)$ exactly like $\mathsf{NIZK.Gen}(\mathbb{G})$, but remembers the signing key sk to vk. When asked to generate a proof for some set S of PPEs, $\mathcal{S}$ proceeds like $\mathsf{NIZK.Prove}$, but generates a proof π_{GS} for $OR(S, S^{\mathsf{TSig}}_{vk, vk_{\mathsf{tots}}})$ using a TSig-signature Σ for vk_{tots} as witness.

We now check the properties of our proof system:

Completeness. Perfect completeness follows from the perfect completeness of GS proofs.

Soundness. Assume an adversary $\mathcal{A}$ who generates an *unsatisfiable* set S of PPEs, along with a (TOTS-signed) GS proof π_{GS} for $OR(S, S^{\mathsf{TSig}}_{vk, vk_{\mathsf{tots}}})$. By the strong one-time EUF-CMA security of TOTS, we have that vk_{tots} has not been TSig-signed before. (Note that here, a *tight* reduction to TOTS's strong n-fold one-time EUF-CMA security is possible.) By the perfect soundness of GS proofs, this implies that $\mathcal{A}$ has proven knowledge of a TSig-signature Σ for a fresh message vk_{tots}. Since crs_{GS} is binding, we can extract that signature using the GS witness-extraction trapdoor and break the weak existential unforgeability of TSig. We obtain that the soundness error ϵ_{snd} is at most $\epsilon_{\mathsf{tots}} + \epsilon_{\mathsf{tsig}}$ for the advantages of suitable TOTS- and TSig-forgers.[6]

Zero-knowledge. First note that simulated and real CRSs are identically distributed. Furthermore, simulated proofs differ from real proofs only by their GS-witness. A straightforward

[6] We note that perfect soundness (i.e., $\epsilon_{\mathsf{snd}} = 0$) can be achieved as in [34, Section 6] with a slightly more complicated setup. In a nutshell, we could add a non-DLIN-tuple $T \in \mathbb{G}^6$ to CRS and prove that *either* S is satisfiable, *or* T is a DLIN-tuple and we know a TSig-signature for vk_{tots} (or both). A simulator $\mathcal{S}$ would of course change T to a DLIN-tuple in simulated CRSs. We omit the details.

reduction to the witness-indistinguishability of GS proofs shows that an adversary's advantage ϵ_{ZK} in distinguishing real from simulated CRSs and proofs is bounded by $2|\epsilon_{\mathsf{GS}}|$ for the distinguishing advantage ϵ_{GS} between hiding and binding GS CRSs.

Simulation-soundness. A similar argument as for soundness shows that the simulation-soundness error $\epsilon_{\mathsf{simsnd}}$ is at most $\epsilon_{\mathsf{tots}} + \epsilon_{\mathsf{tsig}}$ for the advantages of suitable TOTS- and TSig-forgers.

We stress that all the reductions just sketched are tight. In particular, the (simulation-) soundness and zero-knowledge errors do not depend on the number of simulated proofs. □

On the extractability of our proof system. The proof system we have just described inherits a useful feature from GS proofs. Namely, when operated with a binding CRS, a special trapdoor td allows to efficiently extract witnesses from valid proofs. This trapdoor works even in a setting in which an adversary has access to simulated proofs; but obviously, td cannot be used to extract witnesses from the simulated proofs themselves. However, care must be taken when using td: for instance, td allows to distinguish between simulated and non-simulated proofs. Consequently, the Zero-Knowledge property only holds in the absence of td. This "incompatibility" between the Zero-Knowledge and extraction properties is the main reason why we have chosen not to formalize extraction. (For our purposes, an explicit extraction provided by the NIZK proof system will not be necessary.) However, we note that a suitable formalization of "simulation-extractability" can be very useful, see, e.g., Dodis et al. [21].

5 Tight IND-CCA Security in the Multi-User Setting

Syntax. A public-key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Param}, \mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ consists of four algorithms. The parameter generation algorithm $\Pi \xleftarrow{\$} \mathsf{PKE.Param}(\kappa)$ takes as input a security parameter κ and outputs parameters Π. The key generation algorithm $(pk, sk) \xleftarrow{\$} \mathsf{PKE.Gen}(\Pi)$ generates, on input Π, a public encryption key pk and a secret decryption key sk. The probabilistic encryption algorithm $c \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m)$ takes as input a public key pk and a message m, and outputs a ciphertext c. The deterministic decryption algorithm $\mathsf{PKE.Dec}(sk, c) \in \{m, \perp\}$ takes as input a secret key sk and a ciphertext c, and outputs a message m or an error symbol $\perp$. We require the usual correctness properties.

Security. The following security experiment, played between a challenger and an adversary $\mathcal{A}$, is based on the multi-user security definition from [10]. The experiment is parametrized by two integers $\mu, q \in \mathbb{N}$.

