Linearly Homomorphic Structure-Preserving Signatures and Their Applications

Benoît Libert[1], Thomas Peters[2⋆], Marc Joye[1], and Moti Yung[3]

[1] Technicolor (France)
[2] Université catholique de Louvain, Crypto Group (Belgium)
[3] Google Inc. and Columbia University (USA)

Abstract. Structure-preserving signatures (SPS) are signature schemes where messages, signatures and public keys all consist of elements of a group over which a bilinear map is efficiently computable. This property makes them useful in cryptographic protocols as they nicely compose with other algebraic tools (like the celebrated Groth-Sahai proof systems). In this paper, we consider SPS systems with homomorphic properties and suggest applications that have not been provided before (in particular, not by employing ordinary SPS). We build linearly homomorphic structure-preserving signatures under simple assumptions and show that the primitive makes it possible to verify the calculations performed by a server on outsourced encrypted data (*i.e.*, combining secure computation and authenticated computation to allow reliable and secure cloud storage and computation, while freeing the client from retaining cleartext storage). Then, we give a generic construction of non-malleable (and actually simulation-sound) commitment from any linearly homomorphic SPS. This notably provides the first constant-size non-malleable commitment to group elements.

Keywords: Structure-preserving cryptography, signatures, homomorphism, commitment schemes, non-malleability.

1 Introduction

Composability is an important cryptographic design notion for building systems and protocols. Inside protocols, cryptographic tools need to compose well with each other in order to be used in combination. Structure-preserving cryptography [3], in turn, is a recent paradigm that takes care of composing algebraic tools, and primarily within groups supporting bilinear maps to allow smooth composition with the Groth-Sahai proof systems [47]. The notion allows for modular and simplified designs of various cryptographic protocols and primitives. In the last three years, a large body of work has analyzed the feasibility and the efficiency of structure-preserving signatures (SPS) [45, 28, 38, 1, 3, 4, 20, 29, 51, 5, 6], public-key encryption [21] and commitments schemes [48, 2].

In this paper, we consider SPS schemes with linearly homomorphic properties and argue that such primitives have many applications, even independently of Groth-Sahai proofs. Let us next review our results and then review related work.

1.1 Our Contributions

Linearly Homomorphic Structure-Preserving Signatures. In this paper, we put forth the notion of linearly homomorphic structure-preserving signatures (linearly homomorphic signatures and structure-preserving signatures have been defined before, as we review in the sequel, but the combination of the earlier notions is useful and non-trivial). These signature schemes function exactly like ordinary homomorphic signatures with the additional restriction that signatures and messages only consist of (vectors of) group elements whose discrete logarithms may not be available. We describe three constructions and prove their security under established complexity assumptions in symmetric bilinear groups.

⋆ This author was supported by the CAMUS Walloon Region Project.

Applications. As in all SPS systems, the structure-preserving property makes it possible to efficiently prove knowledge of a homomorphic signature on a committed vector. However, as indicated above, we describe applications of linearly homomorphic SPS beyond their compatibility with the Groth-Sahai techniques.

First, we show that the primitive enables verifiable computation mechanisms on encrypted data.[4] Specifically, it allows a client to store encrypted files on an untrusted remote server. While the dataset is encrypted using an additively homomorphic encryption scheme, the server is able to blindly compute linear functions on the original data and provide the client with a short homomorphically derived signature vouching for the correctness of the computation. This is achieved by having the client sign each ciphertext using a homomorphic SPS scheme and handing the resulting signatures to the server at the beginning. After this initial phase, the client only needs to store a short piece of information, no matter how large the file is. Still, he remains able to authenticate linear functions on his data and the whole process is completely non-interactive. The method extends when datasets are encrypted using a CCA1-secure encryption schemes. Indeed, we will observe that linearly homomorphic SPS schemes yield simple homomorphic IND-CCA1-secure cryptosystems with publicly verifiable ciphertexts.

As a second and perhaps more surprising application, we show that linearly homomorphic SPS schemes generically yield non-malleable [35] trapdoor commitments to group elements. We actually construct a simulation-sound trapdoor commitment [40] — a primitive known (by [40, 54]) to imply re-usable non-malleable commitments with respect to opening [31] — from any linearly homomorphic SPS satisfying a relatively mild condition. To our knowledge, we thus obtain the first constant-size trapdoor commitments to group elements providing re-usable non-malleability with respect to opening. Previous non-interactive commitments to group elements were either malleable [47, 48] or inherently length-increasing [36]: if we disregard the trivial solution consisting of hashing the message first (which is not an option when we want to allow for efficient proofs of knowledge of an opening), no general technique has been known, to date, for committing to many group elements at once using a short commitment string.

