Tightly-Secure Signatures from Chameleon Hash Functions

Olivier Blazy[1], Saqib A. Kakvi[2], Eike Kiltz[2], and Jiaxin Pan[2]

[1] Université de Limoges, XLim, France
olivier.blazy@unilim.fr
[2] Horst Görtz Institute for IT-Security and Faculty of Mathematics,
Ruhr-University Bochum, Germany
{saqib.kakvi,eike.kiltz,jiaxin.pan}@rub.de

Abstract. We give a new framework for obtaining signatures with a tight security reduction from standard hardness assumptions. Concretely, we show that any Chameleon Hash function can be transformed into a (binary) tree-based signature scheme with tight security. The transformation is in the standard model, i.e., it does not make use of any random oracle. For specific assumptions (such as RSA, Diffie-Hellman and Short Integer Solution (SIS)) we further manage to obtain a more efficient flat-tree construction. Our framework explains and generalizes most of the existing schemes as well as providing a generic means for constructing tight signature schemes based on arbitrary assumptions, which improves the standard Merkle tree transformation. Moreover, we obtain the first tightly secure signature scheme from the SIS assumption and several schemes based on Diffie-Hellman in the standard model.

Some of our signature schemes can (using known techniques) be combined with Groth-Sahai proof methodology to yield tightly secure and efficient simulation-sound NIZK proofs of knowledge and CCA-secure encryption in the multi-user/-challenge setting under classical assumptions.

Keywords: signature, standard model, tight reduction, chameleon hash

1 Introduction

Digital Signatures are one of the most fundamental cryptographic primitives. They are used as a building block in numerous high-level cryptographic protocols. Their security is commonly proven in terms of a security reduction showing that any successful adversary $\mathcal{A}$ attacking the scheme can be transformed into a successful adversary $\mathcal{B}$ breaking the underlying hard intractability assumption. Naturally, we would desire that $\mathcal{B}$'s success $\varepsilon_{\mathcal{B}}$ is approximately the same as $\mathcal{A}$'s success $\varepsilon_{\mathcal{A}}$ in attacking the system and also the running times of $\mathcal{A}$ and $\mathcal{B}$ are approximately the same. Such a scheme is said to have a tight security reduction and does not require to compensate reduction's security loss with increased parameters.

Signature schemes with a tight security reduction are known based on standard intractability assumptions such as the RSA [6] or the (bilinear) Computational Diffie-Hellman (CDH) assumption [25]. However, their security can only be proven in the random oracle model [5] with all its limitations (e.g., [14,21]).

Standard Model Signatures. We now discuss signature schemes in the standard model (i.e., without using random oracles). On the one hand, there exist signature schemes with a tight security reduction (e.g., [18,44]) but they usually rely on specific relatively strong "q assumptions," such as the Strong (or, Flexible) RSA assumption [20] and the q-Diffie-Hellman Inversion Assumption (q-CDHI) [10].[3] On the other hand, known signature schemes from "standard assumptions" (i.e., general assumptions such as the one-wayness of trapdoor permutations [43,27,22] or more specific assumptions such as the RSA assumption [32,31] , the CDH assumption [45], or the Short Integer Solution (SIS) assumption [15,38]) have non-tight security reductions, meaning their security reduction loses a multiplicative factor of q, which is the maximal number of established signatures. Since q can be as large as 2^{40}, this security loss can have a dramatic impact on the scheme's parameters.

To the best of our knowledge, there are only a few exceptions to the above. The flat d-ary tree-based signature scheme by Cramer and Damgård [19] from almost two decades ago is based on a standard assumption (the RSA assumption) and (even though not explicitly mentioned in [19]) the security reduction is tight. In follow-up papers [16] and [13] extend their methods to an assumption related to factoring and CDH, respectively. Hofheinz and Jager [30] proposed a binary tree-based construction from the Linear (LIN) assumption. More recently, works on identity-based encryption [9,17] imply tight signatures from LIN.

1.1 Our contributions

Overview. In this work we revisit the question of construction standard-model signatures with a tight security reduction. Our main result shows that, surprisingly, *tightly-secure* signatures can be constructed from any Chameleon Hash function CHF. Our transformation is based on binary trees and hence a signature contains λ Chameleon Hashes plus λ elements from the randomness space of CHF, where λ is the security parameter. As tightly secure Chameleon Hash functions exist from generic primitives such as claw-free permutations (CFP) [35], Σ protocols [4] and specific assumptions such as the RSA [32], the factoring (FAC) [35], and the discrete logarithm (DLOG) [35] assumptions, we immediately obtain a number of new signature schemes with tight security reductions. We improve the well-known Merkle tree construction [37] and its variant [41] in the sense that our signature size is the same as the original Merkle tree construction, but our security loss is independent of the number of signing queries.

[3] In q-assumptions an adversary is provided with q (polynomially many) random "solved instances" and has to compute a new solved instance. Examples are the strong RSA and the q-Diffie-Hellman Inversion assumptions. Both are considerably stronger than their "non-q" counterparts.

In fact, our transformation can be generalized to flat-tree signatures with improved efficiency. From a general primitive called d-time two-tier signatures TTSig (a generalization of two-tier signatures [7] to any $d \geq 1$), we build flat d-ary (instead of binary, 2-ary) trees via our second transformation d-Tree, such that a signature only contains $O(\lambda/\log d)$ many elements. Whereas Chameleon Hash functions only imply *one-time* two-tier signatures, for specific assumptions such as RSA, CDH and SIS we are able to construct efficient d-time two-tier signatures, hence also d-ary tree signatures. Our reduction loses a factor of d which is still (almost) tight as d is generally assumed to be small and, in particular, is independent of the number of signing queries. See Figure 1 for a schematic overview of our transformations.

We stress that while all our schemes are only secure in a non-adaptive sense (a.k.a. weak security), they can be transformed into adaptively secure signature schemes using a Chameleon Hash or a one-time signature, without losing efficiency or tightness (such transformations have been used several times [35,11,33,40]).

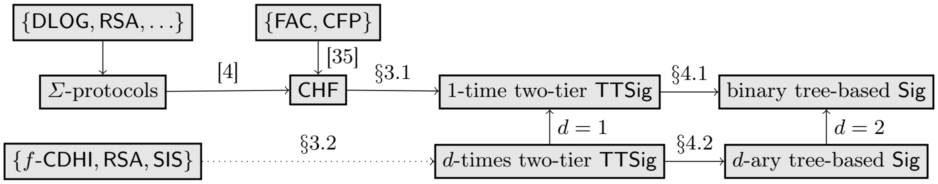

Fig. 1. Schematic overview of our constructions from the assumption level (left) over two-tier signatures (middle) to signatures (right). All implications have a tight security reduction, except the dotted line which loses a factor of d.

Interestingly, our framework also offers a theoretical explanation of almost all known tightly secure signature schemes. Our d-ary transformation d-Tree instantiated with an RSA-based d-times two-tier signature essentially equals the scheme by Cramer and Damgård [19]. The scheme by Hofheinz and Jager [30] can be obtained by using a Chameleon Hash function based on the LIN assumption (which is given in the full version [8]), it can in fact be generalized by building a chameleon hash based on any of the matrix assumptions from [23]. The CDH-based signature scheme from [13] is a less efficient version of our construction from the f-CDHI assumption with the parameters $f = 1$. Table 1 gives an overview over all known tightly secure signature schemes from standard assumptions. Some of our schemes are also (almost) structure preserving, a property with important applications, which we will discuss later.

Details. First, we transform a Chameleon Hash into a two-tier signature and then, we show how to transform the latter into a binary tree-based signature scheme.

The concept of d-time two-tier signatures is a natural generalization of (one-time, $d = 1$) two-tier signatures introduced by Bellare and Shoup [7]. A two-tier signature scheme is like a standard signature scheme except that the public (secret) key is split into fixed primary part ppk (psk) and a variable secondary part spk (ssk). In terms of security we require that an adversary possessing the

Scheme	Origin	Assumption	$\lvert\mathsf{pk}\rvert$	Signature size	Loss	Structure
						preserving
$\mathsf{BinTree}+\mathsf{CHF}_{\mathsf{DLOG}}$	new	DLOG	$O(1) \times \mathbb{G}$	$O(\lambda) \times (\mathbb{G} + \mathbb{Z}_p)$	$O(1)$	almost
$\mathsf{BinTree}+\mathsf{CHF}_{\mathsf{FAC}}$	new	FAC	$O(1) \times \mathbb{Z}_N$	$O(\lambda) \times \mathbb{Z}_N$	$O(1)$	$--$
$\mathsf{BinTree}+\mathsf{CHF}_{\mathsf{RSA}}$	new	RSA	$O(1) \times \mathbb{Z}_N$	$O(\lambda) \times \mathbb{Z}_N$	$O(1)$	$--$
$\mathsf{BinTree}+\mathsf{CHF}_{\mathsf{LIN}}$	[30]	LIN	$O(1) \times \mathbb{G}$	$O(\lambda) \times (\mathbb{G} + \mathbb{Z}_p)$	$O(1)$	$\surd$
d-$\mathsf{Tree}+\mathsf{TTSig}_{f\text{-}\mathsf{CDHI}}$	new	f-CDHI	$O(d/f) \times \mathbb{G}$	$O(\lambda/\log(d)) \times (\mathbb{G} + \mathbb{Z}_p)$	$O(d)$	almost
d-$\mathsf{Tree}+\mathsf{TTSig}_{\mathsf{RSA}}$	[19]	RSA	$O(d) \times \mathbb{Z}_N$	$O(\lambda/\log(d)) \times \mathbb{Z}_N$	$O(d)$	$--$
d-$\mathsf{Tree}+\mathsf{TTSig}_{\mathsf{SIS}}$	new	SIS	$O(d) \times \mathbb{Z}_p^{(\lambda \times \lambda \log(p))}$	$O(\lambda/\log(d)) \times \mathbb{Z}_p^{\lambda \log(p)}$	$O(d)$	$--$
BKP14	[9]	k-LIN	$O(\lambda k^2) \times \mathbb{G}$	$O(k) \times \mathbb{G}$	$O(\lambda)$	almost
CW13	[17]	k-LIN	$O(\lambda k^2) \times \mathbb{G}$	$O(k) \times \mathbb{G}$	$O(\lambda)$	almost

Table 1. Comparison between known tightly-secure signature schemes from standard (non-q) assumptions, where λ is the security parameter.

primary public key and having access to an oracle generating q independent secondary public keys, together with d signatures for each of oracle queries, cannot forge a fresh signature relative to one of the established public keys. The challenge will be to construct a d-time two-tier signature scheme with a tight (i.e., independent of q) security reduction from a standard assumption.