1. The challenger runs $\Pi \xleftarrow{\$} \mathsf{PKE.Param}(\kappa)$ once and then $\mathsf{PKE.Gen}(\Pi)$ μ times to generate μ key pairs $(pk^{(i)}, sk^{(i)})$, $i \in [\mu]$. Then it tosses a coin $b \xleftarrow{\$} \{0,1\}$, initializes a list $\mathsf{Clist} := \emptyset$ to the empty list, and defines a counter $j_i := 0$ for each $i \in [\mu]$.

2. The adversary receives the public keys $pk^{(1)}, \ldots, pk^{(\mu)}$ as input. It may query the challenger for two types of operations.

   Encryption queries. The adversary submits two messages m_0, m_1 and an index $i \in [\mu]$. If $j_i \geq q$ then the challenger returns $\perp$. Otherwise it encrypts m_b under public key $pk^{(i)}$

by computing $c = \mathsf{PKE.Enc}(pk^{(i)}, m_b)$. Then it appends (c, i) to Clist, updates counter j_i as $j_i := j_i + 1$, and returns c.

Decryption queries. The adversary submits a ciphertext c and an index $i \in [\mu]$. If $(c, i) \in \mathsf{Clist}$ then the challenger returns $\perp$. Otherwise it returns whatever $\mathsf{PKE.Dec}(sk^{(i)}, c)$ returns.

3. Eventually the adversary $\mathcal{A}$ outputs a bit b'. We say that the adversary *wins* the game, if $b = b'$.

Definition 5. Let $\mathcal{A}$ be an adversary that runs in time t and wins with probability $1/2 + \epsilon$. Then $\mathcal{A}$ (ϵ, t)*-breaks the* (μ, q)*-IND-CCA security* of PKE. If $\mathcal{A}$ never asks any decryption query, then $\mathcal{A}$ (ϵ, t)*-breaks the* (μ, q)*-IND-CPA security* of PKE. For ATK $\in$ {CPA,CCA} we say that PKE is (ϵ, t, μ, q)*-IND-ATK secure*, if there exists no adversary that (ϵ, t)-breaks the (μ, q)-IND-ATK security of PKE.

Note that for ATK $\in$ {CPA,CCA} the classical definitions of IND-ATK security [31, 50] are identical to $(1, 1)$-IND-ATK security in the above sense. Moreover, the generic reduction from [10] shows that an adversary $\mathcal{A}$ that (ϵ, t)-breaks the (μ, q)-IND-ATK security of public-key encryption scheme PKE implies an adversary $\mathcal{A}'$ that (ϵ', t')-breaks the $(1, 1)$-IND-ATK security of PKE with $t' \approx t$ and $\epsilon' \geq \epsilon/(q\mu)$. Thus, the generic reduction loses a factor of $q\mu$.

5.1 Generic Construction

The construction of the public-key encryption scheme $\mathsf{Enc_{CCA}}$ with tight security reduction follows the Naor-Yung paradigm [48, 51, 46]. It uses as building blocks an (IND-CPA secure) public-key encryption scheme $\mathsf{Enc_{CPA}} = (\mathsf{CPA.Param}, \mathsf{CPA.Gen}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ and a (simulation-sound) non-interactive zero-knowledge proof system $\mathsf{NIZK} = (\mathsf{NIZK.Gen}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vfy})$.

We define scheme $\mathsf{Enc_{CCA}} = (\mathsf{CCA.Param}, \mathsf{CCA.Gen}, \mathsf{CCA.Enc}, \mathsf{CCA.Dec})$ as follows.

$\mathsf{CCA.Param}(\kappa)$ generates a common reference string for the NIZK proof system $crs \xleftarrow{\$} \mathsf{NIZK.Gen}(\kappa)$ for the language

$$\mathcal{L} := \{(pk_0, pk_1, c_0, c_1)\}$$

such that $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$ if and only if

$$c_0 = \mathsf{CPA.Enc}(pk_0, m) \wedge c_1 = \mathsf{CPA.Enc}(pk_1, m).$$

That is, we have $(pk_0, pk_1, vk_{\mathsf{ots}}, c_0, c_1, c_\Pi) \in \mathcal{L}$ iff c_0 and c_1 encrypt the same message m.

$\mathsf{CCA.Gen}(\Pi)$ generates two key pairs $(pk_0, sk_0), (pk_1, sk_1) \xleftarrow{\$} \mathsf{CPA.Gen}$ of the public-key encryption scheme. The resulting public key is $pk = (pk_0, pk_1, \Pi)$, the secret key is $sk = sk_0$.

$\mathsf{CCA.Enc}(pk, m)$ encrypts a message m by computing $c_0 = \mathsf{CPA.Enc}(pk_0, m)$, $c_1 = \mathsf{CPA.Enc}(pk_1, m)$, and a proof π that $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$, using the encryption randomness of c_0 and c_1 as witness. The resulting ciphertext is

$$c = (c_0, c_1, \pi)$$

$\mathsf{CCA.Dec}(sk, c)$ decrypts a given ciphertext as follows. First it checks whether $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$ by verifying the proof π. If false, then it returns $\perp$. Otherwise it computes and returns $m = \mathsf{CPA.Dec}(sk_0, c_0)$.