In the structure-preserving case, our transformation is purely generic as it applies to a template which any linearly homomorphic SPS necessarily satisfies in symmetric bilinear groups. We also generalize the construction so as to build simulation-sound trapdoor commitments to vectors from any pairing-based (non-structure-preserving) linearly homomorphic signature. In this case, the conversion is only semi-generic as it imposes conditions which are only met by pairing-based systems for the time being: essentially, we need the underlying signature scheme to operate over groups of finite, public order. While only partially generic, this construction of non-malleable commitments from linearly homomorphic signatures is somewhat unexpected considering that the terms "non-malleability" and "homomorphism" are antagonistic, and thus may be considered incompatible.

Techniques and ideas. At first, the very name of our primitive may sound almost self-contradictory when it comes to formally define its security. Indeed, the security of a linearly homomorphic scheme [17] notably requires that it be infeasible to publicly compute a signature on a vector outside the linear span of originally signed vectors. The problem is that, when vector entries live in a discrete-logarithm hard group, deciding whether several vectors are independent or not is believed to be a hard problem. Yet, this will not prevent us from applying new techniques and constructing schemes with security proofs under simple assumptions and the reduction will be able to detect when the adversary has won by simply solving the problem instance it received as input.

Our first scheme's starting point is the one-time (regular) SPS scheme of Abe *et al.* [1]. By removing certain public key components, we obtain the desired linear homomorphism, and prove the security using information-theoretic arguments as in [1]. The key observation here is that, as long as the adversary does not output a signature on a linear combination of previously signed vectors,

[4] Our goals are very different from those of [42], where verifiable computation on homomorphically encrypted data is also considered. We do not seek to outsource computation but rather save the client from storing large datasets.

it will be unable to sign its target vector in the same way as the reduction would, because certain private key components will remain perfectly hidden.

Our initial scheme inherits the one-time restriction of the scheme in [1] in that only one linear subspace can be safely signed with a given public key. Nevertheless, we can extend it to build a full linearly homomorphic SPS system. To this end, we suitably combine our first scheme with Waters signatures [60]. Here, Waters signatures are used as a resting ground for fresh random exponents which are introduced in each signed vector and help us refresh the state of the system and apply each time the same argument as in the one-time scheme. We also present techniques to turn the scheme into a fully randomizable one, where a derived signature has the same distribution as a directly signed message.

In our simulation-sound commitments to group elements, the commitment generation technique appeals to the verification algorithm of the signature scheme, and proceeds by evaluating the corresponding pairing-product equations on the message, but using random group elements instead of actual signatures. The binding and simulation-binding properties, in turn, stem from the infeasibility of forging signatures while the signature homomorphism allows equivocating fake commitments when simulating the view of an adversary. It was already known how to build simulation-sound and non-malleable commitments [40, 54, 31, 41, 24] from signature schemes with efficient Σ protocols. Our method is, in fact, different and immediately yields length-reducing structure-preserving commitments to vectors without using Σ protocols.

1.2 Related Work

Structure-Preserving Signatures. Signature schemes where messages only consist of group elements appeared for the first time — without the "structure-preserving" terminology — as ingredients of Groth's construction [45] of group signatures in the standard model. The scheme of [45] was mostly a proof of concept, with signatures consisting of thousands of group elements. More efficient realizations were given by Cathalo, Libert and Yung [28] and Fuchsbauer [38]. Abe, Haralambiev and Ohkubo [1, 3] subsequently showed how to sign messages of n group elements at once using $O(1)$-size signatures. Lower bounds on the size of structure-preserving signatures were given in [4] while Abe *et al.* [7] provided evidence that optimally short SPS necessarily rely on interactive assumptions. As an ingredient for their tightly secure cryptosystems, Hofheinz and Jager [51] gave constructions based on the Decision Linear assumption [16] while similar results were independently achieved in [20, 29]. Quite recently, Abe *et al.* [5, 6] obtained constant-size signatures without sacrificing the security guarantees offered by security proofs under simple assumptions.

Regarding primitives beyond signature schemes, Camenisch *et al.* [21] showed a structure-preserving variant of the Cramer-Shoup cryptosystem [30] and used it to implement oblivious third parties [22]. Groth [48] described length-reducing trapdoor commitments (*i.e.*, where the commitment is shorter than the committed message) to group elements whereas [2] showed the impossibility of realizing such commitments when the commitment string lives in the same group as the message. Sakai *et al.* [58] recently suggested to use structure-preserving identity-based encryption [59] systems to restrict the power of the opening authority in group signatures.

Linearly Homomorphic Signatures. The concept of homomorphic signatures can be traced back to Desmedt [33] while proper definitions remained lacking until the work of Johnson *et al.* [53]. Since then, constructions have appeared for various kinds of homomorphisms (see [8] and references therein).

Linearly homomorphic signatures are an important class of homomorphic signatures for arithmetic functions, whose study was initiated by Boneh, Freeman, Katz and Waters [17]. While initially motivated by applications to network coding [17], they are also useful in proofs of storage [9, 10] or in verifiable computation mechanisms, when it comes to authenticate servers' computations on out-

sourced data (see, *e.g.*, [8]). The recent years, much attention was given to the notion and a variety of constructions [43, 11, 18, 19, 26, 27, 37, 12, 13] based on various assumptions have been studied.