- Any Chameleon Hash implies a 1-time two-tier signature scheme. While it is well-known that a Chameleon Hash implies a (standard) 1-time signature [40], the novelty of our observation lies in the tight security reduction for two-tier signatures.
- We give constructions of d-time two-tier signatures for any $d \geq 2$ with a tight security reduction from a number of standard number theoretic assumptions such as the RSA, the SIS, the CDH and the f-CDHI[4] ($1 \leq f \leq d$) assumption. The important feature of our new constructions is the constant number of elements in the secondary public key while maintaining the tight reduction.
- We show that d-time two-tier signatures imply d-ary tree-based signatures with a tight security reduction. In our construction the verification/signing keys are the primary public/secret key of the d-times two-tier signature scheme. The signer implicitly maintains a d-ary authenticated tree of height $k = \lambda/\log(d)$, where λ is the security parameter. Each internal node is assigned a secondary public/secret key, the secret key is used to authenticate the key of the d distinct children via a signature. To sign a message, the signer picks the next unused leaf and outputs the authenticated path to the leaf plus a signature of the message under the leaf's secret key.

Applications. We remark that some of our tightly secure signature schemes are almost structure preserving (cf. Table 1) in the sense that they do not satisfy the structure preserving definition in [2], but, following a similar method as Hofheinz and Jager [30], these schemes can be used to build tightly-secure simulation-sounds NIZK and tightly-secure encryption in the multi-user/multi-challenge setting. A discussion of this can be found in Appendix B. We also note that our results can be used to improve the Key Agreement of Bader et al. [3].

[4] The f-CDHI assumption is a generalization of CDH and states that given $g, g^x, \ldots g^{x^f}$, it is hard to compute $g^{1/x}$. Note that f is small (constant) in our applications and does not depend on the number of signing queries.

Open Problems. Since our signature schemes contain $O(\lambda/\log d)$ many group elements, they cannot be considered to be practical. More recently, Blazy, Kiltz and Pan [9] and Chen and Wee [17] proposed tightly secure identity-based encryptions from the LIN assumption independently, which imply tightly secure signature schemes with constant signature size. However, it is not clear how to extend their methods to constructing tight signatures based on the RSA assumption or any lattice assumption. Thus, obtaining a tightly secure signature scheme from the standard RSA assumption or any lattice assumption whose signatures only contain a *constant number* of group elements remains an open problem.

2 Preliminaries

2.1 Notation

We denote our security parameter as λ. For all $n \in \mathbb{N}$, we denote by 1^n the n-bit string of all ones. For any element x in a set S, we use $x \in_R S$ to indicate that we choose x uniformly random in S. All algorithms may be randomized. For any algorithm A, we define $x \leftarrow_\$ A(a_1, \ldots, a_n)$ as the execution of A with inputs $a_1, \ldots, a_n$ and fresh randomness and then assigning the output to x.

A list of classical security definitions and assumptions (CDH, f-CDHI, SIS, RSA) that we require for our results can be found in Appendix A.

2.2 Signatures

We first recall the definition of a digital signature scheme.

Definition 1 (Signature scheme). *A digital signature scheme* Sig *with message space* $\mathcal{M}$ *is defined as a triple of probabilistic polynomial time (PPT) algorithms* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- Gen *takes as an input the unary representation of our security parameter* 1^λ *and outputs a signing key* sk *and verification key* pk*.*
- Sign *takes as input a signing key* sk*, message* m *and outputs a signature* σ*.*
- Verify *is a deterministic algorithm, which on input of a public key and a message-signature pair* (m, σ) *outputs 1 (accept) or 0 (reject).*

Sig *is perfectly correct if for any* $\lambda \in \mathbb{N}$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$*, all* $m \in \mathcal{M}$*, and all* $\sigma \leftarrow_\$ \mathsf{Sign}(\mathsf{sk}, m)$ *that* $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$*.*

Some of the signature schemes we present are stateful. This means that the signer maintains a state that is updated after each execution of the signing algorithm. Fortunately, our stateful schemes can be transformed to be stateless by using the technique from [26].

Definition 2 (Security of signatures). *Signature scheme* Sig *is* (t, ε, q)*-existential unforgeable under non-adaptive chosen-message attacks (*$\mathsf{EUF\text{-}NCMA}$*) iff*

$$\Pr[\mathsf{Exp}^{\mathsf{EUF\text{-}NCMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda) = 1] \leq \varepsilon$$

holds for any PPT adversary $\mathcal{F}$ with running time t, where $\mathsf{Exp}^{\mathsf{EUF\text{-}NCMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda)$ *is defined in Table 2. The existential unforgeability under chosen-message attacks is defined in the similar way.*

Experiment $\mathsf{Exp}^{\mathsf{EUF\text{-}NCMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda)$	**Experiment** $\mathsf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda)$
$\mathcal{Q} := (m_1, \ldots, m_q) \leftarrow_{\$} \mathcal{F}(1^\lambda)$;	$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$;
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$;	$(m^*, \sigma^*) \leftarrow_{\$} \mathcal{F}^{\mathtt{OSign}(\cdot)}(\mathsf{pk})$, where the oracle
$\sigma_i \leftarrow_{\$} \mathsf{Sign}(\mathsf{sk}, m_i)$ for $i = 1, \ldots, q$;	$\mathtt{OSign}(\cdot) := \mathsf{Sign}(\mathsf{sk}, \cdot)$
$(m^*, \sigma^*) \leftarrow_{\$} \mathcal{F}(\mathsf{pk}, \sigma_1, \ldots, \sigma_q)$;	If $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q} := \{m_1, \ldots, m_q\}$
If $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q}$	where m_i is the i-th query, then return 1;
then return 1, else return 0.	else return 0.

Table 2. EUF-NCMA and EUF-CMA experiments for the signature scheme.

We also consider a stronger security notion than EUF, namely strong unforgeability, SUF. In the strong unforgeability experiment, the adversary is allowed to forge a new signature on a message for which he has already seen a signature on. To accommodate this, we adjust our list $\mathcal{Q} := \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$. Furthermore for the valid forgery, we require $(m^*, \sigma^*) \notin \mathcal{Q}$. This stronger notion applies for both adaptive and non-adaptive definitions, which we refer to as SUF-CMA and SUF-NCMA respectively.

2.3 Two-Tier Signatures

We now present a generalization of two-tier signature schemes, due to Bellare and Shoup [7]. In a two-tier signature scheme, the key generation algorithm is split into two algorithms, the primary and secondary key generation algorithms. The primary key is static and used for all signatures. The secondary key is ephemeral and used for only one or many messages. To generate the signature, we need both a primary and secondary key. In the original definition [7], each secondary key was allowed to be used to sign exactly once. We generalize to allow each secondary key to be used to sign at most d messages. We refer to this generalization as the d-time two-tier signature, the constructions presented in [7] are 1-time two-tier signatures.

Definition 3 (d-time two-tier signature scheme). *A two-tier signature* TTSig *is defined as a quadruple of probabilistic algorithms* (PriGen, SecGen, TTSign, TTVerify)*:*

- $\mathsf{PriGen}(1^\lambda, d)$ *outputs a primary signing key* psk *and primary verification key* ppk.
- $\mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ *outputs a fresh secondary verification and signing key pair* $(\mathsf{spk}, \mathsf{ssk})$.
- $\mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}, m)$ *outputs a signature σ. We denote the stateful variant by* $\mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}, m; j)$ *where j is the state.*

- $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}, m, \sigma)$ *deterministically outputs 1 (accept) or 0 (reject). We denote the stateful variant by* $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}, m, \sigma; j)$ *where* j *is the state.*

Correctness is defined in a natural way as in Definition 1.

Definition 4 (Security of two-tier signatures). *A two-tier signature* TTSig *is* (t, q, d, ε)*-existential unforgeable under non-adaptively chosen-message attacks (*$\mathsf{TT\text{-}EUF\text{-}NCMA}$*) iff*

$$\Pr[\mathsf{Exp}^{\mathsf{TT\text{-}EUF\text{-}NCMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda, d) = 1] \le \varepsilon$$

holds for any PPT adversary $\mathcal{F}$ *with running time* t*, where* $\mathsf{Exp}^{\mathsf{TT\text{-}EUF\text{-}NCMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda, d)$ *is defined in Table 3. The existential unforgeability under (adaptively) chosen-message attacks (*$\mathsf{TT\text{-}EUF\text{-}CMA}$*) is defined in the similar way.*

Experiment $\mathsf{Exp}^{\mathsf{TT\text{-}EUF\text{-}NCMA}}_{\mathsf{TTSig},\mathcal{F},q}(\lambda, d)$ $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_\$ \mathsf{PriGen}(1^\lambda, d)$; $(m^*, \sigma^*, i^*) \leftarrow_\$ \mathcal{F}^{\mathtt{NTTSign}(\cdot)}(\mathsf{ppk})$; If $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}_{i^*}, m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q}_{i^*}$ then return 1, else return 0.	**Experiment** $\mathsf{Exp}^{\mathsf{TT\text{-}EUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{F},q}(\lambda, d)$ $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_\$ \mathsf{PriGen}(1^\lambda, d)$; $(m^*, \sigma^*, i^*) \leftarrow_\$ \mathcal{F}^{\mathtt{OSKey}(),\mathtt{TTSign}(\cdot,\cdot)}(\mathsf{ppk})$; If $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}_{i^*}, m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q}_{i^*}$ then return 1, else return 0.
Oracle $\mathtt{NTTSign}(m_1, \ldots, m_d)$ $i = i + 1$ and $(\mathsf{spk}_i, \mathsf{ssk}_i) \leftarrow_\$ \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$; $\sigma_j \leftarrow_\$ \mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}_i, m_j)$ for $j = 1, \ldots, d$; Store $(m_1, \ldots, m_d)$ in the list $\mathcal{Q}_i$; Return $(\mathsf{spk}_i, \sigma_1, \ldots, \sigma_d)$.	**Oracle** $\mathtt{OSKey}()$ $i = i + 1$ and $j_i = 0$; $(\mathsf{spk}_i, \mathsf{ssk}_i) \leftarrow_\$ \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$; Return spk_i.
	Oracle $\mathtt{TTSign}(i', m)$ $j_{i'} = j_{i'} + 1$; $m_{j_{i'}} := m$ If $j_{i'} > d$ or $(\mathsf{spk}_{i'}, \mathsf{ssk}_{i'})$ is undefined then return $\perp$; $\sigma \leftarrow_\$ \mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}_{i'}, m_{j_{i'}})$ and store $m_{j_{i'}}$ in $\mathcal{Q}_{i'}$; Return σ.

Table 3. $\mathsf{TT\text{-}EUF\text{-}NCMA}$ and $\mathsf{TT\text{-}EUF\text{-}CMA}$ experiments for the two-tier signature scheme.

We also define the strong unforgeability of two-tier signatures, in both the adaptive case, $\mathsf{TT\text{-}SUF\text{-}CMA}$, and the non-adaptive case, $\mathsf{TT\text{-}SUF\text{-}NCMA}$, analogously as to how we defined it for standard signatures.

2.4 Chameleon Hash Functions

A Chameleon Hash Function is defined as $\mathsf{CHF} = (\mathsf{CHGen}, \mathsf{CHash}, \mathsf{Coll})$:

- $\mathsf{CHGen}(1^\lambda)$ outputs the hash key chk and the trapdoor td.
- $\mathsf{CHash}(\mathsf{chk}, m, r)$ outputs the hash value h.
- $\mathsf{Coll}(\mathsf{td}, (m, r), \hat{m})$ outputs a randomness $\hat{r}$ such that $\mathsf{CHash}(\mathsf{chk}, m, r) = \mathsf{CHash}(\mathsf{chk}, \hat{m}, \hat{r})$.