It is a classical result [51] that the above encryption scheme is $(1,1)$-IND-CCA secure, if $\mathsf{Enc_{CPA}}$ is $(1,1)$-IND-CPA secure and NIZK is one-time simulation sound. In the sequel we generalize this to showing that the (μ, q)-IND-CCA security of $\mathsf{Enc_{CCA}}$ reduces *tightly* (i.e., independent of μ and q) to the (μ, q)-IND-CPA security of $\mathsf{Enc_{CPA}}$ and the μq-security of NIZK.

We remark that our NIZK proof system from Section 4 inherits a certain form of (witness-)extractability from GS proofs. (See also the comment at the end of Section 4.3.) Hence, one could think of treating the NIZK system NIZK as one instance of an IND-CPA secure PKE scheme (with the extraction trapdoor as decryption key). It would seem natural to expect that a variant of scheme $\mathsf{Enc_{CCA}}$ above with only one $\mathsf{Enc_{CPA}}$ instance might be IND-CCA secure. We do not know if this holds, however: concretely, to show witness-indistinguishability of our proof system NIZK, we will at some point need to switch NIZK into hiding mode. In this mode, no extraction trapdoor exists, and it is unclear how to go answer decryption queries for $\mathsf{Enc_{CCA}}$.

Theorem 5. *Let* $\mathsf{Enc_{CPA}}$ *be* $(\epsilon_{\mathsf{CPA}}, t_{\mathsf{CPA}}, \mu, q)$*-IND-CPA secure, and let* NIZK *be* $(\epsilon_{\mathsf{ZK}}, \epsilon_{\mathsf{snd}}, \epsilon_{\mathsf{simsnd}}, t_{\mathsf{NIZK}}, \mu q)$*-secure. Then* $\mathsf{Enc_{CCA}}$ *is* (ϵ, t, μ, q)*-IND-CCA secure, where* t_{NIZK} *and* t_{CPA} *are roughly the runtime of the IND-CCA experiment with an adversary of runtime* t*, and*

$$\epsilon \leq 2 \cdot (\epsilon_{\mathsf{CPA}} + \epsilon_{\mathsf{ZK}}) + \epsilon_{\mathsf{snd}} + \epsilon_{\mathsf{simsnd}}.$$

Proof. The proof structure follows [51]. We proceed in a *sequence of games* [6, 53]. Let Win_i denote the probability that $\mathcal{A}$ wins in Game i.

Game 0. This is the (μ, q)-IND-CCA security experiment from Definition 5, executed with $b = 0$. Thus, the challenger always returns encryptions of m_0.

Game 1. This game is identical to Game 0, except that we change the way the NIZK proof is generated. Instead of using the real proving algorithm, we compute all proofs π using the zero-knowledge simulator of NIZK. Due to the zero-knowledge property of NIZK, we have

$$\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0] \leq \epsilon_{\mathsf{ZK}}.$$

Game 2. This game is identical to Game 1, except that we change the way challenge ciphertexts are created. For each ciphertext $c = (c_0, c_1, \pi)$, created by the challenger under public key $pk^{(j)}$ for some $j \in [\mu]$, the challenger computes $c_0 = \mathsf{CPA.Enc}(pk_0^{(j)}, m_0)$ and π as before, but computes $c_1 = \mathsf{CPA.Enc}(pk_1^{(j)}, m_1)$ as an encryption of m_1. Due to the $(\epsilon_{\mathsf{CPA}}, t_{\mathsf{CPA}}, \mu, q)$-IND-CPA security of $\mathsf{Enc_{CPA}}$, we have

$$\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1] \leq \epsilon_{\mathsf{CPA}}.$$

Game 3. This game is identical to Game 2, except that now the key pairs $(pk^{(j)}, sk^{(j)})$, $j \in [\mu]$, are computed and used differently. In the previous game, the challenger computes $(pk_0^{(j)}, sk_0^{(j)}) \xleftarrow{\$} \mathsf{CPA.Gen}$, $(pk_1^{(j)}, sk_1^{(j)}) \xleftarrow{\$} \mathsf{CPA.Gen}$, and sets $pk^{(j)} = (pk_0^{(j)}, pk_1^{(j)})$ and $sk^{(j)} = sk_0^{(j)}$. In this game, the challenger sets $pk^{(j)} = (pk_0^{(j)}, pk_1^{(j)})$ as before, but $sk^{(j)} = sk_1^{(j)}$. Moreover, for a given ciphertext $c = (c_0, c_1, \pi)$ the challenger now first checks the proof π and, if the proof verifies correctly, returns the decryption $m = \mathsf{CPA.Dec}(sk_1^{(j)}, c_1)$ of c_1. Key $sk_0^{(j)}$ is never used throughout the game.