1.3 Organization

Section 2 first gives security definitions for linearly homomorphic SPS systems, for which efficient constructions are provided in Section 3. Their applications to verifiable computation on encrypted data are explained in Section 4 while Section 5 shows how to build simulation-sound commitments to group elements. Implications and generalizations of the latter are then given in Appendix E.

2 Background

2.1 Definitions for Linearly Homomorphic Signatures

Let $(\mathbb{G}, \mathbb{G}_T)$ be a configuration of (multiplicatively written) groups of prime order p over which a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is efficiently computable.

Following [1, 3], we say that a signature scheme is *structure-preserving* if messages, signature components and public keys live in the group $\mathbb{G}$.

We consider linearly homomorphic signatures for which the message space $\mathcal{M}$ consists of pairs $\mathcal{M} := \mathcal{T} \times \mathbb{G}^n$, for some $n \in \mathbb{N}$, where $\mathcal{T}$ is a tag space. We remark that, in the applications considered in this paper, tags do not need to be group elements. We thus allow them to be arbitrary strings.

Definition 1. *A linearly homomorphic structure-preserving signature scheme over $(\mathbb{G}, \mathbb{G}_T)$ consists of a tuple of efficient algorithms $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ for which the message space is $\mathcal{M} := \mathcal{T} \times \mathbb{G}^n$, for some $n \in \mathsf{poly}(\lambda)$ and some set $\mathcal{T}$, and with the following specifications.*

$\mathsf{Keygen}(\lambda, n)$: *is a randomized algorithm that takes in a security parameter $\lambda \in \mathbb{N}$ and an integer $n \in \mathsf{poly}(\lambda)$ denoting the dimension of vectors to be signed. It outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ and the description of a tag (i.e., a file identifier) space $\mathcal{T}$.*

$\mathsf{Sign}(\mathsf{sk}, \tau, \vec{M})$: *is a possibly probabilistic algorithm that takes as input a private key sk, a file identifier $\tau \in \mathcal{T}$ and a vector $\vec{M} \in \mathbb{G}^n$. It outputs a signature $\sigma \in \mathbb{G}^{n_s}$, for some $n_s \in \mathsf{poly}(\lambda)$.*

$\mathsf{SignDerive}(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell})$: *is a (possibly probabilistic) signature derivation algorithm. It takes as input a public key pk, a file identifier τ as well as ℓ pairs $(\omega_i, \sigma^{(i)})$, each of which consists of a weight $\omega_i \in \mathbb{Z}_p$ and a signature $\sigma^{(i)} \in \mathbb{G}^{n_s}$. The output is a signature $\sigma \in \mathbb{G}^{n_s}$ on the vector $\vec{M} = \prod_{i=1}^{\ell} \vec{M}_i{}^{\omega_i}$, where $\sigma^{(i)}$ is a signature on $\vec{M}_i$.*

$\mathsf{Verify}(\mathsf{pk}, \tau, \vec{M}, \sigma)$: *is a deterministic algorithm that takes in a public key pk, a file identifier $\tau \in \mathcal{T}$, a signature σ and a vector $\vec{M}$. It outputs 1 if σ is deemed valid and 0 otherwise.*

Correctness is expressed by imposing that, for all security parameters $\lambda \in \mathbb{N}$, all integers $n \in \mathsf{poly}(\lambda)$ and all triples $(\mathsf{pk}, \mathsf{sk}, \mathcal{T}) \leftarrow \mathsf{Keygen}(\lambda, n)$, the following holds:

1. For all $\tau \in \mathcal{T}$ and all n-vectors $\vec{M}$, if $\sigma = \mathsf{Sign}(\mathsf{sk}, \tau, \vec{M})$, then we have $\mathsf{Verify}(\mathsf{pk}, \tau, \vec{M}, \sigma) = 1$.
2. For all $\tau \in \mathcal{T}$, any $\ell > 0$ and any set of triples $\{(\omega_i, \sigma^{(i)}, \vec{M}_i)\}_{i=1}^{\ell}$, if $\mathsf{Verify}(\mathsf{pk}, \tau, \vec{M}_i, \sigma^{(i)}) = 1$ for each $i \in \{1, \ldots, \ell\}$, then $\mathsf{Verify}\big(\mathsf{pk}, \tau, \prod_{i=1}^{\ell} \vec{M}_i{}^{\omega_i}, \mathsf{SignDerive}(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell})\big) = 1$.

Security. In linearly homomorphic signatures, we use the same definition of unforgeability as in [12]. This definition implies security in the stronger model used by Freeman [37] since the adversary can interleave signing queries for individual vectors belonging to distinct subspaces. Moreover, file identifiers can be chosen by the adversary (which strengthens the definition of [17]) and are not assumed to be uniformly distributed. As a result, a file identifier can be a low-entropy, easy-to-remember string such as the name of the dataset's owner.