The standard security notion for Chameleon Hashes is collision resistance (coll). Formally, CHF is (t, ε)-coll if for the adversary $\mathcal{A}$ running in time at most t we have:

$$\Pr\left[\begin{array}{c}(\mathsf{chk}, \mathsf{td}) \leftarrow_\$ \mathsf{CHGen}(1^\lambda); ((m_1, r_1), (m_2, r_2)) \leftarrow_\$ \mathcal{A}(\mathsf{chk}) \\ \wedge \mathsf{CHash}(\mathsf{chk}, m_1, r_1) = \mathsf{CHash}(\mathsf{chk}, m_2, r_2) \wedge (m_1, r_1) \neq (m_2, r_2)\end{array}\right] \le \varepsilon.$$

However, any user in possession of the trapdoor td is able to find a collision using Coll. Additionally, Chameleon Hash functions have the uniformity property, which means the hash value leaks nothing about the message input. Formally, for all pair of messages m_1 and m_2 and the randomly chosen r, the probability distributions of the random variables $\mathsf{CHash}(\mathsf{chk}, m_1, r)$ and $\mathsf{CHash}(\mathsf{chk}, m_2, r)$ are computationally indistinguishable.

3 Constructions of Two-Tier Signatures

In this section we show different constructions of d-time two-tier signatures for $d = 1$ (Section 3.1) and $d \geq 2$ (Section 3.2).

3.1 Construction from any Chameleon Hash function

We construct a non-adaptively strongly secure one-time two-tier signature $\mathsf{TTSig}_{\mathsf{CHF}} = (\mathsf{PriGen}, \mathsf{SecGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ from any Chameleon Hash $\mathsf{CHF} = (\mathsf{CHGen}, \mathsf{CHash}, \mathsf{Coll})$ with message space $\mathcal{M}$ and randomness space $\mathcal{R}$.

- $\mathsf{PriGen}(1^\lambda)$: Generate a Chameleon Hash key and the corresponding trapdoor $(\mathsf{chk}, \mathsf{td}) \leftarrow_\$ \mathsf{CHGen}(1^\lambda)$. Define $\mathsf{ppk} = \mathsf{chk}$ and $\mathsf{psk} = \mathsf{td}$.
- $\mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$: Pick random $\hat{\sigma} \in_R \mathcal{R}$ and compute $h = \mathsf{CHash}(\mathsf{ppk}, \hat{m}, \hat{\sigma})$, for an arbitrary public $\hat{m} \in \mathcal{M}$ (possibly $\hat{m} = 0$). Define $\mathsf{spk} = h$ and $\mathsf{ssk} = \hat{\sigma}$.
- $\mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}, m)$: The signer uses the trapdoor of the chameleon hash to compute a collision as $\sigma = \mathsf{Coll}(\mathsf{psk}, \hat{m}, \hat{\sigma}, m)$, which means $\mathsf{CHash}(\mathsf{ppk}, m, \sigma) = \mathsf{spk}$. The signature on m is $\sigma \in \mathcal{R}$.
- $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}, m, \sigma)$: Check if $\mathsf{CHash}(\mathsf{ppk}, m, \sigma) = \mathsf{spk}$.

Correctness of the scheme follows by correctness of the Chameleon Hash function.

Theorem 1. *If* CHF *is a* (t, ε)-coll *Chameleon Hash function, then for any* $q \in \mathbb{N}$*,* $\mathsf{TTSig}_{\mathsf{CHF}}$ *is a* $(t', q, 1, \varepsilon')$-$\mathsf{TT\text{-}SUF\text{-}NCMA}$ *signature where* $\varepsilon' = \varepsilon$ *and* $t' = t - O(q)$*.*

Proof. Let $\mathcal{F}$ be a PPT adversary that $(t', q, 1, \varepsilon')$-breaks the $\mathsf{TT\text{-}SUF\text{-}NCMA}$ security of $\mathsf{TTSig}_{\mathsf{CHF}}$. Then we construct an adversary $\mathcal{B}$ that (t, ε)-breaks the collision resistance of CHF. Formally, $\mathcal{B}$ is given the challenge Chameleon Hash key chk and asked to come up with two distinct inputs $(m, r) \neq (m', r')$ such that $\mathsf{CHash}(\mathsf{chk}, m, r) = \mathsf{CHash}(\mathsf{chk}, m', r')$.

Simulation. $\mathcal{B}$ simulates $\mathsf{PriGen}(1^\lambda)$ as follows: it sets $\mathsf{ppk} = \mathsf{chk}$ and returns ppk to $\mathcal{F}$. Now $\mathcal{B}$ does not have the Chameleon Hash trapdoor and psk is empty.

Upon receiving the ith message m_i from $\mathcal{F}$, $\mathcal{B}$ simulates $\mathtt{NTTSign}(m_i)$ as follows: it picks a random $\sigma_i \in_R \mathcal{R}$ and computes $h_i = \mathsf{CHash}(\mathsf{ppk}, m_i, \sigma_i)$. Define the secondary public key $\mathsf{spk}_i = h_i$ and return spk_i and the signature σ_i.

The simulation is identical to the real execution. Firstly, chk is from the Chameleon Hash challenge and, thus, the simulation of PriGen is identical to the definition. Secondly, in the original definition $\mathsf{spk}_i = \mathsf{CHash}(\mathsf{ppk}, 0, r_i)$, while $\mathsf{spk}_i = \mathsf{CHash}(\mathsf{ppk}, m_i, \sigma_i)$ in the simulation. These two distributions are identical based on the uniformity property of CHF. Thirdly, it is easy to see the simulated signatures are well-formed.

Extracting the collision. Once $\mathcal{F}$ outputs a forgery (m^*, σ^*, i^*), $\mathcal{B}$ aborts if spk_{i^*} is undefined. Otherwise, $\mathcal{B}$ checks if $\mathsf{CHash}(\mathsf{ppk}, m_{i^*}, \sigma_{i^*}) = \mathsf{spk}_{i^*} = \mathsf{CHash}(\mathsf{ppk}, m^*, \sigma^*)$. If that is the case, then $\mathcal{B}$ returns the collision $((m^*, \sigma^*), (m_{i^*}, \sigma_{i^*}))$. By the strong unforgeability of $\mathsf{TTSig}_{\mathsf{CHF}}$, $(m^*, \sigma^*) \neq (m_{i^*}, \sigma_{i^*})$. Thus, if $\mathcal{F}$ outputs a successful forgery then $\mathcal{B}$ finds a collision for the Chameleon Hash with probability $\varepsilon = \varepsilon'$. □

3.2 Direct Constructions of d-time Two-Tier Signatures

The construction from Section 3.1 can be extended to yield a d-time two-tier signature scheme for any $d \geq 1$ but the size of the secondary public-key is linear in d which is not useful for constructing efficient flat-tree signatures. In this section, we present stateful d-time two-tier signature schemes with *constant size* secondary key, from the f-CDHI, and SIS assumptions. Two more constructions from RSA and factoring are given in the full version [8].

Construction from f-CDHI The construction from this section has an additional parameter $1 \leq f \leq d$ which offers a trade-off between the size of ppk ($O(d/f)$ group elements) and the underlying hardness assumption f-CDHI relative to a pairing group generator algorithm $\mathsf{PGroupGen}$. (See Appendix A for a formal definition of f-CDHI.) We now present the stateful d-time two-tier signature scheme $\mathsf{TTSig}_{f\text{-}\mathsf{CDHI}} = (\mathsf{PriGen}, \mathsf{SecGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ from f-CDHI with message space $\mathbb{Z}_p$. For simplicity we assume there exists an integer c such that $c \cdot f = d$.

- $\mathsf{PriGen}(1^\lambda, d)$: generates a pairing group $\mathcal{PG} = (\mathbb{G}, g, p, \mathbb{G}_T, e) \leftarrow_\$ \mathsf{PGroupGen}(1^\lambda)$, picks random scalars $x_0, \dots, x_c \in_R \mathbb{Z}_p$ and computes $h_i = g^{x_i}$ for $i = 0 \dots, c$ and defines $\mathsf{psk} = (x_0, \dots, x_c), \mathsf{ppk} = (\mathcal{PG}, (h_0, \dots, h_c))$.
- $\mathsf{SecGen}(\mathsf{psk}, \mathsf{ppk})$: picks a random $u \in_R \mathbb{G}$, and defines $\mathsf{spk} = u$, the secondary signing key is empty.
- $\mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}, m_j; j)$: to sign the $j = (\alpha \cdot f + \beta)$-th message m_j ($j \in [\![1, d]\!], \alpha \in [\![0, c]\!], \beta \in [\![0, f-1]\!]$), compute $\sigma_j = (g^{m_j} u)^{1/(x_\alpha + \beta)}$.
- $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}, m_j, \sigma_j; j)$: parses $j = \alpha \cdot f + \beta$ and checks if $e(\sigma, h_\alpha \cdot g^\beta) = e(g^{m_j} \cdot u, g)$.

It is easy to verify correctness.

Theorem 2. *If the f-CDHI assumption is (t, ε)-hard, then for any $q \in \mathbb{N}$, $\mathsf{TTSig}_{d,f\text{-}\mathsf{CDHI}}$ is a (t', q, d, ε')-TT-EUF-NCMA signature scheme where $\varepsilon' = d\varepsilon$ and $t' = t - O(dq)$.*

We stress that f is a fixed small parameter of the scheme. In particular, as 1-CDHI is equivalent to CDH, $\mathsf{TTSig}_{1\text{-}\mathsf{CDHI}}$ is secure under the standard CDH assumption, which is equivalent to the scheme from [13].

Proof. Let $\mathcal{F}$ be an adversary that (t', q, d, ε')-breaks the $\mathsf{TT\text{-}EUF\text{-}NCMA}$ security of $\mathsf{TTSig}_{f\text{-}\mathsf{CDHI}}$. Then we construct an adversary $\mathcal{B}$ that (t, ε)-breaks the f-CDHI Assumption. Adversary $\mathcal{B}$ takes as input a pairing group description $\hat{\mathcal{PG}} = (\mathbb{G}, \mathbb{G}_T, \hat{g}, p, e)$ and a f-CDHI-challenge $(\hat{g}, \hat{g}^x, \ldots, \hat{g}^{x^f})$. Its goal is to compute $\hat{g}^{\frac{1}{x}}$.