Note that an adversary cannot distinguish Game 3 from Game 2, unless it submits a ciphertext $\tilde{c} = (\tilde{c}_0, \tilde{c}_1, \tilde{\pi})$ such that proof $\tilde{\pi}$ verifies, but $\mathsf{CPA.Dec}(sk_0^{(j)}, \tilde{c}_0) \neq \mathsf{CPA.Dec}(sk_1^{(j)}, \tilde{c}_1)$. Thus, the adversary has to provide a proof $\tilde{\pi}$ for a false statement. Note that the adversary sees at most μq simulated proofs throughout the game, thus, due to the μq-simulation-soundness of NIZK, we have

$$\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2] \leq \epsilon_{\mathsf{simsnd}}.$$

Game 4. Again we change the way challenge ciphertexts are created. For each ciphertext $c = (c_0, c_1, \pi)$, created by the challenger under $pk^{(j)}$ for some $j \in [\mu]$, the challenger computes $c_1 = \mathsf{CPA.Enc}(pk_1^{(j)}, m_1)$ and π as before, but now also each $c_0 = \mathsf{CPA.Enc}(pk_0^{(j)}, m_1)$ will be an encryption of m_1. Due to the $(\epsilon_{\mathsf{CPA}}, t_{\mathsf{CPA}}, \mu, q)$-IND-CPA security of $\mathsf{Enc}_{\mathsf{CPA}}$, we have

$$\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3] \leq \epsilon_{\mathsf{CPA}}.$$

Game 5. This game is identical to Game 4, but now the challenger uses the real proving algorithm for NIZK, instead of using simulated proofs. Due to the zero-knowledge property of NIZK, again we have

$$\Pr[\mathsf{Win}_5] - \Pr[\mathsf{Win}_4] \leq \epsilon_{\mathsf{ZK}}.$$

Game 6. This game is identical to Game 5, except that now we return to using $sk_0^{(j)}$ for encryption. Again an adversary cannot distinguish Game 6 from Game 5, unless it provides a proof $\tilde{\pi}$ for a false statement. Note that the adversary does not see any simulated proofs in this game, thus due to the soundness of NIZK we have

$$\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5] \leq \epsilon_{\mathsf{snd}}.$$

Note that Game 6 is identical to the (μ, q)-IND-CCA security experiment from Definition 5, executed with $b = 1$, and we have

$$\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_6] \leq 2 \cdot (\epsilon_{\mathsf{CPA}} + \epsilon_{\mathsf{ZK}}) + \epsilon_{\mathsf{snd}} + \epsilon_{\mathsf{simsnd}}.$$

□

5.2 Public-Key Encryption with Tight IND-CPA Security from DLIN

In order to instantiate the CCA-secure scheme from the previous section, we finally need an IND-CPA secure encryption scheme with tight security reduction. In this section we describe an adequate example.

Random self-reducibility of DLIN. The following lemma extends [10, Lemma 5.2] from DDH to the DLIN assumption.

Lemma 4. *$g, h, k \in \mathbb{G}$ be generators. There exists a probabilistic algorithm $\mathcal{R}$ with the following properties. $\mathcal{R}$ takes as input p, generators $g, h, k \in \mathbb{G}$, a vector $U \in \mathbb{G}^3$, and a flag $\tau \in \{0, 1\}$, and outputs $(h', k') \in \mathbb{G}^2$ and $U' \in \mathbb{G}^3$. The running time of algorithm $\mathcal{R}$ is dominated by at most 9 exponentiations in $\mathbb{G}$. The distribution of the output (h', k', U') depends on τ and whether $U \in \mathsf{DLIN}(g, h, k)$ or not, according to the following table.*

	$\tau = 0$	$\tau = 1$
$U \in \mathsf{DLIN}(g, h, k)$	$h' = h, k' = k$ $U' \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$	$h', k' \xleftarrow{\$} \mathbb{G} \setminus \{1\}$ $U' \xleftarrow{\$} \mathsf{DLIN}(g, h', k')$
$U \notin \mathsf{DLIN}(g, h, k)$	$h' = h, k' = k$ $U' \xleftarrow{\$} \mathbb{G}^3$	$h', k' \xleftarrow{\$} \mathbb{G} \setminus \{1\}$ $U' \xleftarrow{\$} \mathbb{G}^3$

We note that a slight variant of the above Lemma was given in [44, Lemma 7]. Lemma 7 of [44] considers an algorithm $\mathcal{R}$ that outputs (g', h', k', U') with $g', h', k' \xleftarrow{\$} \mathbb{G} \setminus \{1\}$, instead of (h', k', U') as above, in the more general K-linear setting, but only with $\tau = 1$.

Proof. Consider the following algorithm $\mathcal{R} = (\mathcal{R}_0, \mathcal{R}_1)$, which consists of two sub-algorithms $\mathcal{R}_0$ and $\mathcal{R}_1$. Given (p, g, h, k, U, τ), algorithm $\mathcal{R}$ runs $\mathcal{R}_\tau(p, g, h, k, U)$ and returns whatever $\mathcal{R}_\tau$ returns. In the sequel let us write $U = (g^u, h^v, k^w)$.