Definition 2. *A linearly homomorphic SPS scheme* $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ *is secure if no PPT adversary has non-negligible advantage in the game below:*

1. *The adversary* $\mathcal{A}$ *chooses an integer* $n \in \mathbb{N}$ *and sends it to the challenger who runs* $\mathsf{Keygen}(\lambda, n)$ *and obtains* $(\mathsf{pk}, \mathsf{sk})$ *before sending* pk *to* $\mathcal{A}$.
2. *On polynomially-many occasions,* $\mathcal{A}$ *can interleave the following kinds of queries.*
   - *Signing queries:* $\mathcal{A}$ *chooses a tag* $\tau \in \mathcal{T}$ *and a vector* $\vec{M} \in \mathbb{G}^n$. *The challenger picks a handle* h *and computes* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, \vec{M})$. *It stores* $(\mathsf{h}, (\tau, \vec{M}, \sigma))$ *in a table* T *and returns* h.
   - *Derivation queries:* $\mathcal{A}$ *chooses a vector of handles* $\vec{\mathsf{h}} = (\mathsf{h}_1, \ldots, \mathsf{h}_k)$ *and a set of coefficients* $\{\omega_i\}_{i=1}^k$. *The challenger retrieves the tuples* $\{(\mathsf{h}_i, (\tau, \vec{M}_i), \sigma^{(i)})\}_{i=1}^k$ *from* T *and returns* $\perp$ *if one of these does not exist or if there exists* $i \in \{1, \ldots, k\}$ *such that* $\tau_i \neq \tau$. *Otherwise, it computes* $\vec{M} = \prod_{i=1}^k \vec{M}_i^{\omega_i}$ *and runs* $\sigma' \leftarrow \mathsf{SignDerive}\big(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^k\big)$. *It also chooses a handle* h', *stores* $(\mathsf{h}', (\tau, \vec{M}), \sigma')$ *in* T *and returns* h' *to* $\mathcal{A}$.
   - *Reveal queries:* $\mathcal{A}$ *chooses a handle* h. *If no tuple of the form* $(\mathsf{h}, (\tau, \vec{M}), \sigma')$ *exists in* T, *the challenger returns* $\perp$. *Otherwise, it returns* σ' *to* $\mathcal{A}$ *and adds* $((\tau, \vec{M}), \sigma')$ *to the set* Q.
3. $\mathcal{A}$ *outputs an identifier* $\tau^\star$, *a signature* $\sigma^\star$ *and a vector* $\vec{M}^\star \in \mathbb{G}^n$. *The adversary* $\mathcal{A}$ *wins if* $\mathsf{Verify}(\mathsf{pk}, \tau^\star, \vec{M}^\star, \sigma^\star) = 1$ *and one of the conditions below is satisfied:*
   - *(Type I):* $\tau^\star \neq \tau_i$ *for any entry* $(\vec{\tau}_i, .)$ *in* Q *and* $\vec{M}^\star \neq (1_\mathbb{G}, \ldots, 1_\mathbb{G})$.
   - *(Type II):* $\tau^\star = \tau_i$ *for* $k_i > 0$ *entries* $(\tau_i, .)$ *in* Q *and* $\vec{M}^\star \notin V_i$, *where* V_i *denotes the subspace spanned by all vectors* $\vec{M}_1, \ldots, \vec{M}_{k_i}$ *for which an entry of the form* $(\tau^\star, \vec{M}_j)$, *with* $j \in \{1, \ldots, k_i\}$, *appears in* Q.

$\mathcal{A}$*'s advantage is its probability of success taken over all coin tosses.*

In our first scheme, we will consider a weaker notion of *one-time* security. In this notion, the adversary is limited to obtain signatures for only *one* linear subspace. In this case, there is no need for file identifiers and we assume that all vectors are assigned the identifier $\tau = \varepsilon$.

In the following, the adversary will be said *independent* if

- For any given tag τ, it is restricted to only query signatures on linearly independent vectors.
- Each vector is only queried at most once.

Non-independent adversaries are not subject to the above restrictions. It will be necessary to consider these adversaries in our construction of non-malleable commitments. Nevertheless, security against independent adversaries suffices for many applications — including encrypted cloud storage — since the signer can always append unit vectors to each newly signed vector.

At first, one may wonder how Definition 2 can be satisfied at all given that the challenger may not have an efficient way to check whether the adversary is successful. Indeed, in cryptographically useful discrete-logarithm-hard groups $\mathbb{G}$, deciding whether vectors $\{\vec{M}_i\}_i$ of $\mathbb{G}^n$ are linearly dependent is believed to be difficult when $n > 2$. However, it may be possible using some trapdoor information embedded in pk, especially if the adversary additionally outputs signatures on $\{\vec{M}_i\}_i$.