- To simulate PriGen, $\mathcal{B}$ picks a random $j' \in_R \llbracket 1, d \rrbracket$, which defines uniquely α', β' as the quotient and modulo in the euclidean division of j' by f. $\mathcal{B}$ computes $g = \hat{g}^{\prod_{b \neq \beta'}(x+b-\beta')}$ $(b \in \llbracket 0, f-1 \rrbracket)$ from f-CDHI-challenge and chooses c random scalars $(x_0, \ldots, x_{\alpha'-1}, x_{\alpha'+1}, \ldots, x_c) \in_R \mathbb{Z}_p^c$, where $c = d/f$ as defined in the scheme, and for all $\alpha \in \llbracket 0, c \rrbracket$ computes:
$$h_\alpha = \begin{cases} g^{x-\beta'} & \text{if} \quad \alpha = \alpha' \quad (\text{Implicitely}, \; x_{\alpha'} := x - \beta') \\ g^{x_\alpha} & \text{otherwise} \end{cases}$$
The primary public-key is $\mathsf{ppk} = (\mathcal{PG} = (\mathbb{G}, g, p, \mathbb{G}_T, e, g), (h_0, \ldots, h_c))$.
- When receiving the i-th $\texttt{NTTSign}$ query $(i \in \llbracket 1, q \rrbracket)$ on $\boldsymbol{m}_i = (m_{i,1}, \ldots, m_{i,d})$:
  1. SecGen: $\mathcal{B}$ picks a random scalar $r_i \in_R \mathbb{Z}_p$ and defines $\mathsf{spk}_i = u_i = \hat{g}^{r_i \prod\limits_{b=1}^{f}(x+b-\beta')} h_{\alpha'}^{-m_{i,j'}}$.
  2. TTSign: $\mathcal{B}$ then computes the signature vector $\boldsymbol{\sigma}_i = (\sigma_{i,1}, \ldots, \sigma_{i,d})$ on $\boldsymbol{m}_i$ via

$$\begin{aligned} \sigma_{i,j} &= (u_i \cdot h_\alpha^{m_{i,j}})^{\frac{1}{x_\alpha + \beta}} \\ &= \begin{cases} g^{r_i} & \text{if} \quad j = j' \\ \hat{g}^{r_i \prod\limits_{b \neq \beta}(x+b-\beta')} \hat{g}^{(m_{i,j} - m_{i,j'})(x-\beta') \prod\limits_{b \neq \beta, \beta'}(x+b-\beta')} & \text{if} \quad \alpha = \alpha' \wedge \beta \neq \beta' \\ u_i^{1/(x_\alpha+\beta)} h_\alpha^{m_{i,j}/(x_\alpha+\beta)} & \text{otherwise} \end{cases} \end{aligned}$$

where $j = \alpha \cdot f + \beta$ and $\alpha \in \llbracket 0, c \rrbracket$ and $\beta \in \llbracket 0, f-1 \rrbracket$. Since x_α (for $\alpha \neq \alpha'$) is chosen by $\mathcal{B}$, the last equation can be computed. It is easy to see the simulated distribution is identical to the real scheme, since $\hat{g}$ from f-CDHI challenge is a random generator of $\mathbb{G}$.

Eventually, the adversary $\mathcal{F}$ outputs a forgery σ^* on a message m^* for some previously established spk_{i^*} $(i^* \in \llbracket 1, q \rrbracket)$. With probability $1/d$ the forgery is for the j'-th index. As σ^* is valid we have

$$\sigma^* = (u_{i^*} h_{\alpha'}^{m^*})^{1/(x_{\alpha'}+\beta')} = \hat{g}^{r_{i^*} \prod\limits_{b \neq \beta'}(x+b-\beta')} (\hat{g}^{(x-\beta') \cdot (m^* - m_{i,j'}) \cdot \prod\limits_{b \neq \beta'}(x+b-\beta')})^{1/x}$$

As we know $m^*, m_{i,j'}, r_i$, and $m^* \neq m_{i,j'}$ this allows to compute the helper value

$$(\sigma^*/g^{r_{i^*}})^{1/(m^* - m_{i,j'})} = (\hat{g}^{(x-\beta') \prod\limits_{b \neq \beta'}(x+b-\beta')})^{1/x}.$$

The helper value can be written as $\hat{g}^{\frac{\text{poly}(x)}{x}}$, where $\text{poly}(x)$ admits $\{\beta' - b : b \in \llbracket 1, f \rrbracket \wedge b \neq \beta'\} \cup \{\beta'\}$ as roots. Using partial fraction decomposition, it can be rewritten as $\hat{g}^{\text{poly}'(x)} \hat{g}^{\frac{\beta' \prod_{b \neq \beta'}(\beta' - b)}{x}}$ where poly' is a polynomial of degree $f-1$. Due to its degree, $\hat{g}^{\text{poly}'(x)}$ can be efficiently computed from the challenge, so $\mathcal{B}$ can recover $g^{\frac{1}{x}}$ to solve the f-CDHI challenge with probability $\varepsilon = \varepsilon'/d$. □

Construction from SIS Useful facts about lattice are recalled in Appendix A. Our scheme is defined as follows:

Let $k = \lceil \log p \rceil = O(\log \lambda)$, $\bar{m} = O(\lambda k)$ and $m = \bar{m} + \lambda k$ be the dimension of the signature. Let $\mathcal{D} = D_{\mathbb{Z}^{\bar{m}\times\lambda k}, \omega(\sqrt{\log\lambda})}$ be the Gaussian distribution over $\mathbb{Z}^{\bar{m}\times\lambda k}$ with parameter $\omega(\sqrt{\log\lambda})$ and let $s = O(\sqrt{\lambda k})$ be a Gaussian parameter. Then the signature scheme $\mathsf{TTSig}_{\mathsf{SIS}} = (\mathsf{PriGen}, \mathsf{SecGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ with message space $\{0,1\}^{\ell}$ is defined as follows:

- $\mathsf{PriGen}(1^{\lambda}, d)$: pick a random matrix $\mathbf{A}_0 \in_R \mathbb{Z}_p^{\lambda\times\ell}$. For $i = 1, \ldots, d$, sample $(\mathbf{A}_i, \mathbf{R}_i) \leftarrow_{\$} \mathsf{GenTrap}^{\mathcal{D}}(1^{\lambda}, 1^m, p)$. Define $\mathsf{ppk} = (\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_d)$ and $\mathsf{psk} = (\mathbf{R}_1, \ldots, \mathbf{R}_d)$.
- $\mathsf{SecGen}(\mathsf{psk}, \mathsf{ppk}, d)$: choose a random vector $\mathbf{u} \in_R \mathbb{Z}_p^{\lambda}$. Define $\mathsf{spk} = \mathbf{u}$ and $\mathsf{ssk} = \{\}$ is empty.
- $\mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}, \mathbf{m}_j; j)$: to sign the j-th message $\mathbf{m}_j \in \{0,1\}^{\ell}$, compute the syndrome $\mathbf{y}_j = \mathbf{u} - \mathbf{A}_0\mathbf{m}_j$. Then sample $\boldsymbol{\sigma}_j \in \mathbb{Z}^m$ from $D_{\Lambda^{\perp}_{\mathbf{y}_j}(\mathbf{A}_j), s\cdot\omega(\sqrt{\log\lambda})}$, $\boldsymbol{\sigma}_j \leftarrow_{\$} \mathsf{SampleD}(\mathbf{R}_j, \mathbf{A}_j, \mathbf{y}_j, s)$.
- $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}, \mathbf{m}_j, \boldsymbol{\sigma}_j; j)$: accept if $\|\boldsymbol{\sigma}_j\| \leq s \cdot \omega(\sqrt{\log\lambda}) \cdot \sqrt{m}$ and $\mathbf{A}_j\boldsymbol{\sigma}_j = \mathbf{u} - \mathbf{A}_0\mathbf{m}_j$; otherwise, reject.

Correctness of the scheme follows as explained in Lemmas 2 and 1.

Theorem 3. *If $\mathsf{SIS}_{p,\beta}$ is (t,ε)-hard for $\beta = \sqrt{\ell + s^2 \cdot \omega(\log\lambda) \cdot m)}$, then for any $q \in \mathbb{N}$, $\mathsf{TTSig}_{\mathsf{SIS}}$ is a (t', q, d, ε')-TT-SUF-NCMA signature scheme where $\varepsilon' = d\varepsilon + \mathsf{negl}(\lambda)$ and $t' = t - O(d \cdot q)$.*

Proof. Let $\mathcal{F}$ be a PPT adversary that (t', q, d, ε')-breaks the TT-SUF-NCMA security of $\mathsf{TTSig}_{\mathsf{SIS}}$. Then we construct an adversary $\mathcal{B}$ that (t,ε)-breaks the $\mathsf{SIS}_{p,\beta}$ problem. $\mathcal{B}$ is given a $\mathsf{SIS}_{p,\beta}$ instance $\mathbf{A} = [\mathbf{A}'|\mathbf{A}''] \in_R \mathbb{Z}_p^{\lambda\times m'}$ where $m' = \ell + m$ and $\mathbf{A}' \in_R \mathbb{Z}_p^{\lambda\times\ell}$ and $\mathbf{A}'' \in_R \mathbb{Z}_p^{\lambda\times m}$.

Simulation. $\mathcal{B}$ simulates $\mathsf{PriGen}(1^{\lambda}, d)$: it guesses a random $i^* \in_R \{1, \ldots, d\}$ and defines $\mathbf{A}_0 = \mathbf{A}'$ and $\mathbf{A}_{i^*} = \mathbf{A}''$. For $i \neq i^*$, $\mathcal{B}$ generates $\mathbf{A}_i$ and $\mathbf{R}_i$ as in the real scheme. Then $\mathcal{B}$ sends $\mathsf{ppk} = (\mathbf{A}_0, \ldots, \mathbf{A}_d)$ to $\mathcal{F}$.

Upon receiving the d messages $(\mathbf{m}_1, \ldots, \mathbf{m}_d)$ from $\mathcal{F}$, $\mathcal{B}$ simulates the corresponding signatures and the secondary verification key: it samples a $\boldsymbol{\sigma}_{i^*}$ from the Gaussian $D_{\mathbb{Z}^m, s\cdot\omega(\sqrt{\log\lambda})}$ and computes $\mathbf{u} = [\mathbf{A}_0|\mathbf{A}_{i^*}] \cdot \begin{bmatrix} \mathbf{m}_{i^*} \\ \boldsymbol{\sigma}_{i^*} \end{bmatrix}$ and defines $\mathsf{spk} = \mathbf{u}$. $\mathcal{B}$ uses $\mathbf{R}_i$ to compute $\boldsymbol{\sigma}_i$ as in the real scheme for $i \neq i^*$. Then $\mathcal{B}$ responds $\mathcal{F}$ with the signatures $\{\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_d\}$ and the secondary verification key spk.