Algorithm $\mathcal{R}_0$. Algorithm $\mathcal{R}_0$ chooses integers $r_0, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $U' = (u'_0, u'_1, u'_2)$ as

$$\begin{aligned} u'_0 &= (g^u)^{r_0} \cdot g^{r_1} \\ u'_1 &= (h^v)^{r_0} \cdot h^{r_2} \\ u'_2 &= (k^w)^{r_0} \cdot k^{r_1} \cdot k^{r_2}. \end{aligned}$$

Then it returns (h, k, U').

To show that $\mathcal{R}_0$ produces the correct output distribution, let us first consider the case $U \in \mathsf{DLIN}(g, h, k)$, where we have $w = u + v$. Note that $u'_0 = g^{ur_0 + r_1}$ and $u'_1 = h^{vr_0 + r_2}$ are distributed uniformly and independently over $\mathbb{G}$, since r_1, r_2 are uniform and independent. In this case we have $u'_2 = k^{wr_0 + r_1 + r_2} = k^{(ur_0 + r_1) + (vr_0 + r_2)}$, thus it holds that U' is distributed uniformly over $\mathsf{DLIN}(g, h, k)$.

Now let us consider the case $U \notin \mathsf{DLIN}(g, h, k)$, which is equivalent to $w \neq u + v$. Taking logarithms to base g, we have

$$\begin{pmatrix} \log u'_0 \\ \log u'_1 \\ \log u'_2 \end{pmatrix} = \begin{pmatrix} u & 1 & 0 \\ v \log h & 0 & \log h \\ w \log k & \log k & \log k \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \\ r_2 \end{pmatrix} \tag{7}$$

The 3×3 matrix is invertible, because its determinant is equal to $(w - u - v) \cdot \log h \cdot \log k$, and we have $w \neq u + v$ and $\log h \neq 0 \neq \log k$ since both h and k are generators. Thus, for each (r_0, r_1, r_2) there exists a *unique* vector (u'_0, u'_1, u'_2) that satisfies the system of equations given by (7). Since (r_0, r_1, r_2) is distributed uniformly over $\mathbb{Z}_p^3$, vector $U' = (u'_0, u'_1, u'_2)$ is distributed uniformly over $\mathbb{G}^3$.

Algorithm $\mathcal{R}_1$. Algorithm $\mathcal{R}_1$ samples two random integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $h' = h^\alpha$, $k' = k^\beta$, and $U'' = (g^u, (h^v)^\alpha, (k^w)^\beta)$. Then it runs $\mathcal{R}_0(p, g, h', k', U'')$, and returns whatever $\mathcal{R}_0$ returns.

Note that h' and k' are random generators of $\mathbb{G}$, and that we have $U'' \in \mathsf{DLIN}(g, h', k')$ if and only if $U \in \mathsf{DLIN}(g, h, k)$. Thus, correctness of $\mathcal{R}_1$ follows from the correctness of $\mathcal{R}_0$. □

Encryption scheme. Lemma 4 gives rise to a folklore "DLIN-based ElGamal" scheme with tight security reduction in the multi-user setting. Let $\mathsf{Enc}_{\mathsf{CPA}} = (\mathsf{CPA.Gen}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ be the following encryption scheme.

$\mathsf{CPA.Gen}(\Pi)$ takes as input $\Pi = (\mathbb{G}, g)$ where $\mathbb{G}$ is with (description of a) group of prime order p and g is a generator of $\mathbb{G}$. It samples $\hat{h}, \hat{k} \xleftarrow{\$} \mathbb{Z}_p$ and sets $h = g^{\hat{h}}$ and $k = h^{\hat{k}}$. The public key is $pk = (g, h, k)$, the secret key is $sk = (\hat{h}, \hat{k})$.

$\mathsf{CPA.Enc}(pk, m)$ encrypts a message $m \in \mathbb{G}$ by sampling $r_g, r_h \xleftarrow{\$} \mathbb{Z}_p$ and computing $c_0 = g^{r_g}$, $c_1 = h^{r_h}$, $c_2 = m \cdot k^{r_g + r_h}$. The resulting ciphertext is $c = (c_0, c_1, c_2)$.

$\mathsf{CPA.Dec}(sk, c)$ decrypts a given ciphertext by computing $m = c_2 \cdot c_0^{-r_g} \cdot c_1^{-r_h}$.

Theorem 6. *Suppose there exists an adversary $\mathcal{A}$ that (ϵ, t)-breaks the (μ, q)-IND-CPA security of $\mathsf{Enc}_{\mathsf{CPA}}$. Then there exists an adversary $\mathcal{B}$ that (ϵ', t')-breaks the DLIN assumption in $\mathbb{G}$ with $t' \approx t$ and $\epsilon' \geq \epsilon - \mu/p$.*

Note that the success probability of $\mathcal{B}$ contains an *additive* term μ/p, which however is statistically small. It seems that a similar term has been overlooked in [10, Lemma 5.2].

Proof sketch. The proof is based on the random self-reducibility of the DLIN assumption, and nearly identical to the proof of [10, Theorem 5.3]. Therefore we only sketch it here.