2.2 Hardness Assumptions

We rely on the following hardness assumptions, the first of which implies the second one.

Definition 3 ([16]). *The* Decision Linear Problem *(DLIN) in* $\mathbb{G}$, *is to distinguish the distributions* $(g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ *and* $(g^a, g^b, g^{ac}, g^{bd}, g^z)$, *with* $a, b, c, d \xleftarrow{R} \mathbb{Z}_p^*$, $z \xleftarrow{R} \mathbb{Z}_p^*$. *The* Decision Linear Assumption *is the intractability of DLIN for any PPT distinguisher* $\mathcal{D}$.

Definition 4. *The* Simultaneous Double Pairing problem *(SDP) in* $(\mathbb{G}, \mathbb{G}_T)$ *is, given a tuple of elements* $(g_z, g_r, h_z, h_u) \in_R \mathbb{G}^4$, *to find a non-trivial triple* $(z, r, u) \in \mathbb{G}^3 \backslash \{(1_\mathbb{G}, 1_\mathbb{G}, 1_\mathbb{G})\}$ *such that* $e(g_z, z) \cdot e(g_r, r) = 1_{\mathbb{G}_T}$ *and* $e(h_z, z) \cdot e(h_u, u) = 1_{\mathbb{G}_T}$.

3 Constructions of Linearly Homomorphic Structure-Preserving Signatures

As a warm-up, we begin by describing a one-time homomorphic signature, where a given public key allows signing only *one* linear subspace.

3.1 A One-Time Linearly Homomorphic Construction

In the description hereunder, since only one linear subspace can be signed for each public key, no file identifier τ is used. We thus set τ to be the empty string ε in all algorithms.

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of the subspace to be signed, choose bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, choose generators $h, g_z, g_r, h_z \xleftarrow{R} \mathbb{G}$. Pick $\chi_i, \gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$, for $i = 1$ to n. Then, for each $i \in \{1, \ldots, n\}$, compute $g_i = g_z^{\chi_i} g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} h^{\delta_i}$. The private key is $\mathsf{sk} = \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n$ while the public key is defined to be

$$\mathsf{pk} = \big(g_z,\ h_r,\ h_z,\ h,\ \{g_i, h_i\}_{i=1}^n\big) \in \mathbb{G}^{2n+4}.$$

Sign($\mathsf{sk}, \tau, (M_1, \ldots, M_n)$): to sign a vector $(M_1, \ldots, M_n) \in \mathbb{G}^n$ associated with the identifier $\tau = \varepsilon$ using $sk = \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n$, compute the signature consists of $\sigma = (z, r, u) \in \mathbb{G}^3$, where

$$z = \prod_{i=1}^n M_i^{-\chi_i}, \qquad r = \prod_{i=1}^n M_i^{-\gamma_i}, \qquad u = \prod_{i=1}^n M_i^{-\delta_i}.$$

SignDerive($\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): given the public key pk, a file identifier $\tau = \varepsilon$ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parse each signature $\sigma^{(i)}$ as $\sigma^{(i)} = \big(z_i, r_i, u_i\big) \in \mathbb{G}^3$ for $i = 1$ to ℓ. Compute and return the derived signature $\sigma = (z, r, u) = \big(\prod_{i=1}^\ell z_i^{\omega_i}, \prod_{i=1}^\ell r_i^{\omega_i}, \prod_{i=1}^\ell u_i^{\omega_i}\big)$.

Verify($\mathsf{pk}, \sigma, \tau, (M_1, \ldots, M_n)$): given a signature $\sigma = (z, r, u) \in \mathbb{G}^3$, a vector $(M_1, \ldots, M_n)$ and a file identifier $\tau = \varepsilon$, return 1 if and only if $(M_1, \ldots, M_n) \neq (1_\mathbb{G}, \ldots, 1_\mathbb{G})$ and (z, r, u) satisfy

$$1_{\mathbb{G}_T} = e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, M_i), \qquad 1_{\mathbb{G}_T} = e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^n e(h_i, M_i).$$

The proof of security relies on the fact that, while the signing algorithm is deterministic, signatures are not unique. However, the reduction will be able to compute exactly one signature for each vector. At the same time, an adversary has no information about which specific signature the legitimate signer would compute on a vector outside the span of already signed vectors. Moreover, by obtaining two distinct signatures on a given vector, the reduction can solve a given SDP instance.

Theorem 1. *The scheme is unforgeable if the SDP assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$.

Proof. We describe an algorithm $\mathcal{B}$ that takes as input a SDP instance $(g_z, g_r, h_z, h) \in \mathbb{G}^4$ and uses a forger $\mathcal{A}$ to find a triple (z, r, u) such that $e(g_z, z) \cdot e(g_r, r) = e(h_z, z) \cdot e(h, u) = 1_{\mathbb{G}_T}$.