The simulation is statistically close to the real execution. According to Lemma 1, the simulated $\mathbf{A}_{i^*}$ is $\mathsf{negl}(\lambda)$-far from the real distribution. It is easy to see the signatures $\boldsymbol{\sigma}_i$ for $i \neq i^*$ are identical to the scheme definition. It remains to show the simulated joint distribution $\{\mathsf{spk}, \boldsymbol{\sigma}_{i^*}\}$ is statistically close to the real distribution. Firstly, in the real scheme, spk is uniformly random over $\mathbb{Z}_p^{\lambda}$. In the simulation, $\mathsf{spk} = \mathbf{u} = \mathbf{A}_0\mathbf{m}_{i^*} + \mathbf{A}_{i^*}\boldsymbol{\sigma}_{i^*}$, where $\boldsymbol{\sigma}_{i^*} \in D_{\mathbb{Z}^m, s\cdot\omega(\sqrt{\log\lambda})}$ and $s \cdot \omega(\sqrt{\log\lambda}) = O(\sqrt{\lambda k})\omega(\sqrt{\log\lambda}) > \omega(\sqrt{\log m})$. By Lemma 3, for all but a $2p^{-\lambda}$ fraction of all $\mathbf{A}_{i^*} \in \mathbb{Z}_p^{\lambda\times m}$, $\mathbf{A}_{i^*}\boldsymbol{\sigma}_{i^*}$ is statistically close to uniform over

$\mathbb{Z}_p^{\lambda}$, which implies spk is statistically close to the real distribution. Secondly, in the real scheme, $\boldsymbol{\sigma}_{i^*}$ is sampled from the Gaussian $D_{\Lambda^{\perp}_{\mathbf{y}_{i^*}}(\mathbf{A}_{i^*}),s\cdot\omega(\sqrt{\log\lambda})}$ where $\mathbf{y}_{i^*} = \mathbf{u} - \mathbf{A}_0\mathbf{m}_{i^*}$. In the simulation, $\boldsymbol{\sigma}_{i^*}$ is sampled from $D_{\mathbb{Z}^m,s\cdot\omega(\sqrt{\log\lambda})}$ and it is easy to see $\boldsymbol{\sigma}_{i^*} \in \Lambda^{\perp}_{\mathbf{y}_{i^*}}(\mathbf{A}_{i^*})$, since $\mathbf{A}_{i^*}\boldsymbol{\sigma}_{i^*} = \mathbf{u} - \mathbf{A}_0\mathbf{m}_{i^*} = \mathbf{y}_{i^*}$. Thus, the simulated $\boldsymbol{\sigma}_{i^*}$ is identical to the real scheme.

Extracting $\mathsf{SIS}_{p,\beta}$ solution. Once $\mathcal{F}$ outputs a forgery $(\mathbf{m}^*,\boldsymbol{\sigma}^*)$, $\mathcal{B}$ aborts if $(\mathbf{m}^*,\boldsymbol{\sigma}^*)$ is not valid under $\mathbf{A}_{i^*}$. Otherwise, since $(\mathbf{m}^*,\boldsymbol{\sigma}^*)$ is valid signature, we have

$$[\mathbf{A}_0|\mathbf{A}_{i^*}] \cdot \begin{bmatrix}\mathbf{m}^*\\ \boldsymbol{\sigma}^*\end{bmatrix} = \mathbf{u} = [\mathbf{A}_0|\mathbf{A}_{i^*}] \cdot \begin{bmatrix}\mathbf{m}_{i^*}\\ \boldsymbol{\sigma}_{i^*}\end{bmatrix}.$$

Define $\mathbf{z} = \begin{bmatrix}\mathbf{m}^*\\ \boldsymbol{\sigma}^*\end{bmatrix} - \begin{bmatrix}\mathbf{m}_{i^*}\\ \boldsymbol{\sigma}_{i^*}\end{bmatrix}$. By the strong unforgeability of $\mathsf{TTSig}_{\mathsf{SIS}}$, $(\mathbf{m}^*,\boldsymbol{\sigma}^*) \neq (\mathbf{m}_{i^*},\boldsymbol{\sigma}_{i^*})$ and thus $\mathbf{z} \neq \mathbf{0}$. We claim $\mathbf{z}$ is the solution to the $\mathsf{SIS}_{p,\beta}$ problem instance $\mathbf{A}$, since

$$\mathbf{A} \cdot \mathbf{z} = \mathbf{A} \cdot (\begin{bmatrix}\mathbf{m}^*\\ \boldsymbol{\sigma}^*\end{bmatrix} - \begin{bmatrix}\mathbf{m}_{i^*}\\ \boldsymbol{\sigma}_{i^*}\end{bmatrix}) = [\mathbf{A}_0|\mathbf{A}_{i^*}] \cdot (\begin{bmatrix}\mathbf{m}^*\\ \boldsymbol{\sigma}^*\end{bmatrix} - \begin{bmatrix}\mathbf{m}_{i^*}\\ \boldsymbol{\sigma}_{i^*}\end{bmatrix}) = \mathbf{0}.$$

and $\|\mathbf{z}\|^2 \leq \ell + s^2\omega(\sqrt{\log\lambda})^2 m = \beta^2$ by the triangle inequality. The successful probability of $\mathcal{B}$ is $\varepsilon = \frac{\varepsilon'}{d} - \mathsf{negl}(\lambda)$ and its running time is $t = t' + O(d \cdot q)$. □

4 Generic Constructions of Non-Adaptive Signatures

In this section, we give two constructions of non-adaptively secure signature scheme Sig from any non-adaptively secure two-tier signature TTSig. The first construction is from a one-time two-tier signature scheme and the second construction is from a d-time two-tier signature scheme. Both constructions have tight security. The basic idea behind our constructions is as follows.

Basic Idea. In our constructions, the signer implicitly holds a tree. Each node has an out-degree d and the depth of the tree is h. Every node, including the leaves, $v \in \{1,\dots,d\}^{\leq h}$ has a label L_v which is a secondary public key of TTSig. All nodes can be computed “on the fly." Each leaf is used to sign a single message. We have $d^h = 2^{\lambda}$ (or, equivalently, $h \log d = \lambda$), where the scheme can sign up to 2^{λ} messages.

When signing message m, the signer takes the leftmost unused leaf $v_h \in \{1,\dots,d\}^h$ in the tree and generates the label $L_{v_h} \leftarrow_{\$} \mathsf{SecGen}(\mathsf{ppk},\mathsf{psk})$. Define $L_{v_{h+1}} = m$. Then the path from the root v_0 to v_h is computed. For each undefined node v_i on the path, the signer assigns label $L_{v_i} \leftarrow_{\$} \mathsf{SecGen}(\mathsf{ppk},\mathsf{psk})$. After that, every node on the path is signed using the label (i.e., the secondary secret key) of its parent. In this step, we have different signing methods depending on whether $d = 1$ or $d \geq 2$.

- $d = 1$: The signer holds a binary Merkle tree. When signing the nodes on the path, the signer takes the node v_i in the top-down manner and signs both children of v_i under L_{v_i}, $\sigma_{i+1} \leftarrow_{\$} \mathsf{Sign}(\mathsf{psk},\mathsf{ssk}_{v_i},\mathsf{Child}_l||\mathsf{Child}_r)$ where ssk_{v_i}

is the secondary secret key associated with node v_i, and Child_l and Child_r are the left and right children of node v_i respectively. This construction can be viewed as a generalization of the tree-based signature by Hofheinz and Jager [30].

- $d \geq 2$: The signer holds a flat-tree with out-degree d. When signing the nodes on the path, the signer takes the node v_i in the top-down manner. Assume the jth child Child_j of v_i is on the path. Then the signer uses ssk_{v_i} to sign Child_j, $\sigma_{i+1} \leftarrow_{\$} \mathsf{Sign}(\mathsf{psk}, \mathsf{ssk}_{v_i}, \mathsf{Child}_j)$.

The signer outputs the path and the two-tier signatures on the path as the signature of m. Details are given in the definitions of the schemes.

Note that both of our schemes are stateful. One can use the technique of Goldreich [26] to make them stateless. Precisely, the randomness used to generate secondary secret key ssk_{v_i} for each node v_i will be derived by a pseudo-random function. Another pseudo-random function will be used to determine the leaf used to sign a given message. As this technique is quite standard for Merkle-tree-based signatures, we skip the details here and refer the reader to Section 3.2.3 of [34].

Moreover, it is well-known that a non-adaptively secure signature can be tightly transferred to be an adaptively secure signature by using a Chameleon Hash [35]. This is explicitly proven in the full version of [33].

4.1 Construction from any One-Time Two-Tier Signature

Let $\mathsf{TTSig} = (\mathsf{PriGen}, \mathsf{SecGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ be a one-time two-tier signature scheme with message space $\{0,1\}^*$. The stateful signature scheme $\mathsf{BinTree}[\mathsf{TTSig}]$ $= (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is based on a binary tree of height $h = \lambda$ and is defined as follows. Figure 2 shows the nodes involved in signing the i-th message m.

- $\mathsf{Gen}(1^\lambda)$: Generate a primary key $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_{\$} \mathsf{PriGen}(1^\lambda, 1)$. The label of the root node ϵ is also generated $(\mathsf{spk}_\epsilon, \mathsf{ssk}_\epsilon) \leftarrow_{\$} \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ and $L_\epsilon = \mathsf{spk}_\epsilon$. Define the verification key $\mathsf{pk} = (\mathsf{ppk}, \mathsf{spk}_\epsilon)$ and the signing key $\mathsf{sk} = (\mathsf{psk}, \mathsf{ssk}_\epsilon)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: To sign a message m, the signer proceeds in two steps:
  - $\mathsf{Node\ generation\ step}$: The signer takes the leftmost unused leaf $v_h \in \{0,1\}^h$ and searches the binary path $(v_0, v_1, v_2, \ldots, v_h)$ from the root $v_0 = \epsilon$ to v_h, i.e., v_i is the i-th prefix of v_h. For each node v_i on the path (including the leaf v_h), if v_i's label L_{v_i} is not defined, then the signer generates $(\mathsf{spk}_{v_i}, \mathsf{ssk}_{v_i}) \leftarrow_{\$} \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ and assigns $L_{v_i} = \mathsf{spk}_{v_i}$. For the sibling $\bar{v}_i$ of v_i, the corresponding secondary public key and secret key are generated in the same way, $(\mathsf{spk}_{\bar{v}_i}, \mathsf{ssk}_{\bar{v}_i}) \leftarrow_{\$} \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ and $L_{\bar{v}_i} = \mathsf{spk}_{\bar{v}_i}$.
  - $\mathsf{Path\ authentication\ step}$: Define $M_h = m$. For each node v_i $(i = h-1, \ldots, 0)$ on the path, define the message associated with v_i by $M_i = L_{v_i||0}||L_{v_i||1}$, where $L_{v_i||0}$ and $L_{v_i||1}$ are labels of the left and right children of v_i respectively. Then the signer computes the signatures on the path as $\sigma_i = \mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}_{v_i}, M_i)$ for $i = 0, \ldots, h$.

The signer returns $\sigma = (v_h, M_0, \ldots, M_{h-1}, \sigma_0, \ldots, \sigma_h)$ as the signature of m.

- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: A signature $\sigma = (v_h, M_0, \ldots, M_{h-1}, \sigma_0, \ldots, \sigma_h)$ on the message m is verified in the natural way. Define $M_h = m$. Note that each M_{i-1} ($i = 1, \ldots, h$) contains the secondary public keys of v_{i-1}'s children, $L_{v_{i-1}||0}$ and $L_{v_{i-1}||1}$. Hence, we check if $\mathsf{TTVerify}(\mathsf{ppk}, L_{v_i}, M_i, \sigma_i) = 1$. If that is true for $i = 0, \ldots, h$, then it outputs 1, otherwise 0.