The DLIN distinguisher $\mathcal{B}$ receives as input a DLIN challenge (g, h, k, U) where $U \xleftarrow{\$} \mathsf{DLIN}(g, h, k)$ or $U \xleftarrow{\$} \mathbb{G}^3$. It runs $\mathcal{A}$ as a subroutine by implementing the (μ, q)-IND-CPA challenger for $\mathcal{A}$.

- At the beginning of the game, $\mathcal{B}$ tosses a coin $b \xleftarrow{\$} \{0, 1\}$ and runs algorithm $\mathcal{R}$ from Lemma 4 μ times on input $\tau = 1$ and (g, h, k, U), so that it obtains μ vectors $(h^{(i)}, k^{(i)}, U^{(i)})_{i \in [\mu]}$. It defines $pk^{(i)} = (g, h^{(i)}, k^{(i)})$ for $i \in [\mu]$ and provides $\mathcal{A}$ with $(pk^{(i)})_{i \in \mu}$.
- When the adversary issues the j-th encryption query $(pk^{(i)}, m_{j,0}^{(i)}, m_{j,1}^{(i)})$ under $pk^{(i)}$, $\mathcal{B}$ runs $\mathcal{R}$ on input $\tau = 0$ and $(g, h^{(i)}, k^{(i)}, U^{(i)})$, to obtain a vector $U_j^{(i)} = (u_j^{(i)}, v_j^{(i)}, w_j^{(i)})$. Then $\mathcal{B}$ sets

$$c_j^{(i)} := (u_j^{(i)}, v_j^{(i)}, m_{j,b}^{(i)} \cdot w_j^{(i)})$$

  and returns the ciphertext $c_j^{(i)}$.

Eventually $\mathcal{A}$ outputs a guess b'. If $b = b'$ then $\mathcal{B}$ outputs 1, otherwise it outputs 0.

Analysis. Suppose that $U \in \mathsf{DLIN}(g,h,k)$. In this case, by the correctness of $\mathcal{R}$, all $pk^{(i)}$ are correctly distributed, and all public keys, and all ciphertexts are valid encryptions of message $m_{j,b}^{(i)}$. In this case $\mathcal{B}$ outputs 1 with probability $1/2+\epsilon$.

If $U \notin \mathsf{DLIN}(g,h,k)$, then U_i is uniformly distributed over $\mathbb{G}^3$, and thus we have $U_i \notin \mathsf{DLIN}(g,h,k)$ for all $i \in [\mu]$ except with probability μ/p. In this case each vector $U_j^{(i)} = (u_j^{(i)}, v_j^{(i)}, w_j^{(i)})$ is uniform over $\mathbb{G}^3$, and thus contains (information-theoretically) no information about b. In this case $\mathcal{B}$ outputs 1 with probability $1/2+\mu/p$.

6 Tight EUF-CMA Security in the Multi-User Setting

In Sections 3.3 and 3.4 we have described signature schemes with tight security proof in the *single-user* setting. In this section we would like to note, that these signature schemes also have tight security proofs in the *multi-user* setting.

EUF-CMA Security in the Multi-User Setting. Let us first describe the (μ,q)-*existential unforgeability under adaptive chosen-message attacks* ((μ,q)-EUF-CMA) experiment. This experiment is a simple adoption of the classical EUF-CMA experiment from [32] (see also Section 2.1) to the multi-user setting. Consider the following game, played between a challenger and a forger $\mathcal{A}$.

1. The forger, on input (μ,q) and public parameters Π, may ask a *non-adaptive* chosen-message query. To this end it submits, for each index $i \in [\mu]$, a list of messages $M^{(i,1)},\ldots,M^{(i,q_i)}$ to the challenger.

2. The challenger runs the key generation algorithm $\mathsf{Sig.Gen}(\Pi)$ μ times to generate keypairs $(vk_1,sk_1),\ldots,(vk_\mu,sk_\mu)$. The forger receives the list $(vk_1,\ldots,vk_\mu)$ of verification keys as input, as well as a signature $\sigma^{(i,j)}$ for each chosen message $M^{(i,j)}$, $(i,j) \in [\mu] \times [q_i]$.

3. Now the forger may ask *adaptive* chosen-message queries. Each query consists of a tuple $(i, M^{(i,j)})$, where $i \in [\mu]$ is an index and $M^{(i,j)}$ is a message, $j \in [q_i, q]$. The challenger returns a signature $\sigma^{(i,j)}$ under sk_i for $M^{(i,j)}$. Note that the forger may ask at most q signature queries per verification key, thus the forger asks at most μq signature queries in total.

4. Finally the forger outputs a triplet (i^*, M^*, σ^*).

Definition 6. An adversary is *adaptive*, if it asks at least one adaptive chosen-message query. Otherwise it is *non-adaptive*. Let $\mathcal{A}$ be an (adaptive or non-adaptive) adversary that runs in time t and outputs (i^*, M^*, σ^*). We say that $\mathcal{A}$ (ϵ,t)-breaks the (adaptive or non-adaptive) (μ,q)-EUF-CMA security of Sig if

$$\Pr[\mathsf{Sig.Vfy}(vk_{i^*}, M^*, \sigma^*) = 1 \wedge M^* \notin \{M^{(i^*,1)},\ldots,M^{(i^*,q)}\}] \geq \epsilon.$$

Tightly EUF-CMA Secure Signatures in the Multi-User Setting. One can prove that the signature schemes from Sections 3.3 and 3.4 are *tightly* secure under the DLIN assumption in the multi-user setting, too. The idea of these proofs is very simple, only a minor extension to the proofs in the single-user setting, therefore we only sketch it.