To this end, $\mathcal{B}$ honestly runs the key generation algorithm using randomly chosen $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$. Whenever $\mathcal{A}$ requests a signature on a vector $(M_1, \ldots, M_n) \in \mathbb{G}^n$, $\mathcal{B}$ faithfully follows the specification of the signing algorithm. The game ends with the adversary $\mathcal{A}$ outputting a vector $(M_1^\star, \ldots, M_n^\star)$ with a valid signature $(z^\star, r^\star, u^\star)$. At this point, $\mathcal{B}$ computes its own signature

$$(z^\dagger, r^\dagger, u^\dagger) = (\prod_{i=1}^n M_i^{\star -\chi_i}, \prod_{i=1}^n M_i^{\star -\gamma_i}, \prod_{i=1}^n M_i^{\star -\delta_i}) \tag{1}$$

on $(M_1^\star, \ldots, M_n^\star)$. We claim that, with overwhelming probability, $(z^\ddagger, r^\ddagger, u^\ddagger) = (z^\star / z^\dagger, r^\star / r^\dagger, u^\star / u^\dagger)$ is a non-trivial solution to the SDP instance.

To see this, we first note that a given public key has exponentially many corresponding private keys and pk perfectly hides the vector $(\chi_1, \ldots, \chi_n)$. Moreover, for a given pk, each message $(M_1, \ldots, M_n)$ has an exponential number of valid signatures but the one produced by the signing algorithm is completely determined by $(\chi_1, \ldots, \chi_n)$. We will see that, in $\mathcal{A}$'s view, guessing the value $z^\dagger$ of (1) amounts to inferring which vector $(\chi_1, \ldots, \chi_n)$ the reduction $\mathcal{B}$ is using.

Throughout the game, $\mathcal{A}$ obtains signatures $\{(z_i, r_i, u_i)\}_{i=1}^{n-1}$ on at most $n-1$ linearly independent vectors of $\mathbb{G}^n$. If we consider discrete logarithms, these signatures only provide $\mathcal{A}$ with $n-1$ linearly independent equations because, for each triple (z_i, r_i, u_i), z_i uniquely determines (r_i, u_i). Taking into account the information revealed by $\{(g_i, h_i)\}_{i=1}^{n}$, we find that an unbounded adversary is presented with $3n-1$ linear equations in $3n$ unknowns. In $\mathcal{A}$'s view, since $(M_1^\star, \ldots, M_n^\star)$ must be independent of previously signed vectors, predicting $z^\dagger$ is only possible with probability $1/p$. With probability $1 - 1/p$, we thus have $z^\dagger \neq z^\star$, in which case $(z^\ddagger, r^\ddagger, u^\ddagger)$ solves the SDP instance because $(z^\dagger, r^\dagger, u^\dagger)$ and $(z^\star, r^\star, u^\star)$ both satisfy the verification equations. □

The scheme can be modified so as to work in asymmetric pairing configurations and the Double Pairing assumption [1]. However, we need to work with the SDP assumption in the next section.

3.2 A Full-Fledged Linearly Homomorphic SPS Scheme

Here, we upgrade our one-time construction to obtain a scheme allowing us to sign an arbitrary number of linear subspaces. Here, each file identifier τ consists of a L-bit string. The construction builds on the observation that, in the scheme of Section 3.1, signatures (z, r, u) could be re-randomized by computing $(z \cdot g_r^\theta, r \cdot g_z^{-\theta}, u \cdot h_z^{-\log_h(g_r)\cdot\theta})$, with $\theta \xleftarrow{R} \mathbb{Z}_p$, if $h_z^{-\log_h(g_r)}$ were available. Since publicizing $h_z^{-\log_h(g_r)}$ would render the scheme insecure, our idea is to use Waters signatures as a support for introducing extra randomizers in the exponent.

In the construction, the u component of each signature can be seen as an aggregation of the one-time signature of Section 3.1 with a Waters signature $(h_z^{\log_h(g_r)} \cdot H_{\mathbb{G}}(\tau)^{-\rho}, h^\rho)$ [60] on the tag τ.

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of the subspace to be signed, choose bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, conduct the following steps.

1. Choose $h \xleftarrow{R} \mathbb{G}$ and $\alpha_z, \alpha_r, \beta_z \xleftarrow{R} \mathbb{Z}_p$. Define $g_z = h^{\alpha_z}$, $g_r = h^{\alpha_r}$ and $h_z = h^{\beta_z}$.
2. For $i = 1$ to n, pick $\chi_i, \gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$ and compute $g_i = g_z^{\chi_i} g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} h^{\delta_i}$.
3. Choose a random vector $\overline{\mathbf{w}} = (w_0, w_1, \ldots, w_L) \xleftarrow{R} \mathbb{G}^{L+1}$. The latter defines a hash function $H_{\mathbb{G}} : \{0,1\}^L \to \mathbb{G}$ which maps $\tau = \tau[1] \ldots \tau[L] \in \{0,1\}^L$ to $H_{\mathbb{G}}(\tau) = w_0 \cdot \prod_{k=1}^{L} w_k^{\tau[k]}$.