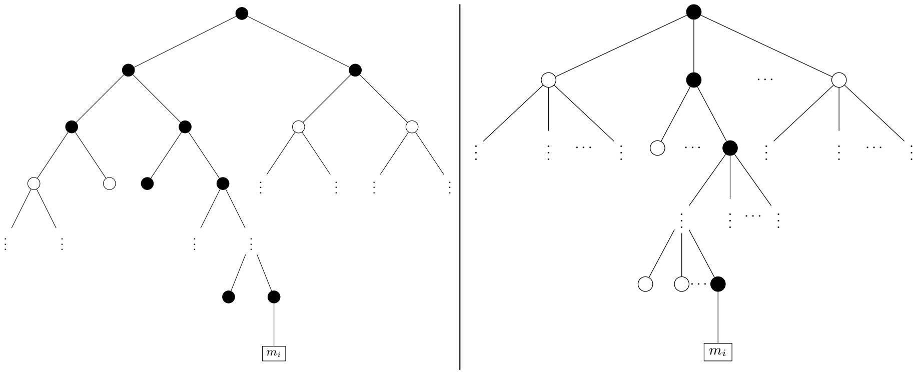

Fig. 2. Nodes in black are used in the i-th Signature with $\mathsf{BinTree}[\mathsf{TTSig}]$, left and d-$\mathsf{Tree}[\mathsf{TTSig}]$, right.

The following theorem shows the non-adaptively security of $\mathsf{BinTree}[\mathsf{TTSig}]$ is tightly reduced to the security of the one-time two-tier signature TTSig.

Theorem 4. *If* TTSig *is* $(t, q, 1, \varepsilon)$-$\mathsf{TT\text{-}EUF\text{-}NCMA}$ *secure, then* $\mathsf{Sig} = \mathsf{BinTree}[\mathsf{TTSig}]$ *is* (t', ε', q')-$\mathsf{EUF\text{-}NCMA}$ *secure, where* $t' = t - O(hq')$, $\varepsilon' = \varepsilon$, *and* $q' = \frac{q}{h+1}$.

Proof. Let $\mathcal{F}'$ be a PPT adversary that breaks the $\mathsf{EUF\text{-}NCMA}$-security of Sig with success probability ε' and time complexity t' and makes q' times non-adaptive message queries. Then we construct an adversary $\mathcal{F}$ to $(t, q, 1, \varepsilon)$-breaks the $\mathsf{TT\text{-}EUF\text{-}NCMA}$ security of TTSig with the parameters given above. First, $\mathcal{F}$ is given a challenge TTSig primary public key ppk.

SIMULATION. Recall that $\mathcal{F}'$ is an adversary for non-adaptive security, which means $\mathcal{F}'$ will output q' messages $m_1, \ldots, m_{q'}$ before seeing the verification key. In the following we explain how $\mathcal{F}$ generates the signatures on each m_i and the verification key of Sig without knowing the real signing key of TTSig.

$\mathcal{F}$ generates the binary tree in a bottom-up fashion by using the oracle $\mathtt{NTTSign}$ (note that the number of leaves are the same as the number of the signing queries q' and, thus, all the leaves are defined after signing q' messages). For each i-th query to $\mathtt{NTTSign}$ ($1 \leq i \leq q'$), $\mathcal{F}$ does the following:

- For a leaf $v_h^{(i)}$, $\mathcal{F}$ defines $M_h^{(i)} = m_i$ and queries $(\mathsf{spk}_{v_h^{(i)}}, \sigma_h^{(i)}) \leftarrow_{\$} \mathtt{NTTSign}(M_h^{(i)})$. Define $L_{v_h^{(i)}} = \mathsf{spk}_{v_h^{(i)}}$.

- For an internal node v_j (for each $0 \leq j \leq h-1$), $\mathcal{F}$ defines $M_j^{(i)} = L_{v_{j-1}||0}||L_{v_{j-1}||1}$. $\mathcal{F}$ queries $(\mathsf{spk}_{v_j}, \sigma_j^{(i)}) \leftarrow_\$ \mathtt{NTTSign}(M_j^{(i)})$. Define $L_{v_j} = \mathsf{spk}_{v_j}$.
- The signature σ_i on m_i is $(v_h^{(i)}, M_0^{(i)}, \ldots, M_{h-1}^{(i)}, \sigma_0^{(i)}, \ldots, \sigma_h^{(i)})$.

Finally, $\mathcal{F}$ returns the verification key $\mathsf{pk} = (\mathsf{ppk}, \mathsf{spk}_\epsilon)$ and the signatures $(\sigma_1, \ldots, \sigma_{q'})$ to $\mathcal{F}'$.

Note that the simulation is identical to the real execution. Firstly, ppk is from the TTSig challenger, which is distributed identically to the real distribution. Secondly, due to the correctness of $\mathtt{NTTSign}$, the binary tree generated by $\mathcal{F}$ is identical to the real one and the same for the corresponding signatures on the path. Thus, the simulated verification key and signatures for q'-messages are identical. Moreover, $\mathcal{F}$ makes one $\mathtt{NTTSign}$ query per node and makes hence a total of $q = q'(h+1)$ queries.

Extracting the forgery for TTSig. Let the set Good contain all the labels L_{v_j} assigned by $\mathcal{F}$. Recall that a forgery (m^*, σ^*) consists of $\sigma^* = (v_h^*, M_0^*, \ldots, M_{h-1}^*, \sigma_0^*, \ldots \sigma_h^*)$ and M_j^* contains the labels of both children of node v_j^*. Then, after $\mathcal{F}'$ outputs a forgery (m^*, σ^*) for Sig, $\mathcal{F}$ can search the largest index $\delta \in \{0, \ldots, h\}$ such that $L_{v_\delta^*}$ is in set Good. $L_{v_\delta^*}$ was previously defined by running $(\mathsf{spk}_{v_\delta^*}, \sigma_\delta) \leftarrow \mathtt{NTTSign}(M')$ for some M'. If (m^*, σ^*) is a valid $\mathsf{EUF\text{-}NCMA}$ forgery, $\mathcal{F}$ can find $(M_\delta^*, \sigma_\delta^*)$ such that $\mathsf{TTVerify}(\mathsf{ppk}, \mathsf{spk}_{v_\delta^*}, M_\delta^*, \sigma_\delta^*) = 1$ where $M_\delta^* \neq M'$. Thus, $\mathcal{F}$ can break the $\mathsf{TT\text{-}EUF\text{-}NCMA}$ security of TTSig with probability $\varepsilon = \varepsilon'$. A similar argument can be applied to prove the strong $\mathsf{EUF\text{-}NCMA}$ security of Sig when TTSig is strongly $\mathsf{TT\text{-}EUF\text{-}NCMA}$ secure. □

4.2 Construction from any d-Time Two-Tier Signature

Let $\mathsf{TTSig} = (\mathsf{PriGen}, \mathsf{SecGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ be a d-time two-tier signature with message space $\{0,1\}^*$. The stateful signature scheme d-$\mathsf{Tree}[\mathsf{TTSig}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is defined as follows, once again you can refer to Figure 2 to see the nodes involved:

- $\mathsf{Gen}(1^\lambda)$: It generates a d-time primary key, $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_\$ \mathsf{PriGen}(1^\lambda, d)$. The label of the root $v_0 = \epsilon$ is also generated $(\mathsf{spk}_\epsilon, \mathsf{ssk}_\epsilon) \leftarrow_\$ \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ and $L_\epsilon := \mathsf{spk}_\epsilon$. Define the verification key $\mathsf{pk} := (\mathsf{ppk}, \mathsf{spk}_\epsilon)$ and the signing key $\mathsf{sk} := (\mathsf{psk}, \mathsf{ssk}_\epsilon)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: To sign a message m, the signer proceeds in two steps:
  - Nodes generation step: The signer takes the leftmost unused leaf $v_h \in \{1, \ldots, d\}^h$ and searches the path $(v_0, \ldots, v_h)$ from the root $v_0 = \epsilon$ to v_h. Define $L_{v_h} := m$ and for each internal node v_i on the path, if v_i's label L_{v_i} is not defined, then the signer generates $(\mathsf{spk}_{v_i}, \mathsf{ssk}_{v_i}) \leftarrow_\$ \mathsf{SecGen}(\mathsf{ppk}, \mathsf{psk})$ and assigns $L_{v_i} := \mathsf{spk}_{v_i}$.
  - Path authentication step: Each L_{v_i} $(i = 1, \ldots, h)$ on the path is signed under $L_{v_{i-1}} = \mathsf{spk}_{v_{i-1}}$, $\sigma_i \leftarrow_\$ \mathsf{TTSign}(\mathsf{psk}, \mathsf{ssk}_{v_{i-1}}, L_{v_i}; j)$ where $v_i = v_{i-1}||j$ and $1 \leq j \leq d$. The d-time TTSign is a stateful algorithm and j is the state.

The signer returns $\sigma = (v_h, L_{v_1}, \ldots, L_{v_{h-1}}, \sigma_1, \ldots, \sigma_h)$ as the signature of m.

- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse $\sigma = (v_h, L_{v_1}, \ldots, L_{v_h}, \sigma_1, \ldots, \sigma_h)$. The verifier defines $L_{v_h} := m$ and checks if $\mathsf{TTVerify}(\mathsf{ppk}, L_{v_{i-1}}, L_{v_i}, \sigma_i; j) = 1$ for all $i = 1, \ldots, h$, where $v_{i+1} = v_i || j$ $(1 \leq j \leq d)$. If that is true, then it outputs 1, otherwise 0. Here the d-time $\mathsf{TTVerify}$ is a stateful algorithm and j is the state.

The following theorem tightly reduces the non-adaptively security of d-$\mathsf{Tree}[\mathsf{TTSig}]$ to the one of the d-time two-tier signature TTSig.

Theorem 5. *If* TTSig *is* (t, q, d, ε)-$\mathsf{TT\text{-}EUF\text{-}NCMA}$ *secure, then* $\mathsf{Sig} = d$-$\mathsf{Tree}[\mathsf{TTSig}]$ *is* (t', ε', q')-$\mathsf{EUF\text{-}NCMA}$ *secure, where* $t' = t - O(hq'), \quad \varepsilon' = \varepsilon,$ *and* $q' = \frac{q}{h}$.

Proof. The security proof is a generalization of the proof of the Cramer-Damgård scheme [19], and it is rather similar to the proof of Theorem 4. Therefore we only sketch it. The major difference between Sig and $\mathsf{BinTree}[\mathsf{TTSig}]$ is that each internal node v in Sig uses a d-time signature to sign its d-many children one by one, while in $\mathsf{BinTree}[\mathsf{TTSig}]$ each internal node v can only sign its both children one-time.

Assume $\mathcal{F}'$ (t', ε', q')-breaks $\mathsf{EUF\text{-}NCMA}$-security of Sig. Then we construct $\mathcal{F}$ break $\mathsf{TT\text{-}EUF\text{-}NCMA}$ security of TTSig:

Simulation. Similar to the proof of Theorem 4, given q' messages, $\mathcal{F}$ can simulate all the tree nodes and the signature on the path by asking the d-time signing oracle `NTTSign` in a bottom-up fashion. By the correctness of `NTTSign`, it is easy to see the simulation is identical to the Sig definition. Moreover, $\mathcal{F}$ makes one `NTTSign` query per node and makes hence a total of $q = q' \cdot h$ queries.