Let us first consider the scheme from Section 3.3. Recall that in the security proof of this scheme we constructed an algorithm $\mathcal{B}$, which takes as input a vector $((g,h,k),U) \in \mathbb{G}^3 \times \mathbb{G}^3$ and uses the

forger to decide whether $U \in \mathsf{DLIN}(g,h,k)$. In the multi-user setting, we follow exactly the same proof strategy, except for the following. Given a single DLIN-challenge $((g,h,k),U)$, we proceed as follows.

1. We first use the random self-reducibility of DLIN, by running the algorithm from Lemma 4 on input (g,h,k,U,τ) with $\tau = 0$, to generate μ vectors $U^{(1)},\ldots,U^{(\mu)}$, such that (with probability negligibly close to 1) for all $i \in [\mu]$ holds that

$$U^{(i)} \in \mathsf{DLIN}(g,h,k) \iff U \in \mathsf{DLIN}(g,h,k). \tag{8}$$

2. Then we run μ copies $\mathcal{B}^{(1)},\ldots,\mathcal{B}^{(\mu)}$ of algorithm $\mathcal{B}$ in parallel to simulate the non-adaptive (μ,q)-EUF-CMA security experiment. When the forger outputs its forgery (i^*,M^*,σ^*), then $\mathcal{B}^{(i^*)}$ will tell us whether $U^{(i^*)} \in \mathsf{DLIN}(g,h,k)$, which also allows us to decide whether $U \in \mathsf{DLIN}(g,h,k)$, due to (8).

This yields the tight non-adaptive (μ,q)-EUF-CMA security of the scheme from Section 3.3. The idea to prove that the scheme from Section 3.4 is *adaptively* (μ,q)-EUF-CMA secure is identical.

Acknowledgements. We would like to thank Masayuki Abe and Kristiyan Haralambiev for pointing out a missing argument in the proof of Lemma 1, and Georg Fuchsbauer for pointing out a mistake in Section 4.3.

References

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, August 2010.

[2] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org/`.

[3] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, August 2011.

[4] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. Cryptology ePrint Archive, Report 2012/285, 2012. `http://eprint.iacr.org/`.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[6] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[7] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 201–216. Springer, April 2007.

[8] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, October 1997.

[9] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, August 1998.

[10] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, May 2000.

[11] Daniel J. Bernstein. Proving tight security for Rabin-Williams signatures. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 70–87. Springer, April 2008.

[12] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.

[13] Dan Boneh, Ilya Mironov, and Victor Shoup. A secure signature scheme from bilinear maps. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 98–110. Springer, April 2003.

[14] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 351–368. Springer, April 2009.

[15] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

[16] Julien Cathalo, Benoît Libert, and Moti Yung. Group encryption: Non-interactive realization in the standard model. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 179–196. Springer, December 2009.

[17] Melissa Chase and Markulf Kohlweiss. A domain transformation for structure-preserving signatures on group elements. Cryptology ePrint Archive, Report 2011/342, 2011. `http://eprint.iacr.org/`.

[18] Benoît Chevallier-Mames and Marc Joye. A practical and tightly secure signature scheme without hash function. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 339–356. Springer, February 2007.

[19] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998.

[20] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002.

[21] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, December 2010.

[22] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[23] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[24] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[25] Georg Fuchsbauer. *Automorphic Signatures and Applications*. PhD thesis, ENS, Paris, 2010.

[26] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, August 1999.

[27] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, May 1999.

[28] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, May / June 2006.

[29] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 437–456. Springer, March 2009.

[30] Oded Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 104–110. Springer, August 1987.

[31] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[32] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[33] Matthew Green and Susan Hohenberger. Practical adaptive oblivious transfer from simple assumptions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 347–363. Springer, March 2011.

[34] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, December 2006.

[35] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

[36] Dennis Hofheinz. All-but-many lossy trapdoor functions. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 209–227. Springer, April 2012.

[37] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 590–607. Springer, August 2012.

[38] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, August 2007.

[39] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 313–332. Springer, April 2009.

[40] Marc Joye. An efficient on-line/off-line signature scheme without random oracles. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *CANS 08: 7th International Conference on Cryptology and Network Security*, volume 5339 of *Lecture Notes in Computer Science*, pages 98–107. Springer, December 2008.

[41] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03: 10th Conference on Computer and Communications Security*, pages 155–164. ACM Press, October 2003.

[42] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *ISOC Network and Distributed System Security Symposium – NDSS 2000*. The Internet Society, February 2000.

[43] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, August 2004.

[44] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09: 16th Conference on Computer and Communications Security*, pages 112–120. ACM Press, November 2009.