The private key is $\mathsf{sk} = \big(h_z^{\alpha_r},\ \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n\big)$ while the public key consists of

$$\mathsf{pk} = \big(g_z,\ g_r,\ h_z,\ h,\ \{g_i, h_i\}_{i=1}^n,\ \overline{\mathbf{w}}\big) \in \mathbb{G}^{2n+4} \times \mathbb{G}^{L+1}.$$

Sign($\mathsf{sk}, \tau, (M_1, \ldots, M_n)$): to sign a vector $(M_1, \ldots, M_n) \in \mathbb{G}^n$ w.r.t. the file identifier τ using $sk = \big(h_z^{\alpha_r},\ \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n\big)$, choose $\theta, \rho \xleftarrow{R} \mathbb{Z}_p$ and output $\sigma = (z, r, u, v) \in \mathbb{G}^4$, where

$$z = g_r^\theta \cdot \prod_{i=1}^{n} M_i^{-\chi_i} \qquad r = g_z^{-\theta} \cdot \prod_{i=1}^{n} M_i^{-\gamma_i}$$

$$u = (h_z^{\alpha_r})^{-\theta} \cdot \prod_{i=1}^{n} M_i^{-\delta_i} \cdot H_{\mathbb{G}}(\tau)^{-\rho} \qquad v = h^\rho$$

SignDerive($\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell}$): given pk, a file identifier τ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parse $\sigma^{(i)}$ as $\sigma^{(i)} = \big(z_i, r_i, u_i, v_i\big) \in \mathbb{G}^4$ for $i = 1$ to ℓ. Then, choose $\rho' \xleftarrow{R} \mathbb{Z}_p$ and compute and return $\sigma = (z, r, u, v)$, where $z = \prod_{i=1}^{\ell} z_i^{\omega_i}$, $r = \prod_{i=1}^{\ell} r_i^{\omega_i}$, $u = \prod_{i=1}^{\ell} u_i^{\omega_i} \cdot H_{\mathbb{G}}(\tau)^{-\rho'}$ and $v = \prod_{i=1}^{\ell} v_i^{\omega_i} \cdot h^{\rho'}$.

Verify($\mathsf{pk}, \sigma, \tau, (M_1, \ldots, M_n)$): given a signature $\sigma = (z, r, u, v) \in \mathbb{G}^4$, a file identifier τ and a vector $(M_1, \ldots, M_n)$, return 1 if and only if $(M_1, \ldots, M_n) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and (z, r, u, v) satisfy

$$1_{\mathbb{G}_T} = e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^{n} e(g_i, M_i), \qquad 1_{\mathbb{G}_T} = e(h_z, z) \cdot e(h, u) \cdot e(H_{\mathbb{G}}(\tau), v) \cdot \prod_{i=1}^{n} e(h_i, M_i). \quad (2)$$

The security of the scheme against *non-independent* Type I adversaries is proved under the SDP assumption. In the case of Type II forgeries, we need to assume the adversary to be independent because, at some point, the simulator is only able to compute a signature for a unique value[5] of θ.

Theorem 2. *The scheme is unforgeable against independent adversaries if the SDP assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Moreover, the scheme is secure against non-independent Type I adversaries.*

Proof. The result is proved by separately considering Type I and Type II forgeries. For simplicity, we first consider Type II adversaries as the case of Type I attacks will be simpler. Lemmas 1 and 2 show how to build an algorithm solving the SDP problem either way. □

The proof of Lemma 1 uses Waters signatures as a handle to randomize signatures. Specifically, whenever the reduction is able to compute a Waters signatures $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\rho}, h^{\rho})$ on the tag τ, it can inject a fresh extra randomizer $\theta \in \mathbb{Z}_p$ in the exponent for each vector associated with τ. By doing so, with non-negligible probability, the specific vector $(\chi_1, \ldots, \chi_n)$ used by the reduction will remain completely undetermined from $\mathcal{A}$'s view.

Lemma 1. *For any Type II independent forger $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ solving the SDP problem such that $\mathbf{Adv}(\mathcal{A}) \leq 8 \cdot q \cdot (L+1) \cdot \left(\mathbf{Adv}^{\mathrm{SDP}}(\mathcal{B}) + \frac{1}{p}\right)$, where q is the number of distinct tags appearing in signing queries.* (The proof is given in Appendix A.1).

Lemma 2. *A Type I forger $\mathcal{A}$ implies an algorithm $\mathcal{B}$ solving the SDP problem with non-negligible advantage. More precisely, we have $\mathbf{Adv}(\mathcal{A}) \leq 8 \cdot q \cdot (L+1) \cdot \left(\mathbf{Adv}^{\mathrm{SDP}}(\mathcal{B}) + \frac{1}{p}\right)$, where q is the number of distinct tags occurring in signing queries. Moreover, the statement holds even for non-independent adversaries.* (The proof is given in Appendix A.2).