Extracting the forgery for TTSig. After $\mathcal{F}'$ outputs a success forgery (m^*, σ^*), $\mathcal{F}$ defines $L_{v^*_{h+1}} := m^*$ and finds the forgery for TTSig following the same step in the proof of Theorem 4. Thus, $\varepsilon = \varepsilon'$. □

Acknowledgements

We thank Mihir Bellare for his valuable comments. Part of this work was done while Olivier Blazy was employed at Ruhr-University Bochum. All authors were (partially) funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research. Jiaxin Pan was also partially funded by the German Israeli Foundation.

References

1. M. Abe, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In K. Kurosawa and G. Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 312–331. Springer, 2013.
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Aug. 2010.

3. C. Bader, D. Hofheinz, T. Jager, E. Kiltz, and Y. Li. Tightly-secure authenticated key exchange. Cryptology ePrint Archive, Report 2014/797, 2014.
4. M. Bellare and T. Ristov. A characterization of chameleon hash functions and new, efficient designs. *Journal of Cryptology*, 27(4):799–823, Oct. 2014.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
6. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, May 1996.
7. M. Bellare and S. Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In T. Okamoto and X. Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, Apr. 2007.
8. O. Blazy, S. Kakvi, E. Kiltz, and J. Pan. Tightly-secure signatures from chameleon hash functions. Cryptology ePrint Archive, Report 2014/1021, 2014.
9. O. Blazy, E. Kiltz, and J. Pan. (Hierarchical) identity-based encryption from affine message authentication. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Aug. 2014.
10. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.
11. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr. 2008.
12. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Aug. 2004.
13. D. Boneh, I. Mironov, and V. Shoup. A secure signature scheme from bilinear maps. In M. Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 98–110. Springer, Apr. 2003.
14. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.
15. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, May 2010.
16. D. Catalano and R. Gennaro. Cramer-Damgård signatures revisited: Efficient flat-tree signatures based on factoring. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 313–327. Springer, Jan. 2005.
17. J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Aug. 2013.
18. B. Chevallier-Mames and M. Joye. A practical and tightly secure signature scheme without hash function. In M. Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 339–356. Springer, Feb. 2007.
19. R. Cramer and I. Damgård. New generation of secure and practical RSA-based signatures. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 173–185. Springer, Aug. 1996.
20. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99*, pages 46–51. ACM Press, Nov. 1999.
21. Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of the full domain hash. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466. Springer, Aug. 2005.

22. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 234–246. Springer, Aug. 1994.
23. A. Escala, G. Herold, E. Kiltz, C. Rafols, and J. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, volume 8043 of *LNCS*, pages 449–475. Springer, Aug. 2013.
24. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
25. E.-J. Goh, S. Jarecki, J. Katz, and N. Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, Oct. 2007.
26. O. Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 104–110. Springer, Aug. 1986.
27. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.
28. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Dec. 2006.
29. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Apr. 2008.
30. D. Hofheinz and T. Jager. Tightly secure signatures and public-key encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Aug. 2012.
31. D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 647–666. Springer, Dec. 2011.
32. S. Hohenberger and B. Waters. Realizing hash-and-sign signatures under standard assumptions. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 333–350. Springer, Apr. 2009.
33. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 654–670. Springer, Aug. 2009.
34. J. Katz. *Digital Signatures.* Springer, 2010.
35. H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS 2000.* The Internet Society, Feb. 2000.
36. Y. Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 241–254. Springer, May 2003.
37. R. C. Merkle. A certified digital signature. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Aug. 1989.
38. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Apr. 2012.
39. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, Oct. 2004.

40. P. Mohassel. One-time signatures and chameleon hash functions. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 302–319. Springer, Aug. 2010.
41. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.
42. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.
43. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.
44. S. Schäge. Tight proofs for signature schemes without random oracles. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 189–206. Springer, May 2011.
45. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

A Hardness Assumptions

We now define the hardness assumptions that we have used in our results.

Group generator algorithms. We define an algorithm $\mathsf{GroupGen}$, that on input of 1^λ gives us $\mathcal{G} = (\mathbb{G}, g, p)$, such that $\mathbb{G} = \langle g \rangle$ is a multiplicative group of order p and $\log p = \lambda$.

Let $\mathsf{PGroupGen}$ be an algorithm that on input 1^λ outputs a description of a bilinear group $\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$ such that $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are two cyclic groups of prime-order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing satisfying the following properties:

1. $\mathbb{G}_T = \langle e(g, g) \rangle$ (in particular $e(g, g) \neq 1$).
2. $\forall a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$.

We now discuss the computational assumptions that we use in this setting. All the assumptions below are defined relative to either $\mathsf{GroupGen}$ or $\mathsf{PGroupGen}$. For compactness, we use the Setup algorithm, which can be in either setting.

Linear Assumption. The linear assumption, denoted by LIN, states that given three random generators g, h, k of $\mathbb{G}$ and a tuple (g^u, h^v, k^c) where $u, v \in_R \mathbb{Z}_p$ and $c = u + v$ or random in $\mathbb{Z}_p$, it is hard for the adversary $\mathcal{A}$ to guess $c = u + v$ or c is random. LIN is said to be (t, ε)-hard if for all adversaries $\mathcal{A}$ running in time at most t, we have

$$\Pr\left[\mathcal{A}(g, h, k, (g^u, h^v, k^c))_{\$} \to \text{`}c = u + v\text{' or not}\right] \leq \varepsilon.$$

Computational Diffie-Hellman Assumption. The Computational Diffie-Hellman Assumption, denoted by CDH, states that given $\mathcal{G} = (\mathbb{G}, g, p)$ and elements g^a, g^b, it is hard to compute g^{ab}. CDH is said to be (t, ε)-hard if for all adversaries $\mathcal{A}$ running in time at most t, we have

$$\Pr\left[\mathcal{G} \leftarrow_{\$} \mathsf{Setup}(1^\lambda), a, b \in_R \mathbb{Z}_p : g^{ab} \leftarrow_{\$} \mathcal{A}(\mathcal{G}, g^a, g^b)\right] \leq \varepsilon.$$

f-Computational Diffie-Hellman Inversion Assumption. The f- Computational Diffie-Hellman Inversion Assumption, denoted by f-CDHI, states that given $\mathcal{G} = (\mathbb{G}, g, p)$ and elements $g^x, g^{x^2}, g^{x^3}, \dots g^{x^f}$, it is hard to compute $(g^{\frac{1}{x}})$. f-CDHI is said to be (t, ε)-hard if for all adversaries $\mathcal{A}$ running in time at most t, we have

$$\Pr\left[\mathcal{G} \leftarrow_{\$} \mathsf{Setup}(1^\lambda), x \in_R \mathbb{Z}_p : g^{\frac{1}{x}} \leftarrow_{\$} \mathcal{A}(\mathcal{G}, g^x, g^{x^2}, g^{x^3}, \dots g^{x^f})\right] \leq \varepsilon.$$

We note that 1-CDHI is tightly equivalent to CDH.

RSA Assumption. The RSA Assumptions, denoted by RSA, states that given (N, e, x^e), where N is a random λ-bit RSA modulus generated by an algorithm $\mathsf{RSAGen}(1^\lambda)$ and $x \in_R \mathbb{Z}_N^*$, it is hard to compute x. RSA is said to be (t, ε)-hard, if for all adversaries $\mathcal{A}$ running in time at most t, we have:

$$\Pr\left[(N, e) \leftarrow_{\$} \mathsf{RSAGen}(1^\lambda), x \in_R \mathbb{Z}_N^* : x = \mathcal{A}(N, e, x^e)\right] \leq \varepsilon.$$

Lattices and SIS Assumption. for integers λ, m and for a prime p, let $\mathbf{A} \in \mathbb{Z}_p^{\lambda \times m}$. The m-dimensional integer lattice $\Lambda^\perp(\mathbf{A})$ is defined as

$$\Lambda^\perp(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \bmod p\}.$$

For any $\mathbf{u} \in \mathbb{Z}_p^\lambda$, define the coset

$$\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \bmod p\}.$$

The *short integer solution* problem $\mathsf{SIS}_{p,\beta}$ $(\beta > 0)$ is an average-case version of the approximate shortest vector problem on $\Lambda^\perp(\mathbf{A})$. It states that, given a uniformly random $\mathbf{A} \in \mathbb{Z}_p^{\lambda \times m}$ for $m = \text{poly}(\lambda)$, find a non-zero $\mathbf{z} \in \Lambda^\perp(\mathbf{A})$ and $\|\mathbf{z}\| \leq \beta$, where $\|\cdot\|$ is the Euclidean norm. $\mathsf{SIS}_{p,\beta}$ is (t, ε)-hard if all adversaries with running time t have a success probability of at most ε. It has been shown if $p \geq \beta\sqrt{\lambda} \cdot \omega(\sqrt{\log \lambda})$ then solving $\mathsf{SIS}_{p,\beta}$ is at least as hard as approximating the Shortest Independent Vectors Problem within approximation factor $\tilde{O}(\beta\sqrt{\lambda})$ in worst case [39,24].

Let $D_{\mathbb{Z}^m,s}$ be the Gaussian distribution over $\mathbb{Z}^m$ with center $\mathbf{0}$ and parameter s and, similarly, let $D_{\Lambda^\perp(\mathbf{A}),s}$ be the Gaussian distribution over $\Lambda^\perp(\mathbf{A})$ with center $\mathbf{0}$ and parameter s.

The following lemmas are useful for the definition and the security proof of our scheme.

Lemma 1 (Theorem 5.1 of [38]). *There is an efficient randomized algorithm* $\mathsf{GenTrap}^{\mathcal{D}}(1^\lambda, 1^m, p)$ *that, given any integers* $\lambda \geq 1$, $p \geq 2$, *and sufficiently large* $m = O(\lambda \log p)$, *outputs a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_p^{\lambda \times m}$ *and a trapdoor* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\mathsf{negl}(\lambda)$*-far from uniform and* $\mathbf{R}$ *is sampled from the Gaussian* $\mathcal{D}$.

Moreover, for any $\mathbf{y} \in \mathbb{Z}_p^\lambda$ *and large enough* $s = O(\sqrt{\lambda \log p})$, *there is an efficient randomized algorithm* $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{y}, s)$ *that samples from a distribution with* $\mathsf{negl}(\lambda)$ *statistical distance of* $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}), s \cdot \omega(\sqrt{\log \lambda})}$.