[45] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, February 2010.

[46] Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 241–254. Springer, May 2003.

[47] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, August 1990.

[48] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*. ACM Press, May 1990.

[49] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, August 1992.

[50] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, August 1992.

[51] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999.

[52] Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 189–206. Springer, May 2011.

[53] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[54] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, August 2009.

A Illustrations

A.1 Illustration of the Tree-based Signature Scheme

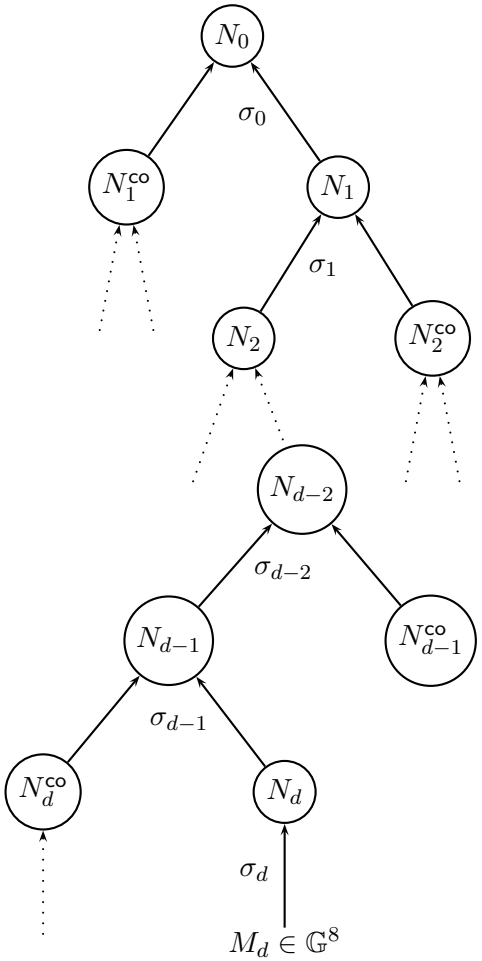

Figure 1: Illustration of the tree-based signature scheme. In this example we have $M_0 = (N_1^{\mathsf{co}}, N_1)$, $M_1 = (N_2, N_2^{\mathsf{co}})$, ..., $M_{d-1} = (N_d^{\mathsf{co}}, N_d)$ and $M_d = M$.

A.2 Illustration of the Definition of Sets $\mathcal{N}_{\mathsf{leaves}}$, $\mathcal{N}_{\mathsf{dir}}$, $\mathcal{N}_{\mathsf{out}}$

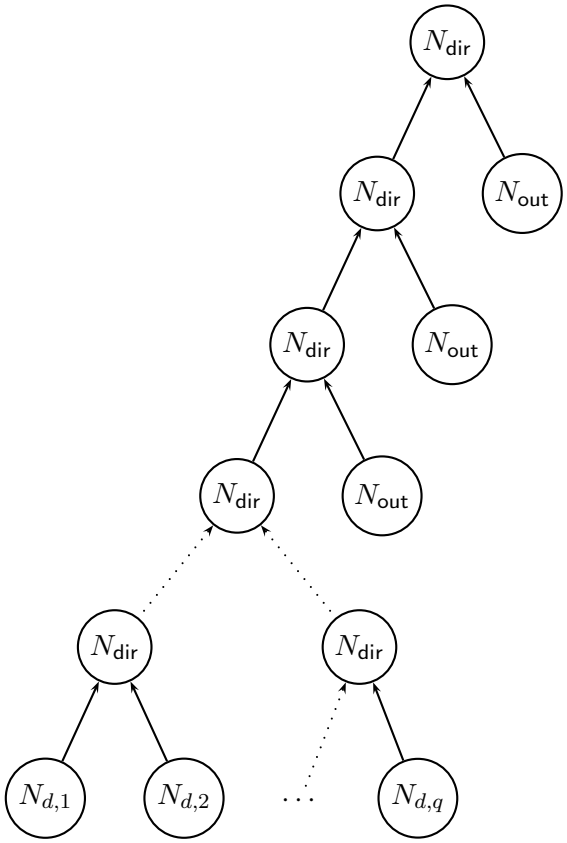

Figure 2: Illustration of the Definition of Sets $\mathcal{N}_{\mathsf{leaves}}$, $\mathcal{N}_{\mathsf{dir}}$, $\mathcal{N}_{\mathsf{out}}$.

We have

- $\mathcal{N}_{\mathsf{leaves}} = \{N_{d,1}, \ldots, N_{d,q}\}$.
- Each node N_{dir} is a direct ancestor of a node in $\mathcal{N}_{\mathsf{leaves}}$, therefore we have $N_{\mathsf{dir}} \in \mathcal{N}_{\mathsf{dir}}$ for all N_{dir}.
- Each node N_{out} is a sibling of a node N_{dir}, but not an ancestor of any node in $\mathcal{N}_{\mathsf{leaves}}$, thus we have $N_{\mathsf{out}} \in \mathcal{N}_{\mathsf{out}}$ for all N_{out}.