Lemma 2 (Lemma 4.4 of [39]). *Let $\mathbf{x} \leftarrow D_{\Lambda^\perp(\mathbf{A}),s}$ where $\mathbf{A} \in \mathbb{Z}_p^{\lambda \times m}$. Then the probability that $\|\mathbf{x}\| > s\sqrt{m}$ is negligible in λ.*

Lemma 3 (Corollary 5.4 of [24]). *Let λ be a positive integer and p be a prime, and let integer $m \geq 2\lambda \log p$. Then for all but a $2p^{-\lambda}$ fraction of all $\mathbf{A} \in \mathbb{Z}_p^{\lambda \times m}$ and for any $s \geq \omega(\sqrt{\log m})$, the distribution of the syndrome $\mathbf{y} = \mathbf{A}\mathbf{x} \mod p$ is statically close to uniform over $\mathbb{Z}_p^{\lambda}$, where $\mathbf{x}$ is from $D_{\mathbb{Z}^m,s}$.*

B Applications

In this appendix we show some applications of our almost structure-preserving signature scheme $\mathsf{Sig}_{d,f\text{-}\mathsf{CDHI}} = d\text{-}\mathsf{Tree}[\mathsf{TTSig}_{f\text{-}\mathsf{CDHI}}]$ and we get a more efficient tightly-secure CCA encryption in the multi-user and multi-challenge setting.

A Structure-Preserving signature over a bilinear group [2] considers signatures fully compatible with the Groth-Sahai methodology. Such signatures assume that messages, signatures and verification keys are in the same space ($\mathbb{G}$) and that verification can be expressed as simple pairing product equations.

When someone wants to commit to a signature, the naive approach consists in computing the signature, and then committing individually to each component of the signature. However, in many signature schemes, like ours, parts of the signature do not require the knowledge of the secret key and therefore do not require to be committed.[5] In the following, we relax the definition of structure preserving signatures and consider signatures where the verification equation is a pairing product equation in the elements that have to be committed. (To be more specific, we will allow hash values and scalars to appear in the verification equation as long as they are uncommitted/public values.)

In a symmetric bilinear group $\mathcal{PG} = (p, \mathbb{G}, g, \mathbb{G}_T, e)$, a Pairing-Product Equation is an equation of the form: $\prod_{i=1}^n e(X_i, A_i) \cdot \prod_{i=1}^n \prod_{j=i}^n e(X_i, X_j)^{\gamma_{i,j}} = t_T$, where A_i are public group elements in $\mathbb{G}$, $\gamma_{i,j}$ are public scalars in $\mathbb{Z}_p$, t_T is a public element in the target group $\mathbb{G}_T$, and X_i are variables in $\mathbb{G}$. In [29], the authors have shown how to build Non-Interactive Zero-Knowledge Proofs of Knowledge of solutions of such equations and have proven that their construction can be improved in the linear case ($\boldsymbol{\gamma} = \mathbf{0}$).

B.1 Tight Simulation-Sound NIZK in Pairing Groups

In this subsection, we revisit a technique introduced in [36,28] to obtain simulation-sound NIZK (also used in [30]) and instantiate it with our new signature scheme $\mathsf{Sig}_{d,f\text{-}\mathsf{CDHI}}$.

A Simulation-Sound Non-Interactive Zero-Knowledge (SSNIZK) Proofs of Knowledge, is a standard NIZK where the soundness holds even if the simulator is given simulated proofs.

[5] A good illustration consists in considering a Waters signature: $\sigma_1 = \mathsf{sk}F(m)^s, \sigma_2 = g^s$, committing σ_1 into C_1 is enough to completely hide the signature. (C_1, σ_2) leaks no information on the validity of the signature.

We build SSNIZK Proofs of Knowledge to prove that variables X verify a set of Pairing-Product Equations $\mathcal{S}$, for which we combine our non-adaptive signature scheme, a one-time two-tier signature (to make it adaptively secure) and Groth-Sahai Proofs of Knowledge [29].

The verification of the validity of our signature can be viewed as several linear Pairing-Product Equations. This will allow us to greatly improve the efficiency of the SSNIZK Proof of Knowledge.

Roadmap of the technique To construct a Simulation-Sound proof that some variables X verify a set $\mathcal{S}$ of equations, one uses the following roadmap assuming the crs contains $\mathsf{crs}_{\mathsf{GS}}$, a verification key pk for the Structure-Preserving Signature scheme Sig, and the prover already possesses a pair of primary keys $\mathsf{psk}, \mathsf{ppk}$ for a one-time two-tier signature scheme $\mathcal{S}_1$.

1. Generates a secondary signing/verification key pair $(\mathsf{ssk}, \mathsf{spk})$ for the one-time two-tier signature
2. Commits to a random tuple of elements R corresponding to a signature (the tuple should be random, but the size and type of elements committed should be consistent with what is expected from a signature).
3. Generates a Groth-Sahai proof π, that either X verifies this set $\mathcal{S}$, or that R is a valid signature under pk in the crs of the verification key spk of the one-time signature scheme.
4. He then sends this Groth-Sahai proof π, the verification key of the one-time signature, a one-time signature under $\mathsf{psk}, \mathsf{ssk}$ of everything.

Referring to [30], it can be shown that this scheme is Zero-Knowledge under the indistinguishability of the two types of Groth-Sahai crs, and that both the simulation-soundness and the soundness come from the unforgeability of two kind of signatures. The reductions inherit the tightness of the underlying signature schemes.

Instantiation from f-CDHI and efficiency comparison We now use our non-adaptive structure-preserving signature scheme based on f-CDHI (obtained by combining the d-time two-tier signature presented in Section 3.2, and the transformation from 4.2), together with the Strong one-time two-tier signature based on DLOG (see Section 3.1 with a DLOG-based Chameleon Hash), we obtain:

- $\mathcal{ZK}.\mathsf{Setup}(1^\lambda)$: generates a crs consisting of a bilinear group $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, an extra generator $\tilde{g}$ for the $\mathsf{ppk}_{\mathcal{S}_1}$ of the one-time signature scheme, a collision resistant Hash Function $\mathcal{H}$, a Groth-Sahai CRS crs_{GS} and the verification key $\mathsf{pk} = g^x_{i \in [\![1,c]\!]}$ for a Structure-Preserving Signature Scheme, which is also strongly unforgeable. The prover possesses a pair $(\mathsf{psk} = \alpha, \mathsf{ppk} = \tilde{g} = g^\alpha)$.
- $\mathcal{ZK}.\mathsf{Prove}(\mathsf{crs}, \mathcal{S}, X)$: where X is a set of variables satisfying the set of equations $\mathcal{S}$. First this samples a fresh secondary key pair for the strong adaptive

one-time two-tier signature scheme: a pair $(\mathsf{ssk}_{\mathcal{S}_1} = (\eta, \mu), \mathsf{spk}_{\mathcal{S}_1} = \tilde{g}^\eta g^\mu)$ for $\eta, \mu \in_R \mathbb{Z}_p$.
It then computes a Groth-Sahai proof π_{GS} stating that either X satisfies $\mathcal{S}$, or that σ is a valid signature on $\mathsf{spk}_{\mathcal{S}_1}$, by picking a fresh leaf in the signature tree, and generating a commitment σ of random values emulating a signature on the path (random group elements $\mathsf{spk}_i \in_R \mathbb{G}$ for the nodes of the tree on the path, reusing those already chosen on the shared path in previous proofs), a random scalar t for the one-time signature of $\mathsf{spk}_{\mathcal{S}_1}$ on the leaf, and $h+1$ commitments to fictive signature S_i of spk_{i+1} valid under spk_i. The proof consists of $h+1$ proofs of linear pairing product equations, so $3h+3$ group elements only where h is the depth of the tree $(h = \lambda/\log(d))$.
It then sends $\pi = \pi_{\mathsf{GS}}, \mathsf{spk}_{\mathcal{S}_1}, \sigma_{\mathsf{spk}_{\mathcal{S}_1}}(\pi_{\mathsf{GS}})$.
- $\mathcal{ZK}.\mathsf{Verify}(\mathsf{crs}, \mathcal{S}, \pi)$, checks the validity of the one-time two-tier signature, and then the validity of the Groth-Sahai proof.

The principal difference between this approach and the one in [30] resides in the signature scheme, in particular the sizes thereof. Their signature requires 10 group elements per node; to hide the signature, 6 of them have to be committed, resulting in 22 elements per node. The verification equation is a quadratic pairing product-equation, hence the sub-proof requires 9 group elements per node. The proof on the committed signature requires overall roughly 31λ group elements.

Recently, Abe et al. [1] have presented an optimization on this initial construction. They evaluated the cost of their corresponding part as roughly $21\lambda + 27$. (They presented several construction, but the others are either less efficient and/or not tight)

On the other hand, our signature based on f-CDHI requires two group elements per node (the child verification key and the signature itself) and one group element and a scalar for the last node. We need to hide one of these elements for each node. This means that we need 4 elements per node, and 3 group elements and a scalar for the last one. As explained previously, the verification equation can be viewed in this case as a linear pairing-product equation so on each node the proof consists of 3 group elements. We end up with a proof on the committed signature consisting of $(7\lambda)/\log(d) + 7$ group elements and 1 scalar. This is where, the trade-off comes into play, for a fair comparison to previous schemes, we need a signature relying on an equivalent assumption, as they are based on LIN, we need to rely on CDH, so $f = 1$-CDHI, we also want to have a reasonable sized CRS, so minimize d/f, and take $d = 2$. In the end, we can show that by increasing the CRS size by one element, we manage to reduce the size of the proofs by a factor 3.

B.2 Tight Multi-Challenge (and Multi-User) IND-CCA Scheme

IND-CCA encryption is a very useful primitive, but in some contexts, one may wish to give even more power to the adversary, he might be allowed to give q challenge tuples and only answer on one of them, or he might ask the challenges to be run on μ encryption keys. There is a transformation based on the

Naor-Yung paradigm [42] which allows to create a (μ, q)-CCA-Encryption from a (μ, q)-Structure-Preserving CPA-Encryption[6], and a SSNIZK in pairing groups.

This technique is described in more details in [30], where they show how to obtain a tight reduction to the CPA-encryption and the SSNIZK.

Roadmap of the technique To encrypt a message M, one obeys the following roadmap (assuming a crs containing two encryption keys $\mathsf{ek}_1, \mathsf{ek}_2$ for an IND-CPA scheme.):

1. Generates two CPA-encryptions of M, one under each two encryption keys in the CRS.
2. Uses the Simulation-Sound NIZK to generate a proof that those two ciphertexts C_1 and C_2 encrypt the same message with respect to the encryption keys.
3. The CCA ciphertext then consists of the two ciphertexts and this proof.

To decrypt the message, one simply has to check the validity of the proof and to decrypt one of the CPA encryptions.

Instantiations The solution presented in [30] uses Linear Encryption [12] for the CPA-encryption. Our SS-NIZK construction works on bilinear groups, so is also compatible with this encryption scheme.

The overall size of the CCA-Encryption is 6 group elements for the two encryptions, 2 for the verification key and the one-time signature, and several elements for the OR proof. The OR proof needs 4 commitments and 5 linear multiscalar multiplications proof to handle the equality of ciphertexts, an extra commitment for the *OR*, and a commitment and proof of validity of the signature.

The signature and its proof of validity are the larger part of the encryption, and as explained before our construction for that is at least 3 times more efficient than the original one. So our CCA-encryption inherits this efficiency and is nearly 3 times more efficient than theirs while our construction is still tight.

[6] A structure-preserving encryption scheme has public keys, messages, and ciphertexts that consist entirely of group elements, and both the encryption and decryption algorithms perform only group operations.