Since the signature component u cannot be publicly randomized, the scheme does not have fully randomizable signatures. In Appendix B, we describe a fully randomizable variant. In applications like non-malleable commitments to group elements, the above scheme is sufficient however.

4 Applications

4.1 Verifiable Computation for Encrypted Cloud Storage

Linearly homomorphic schemes are known (see, *e.g.*, [8]) to provide verifiable computation mechanisms for outsourced data. Suppose that a user has a dataset consisting of n samples $s_1, \ldots, s_n \in \mathbb{Z}_p$. The dataset can be encoded as vectors $\vec{v}_i = (\vec{e}_i | s_i) \in \mathbb{Z}_p^{n+1}$, where $\vec{e}_i \in \mathbb{Z}_p^n$ denotes the i-th unit vector for each $i \in \{1, \ldots, n\}$. The user then assigns a file identifier τ to $\{\vec{v}_i\}_{i=1}^n$, computes signatures $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, \vec{v}_i)$ on the resulting vectors and stores $\{(\vec{v}_i, \sigma_i)\}_{i=1}^n$ at the server. When requested, the server can then evaluate a sum $s = \sum_{i=1}^n s_i$ and provide evidence that the latter computation is correct by deriving a signature on the vector $(1, 1, \ldots, 1, s) \in \mathbb{Z}_p^{n+1}$. Unless the server is able to forge a signature for a vector outside the span of $\{\vec{v}_i\}_{i=1}^n$, it is unable to fool the user. The above method readily extends to authenticate weighted sums or Fourier transforms.

[5] Note that this is not a problem since the signer can derive θ as a pseudorandom function of τ and $(M_1, \ldots, M_n)$ to make sure that a given vector is always signed using the same θ.

One disadvantage of the above method is that it requires the server to retain the dataset $\{s_i\}_{i=1}^n$ in the clear. Using linearly homomorphic structure-preserving signatures, the user can apply the above technique on encrypted samples using the Boneh-Boyen-Shacham (BBS) cryptosystem [16].

The BBS cryptosystem involves a public key $(g, \tilde{g}, f = g^x, h = g^y) \in_R \mathbb{G}^4$, where $(x, y) \in \mathbb{Z}_p^2$ is the private key. The user (or anyone else knowing his public key) can first encrypt his samples $\{s_i\}_{i=1}^n$ by computing BBS encryptions $(C_{1,i}, C_{2,i}, C_{3,i}) = (f^{r_i}, h^{t_i}, \tilde{g}^{s_i} \cdot g^{r_i+t_i})$, with $r_i, t_i \xleftarrow{R} \mathbb{Z}_p$, for each $i \in \{1, \ldots, n\}$. If the user holds a linearly homomorphic structure preserving signature key pair for vectors of dimension $n+3$, he can generate n structure preserving signatures on vectors $((C_{1,i}, C_{2,i}, C_{3,i})|\vec{E}_i) \in \mathbb{G}^{n+3}$, where $\vec{E}_i = (1_\mathbb{G}, \ldots, 1_\mathbb{G}, g, 1_\mathbb{G}, \ldots, 1_\mathbb{G}) = g^{\vec{e}_i}$ for each $i \in \{1, \ldots, n\}$, using the scheme of Section 3.2. The vectors $\{((C_{1,i}, C_{2,i}, C_{3,i})|\vec{E}_i)\}_{i=1}^n$ and their signatures $\{(z_i, r_i, u_i, v_i)\}_{i=1}^n$ are then archived in the cloud in such a way that the server can publicly derive a signature on the vector $\left(f^{\sum_i r_i}, h^{\sum_i t_i}, \tilde{g}^{\sum_i s_i} \cdot g^{\sum_i (r_i+t_i)}, g, g, \ldots, g\right) \in \mathbb{G}^{n+3}$ in order to convince the client that the encrypted sum was correctly computed. Using his private key (x, y), the client can then retrieve the sum $\sum_i s_i$ as long as it remains in a sufficiently small range.

The interest of the above solution lies in that the client can dispense with the need for storing the $O(n)$-size public key of his linearly homomorphic signature. Indeed, he can simply retain the random seed that was used to generate pk and re-compute private key elements $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$ whenever he wants to verify the server's response. In this case, the verification equations (2) become

$$1_{\mathbb{G}_T} = e(g_z, z \cdot \prod_{i=1}^n M_i^{\chi_i}) \cdot e(g_r, r \cdot \prod_{i=1}^n M_i^{\gamma_i}) = e(h_z, z \cdot \prod_{i=1}^n M_i^{\chi_i}) \cdot e(h, u \cdot \prod_{i=1}^n M_i^{\delta_i}) \cdot e(H_\mathbb{G}(\tau), v),$$

so that the client only has to compute $O(1)$ pairings. Moreover, the client does not have to determine an upper bound on the size of his dataset when generating his public key. Initially, he only needs to generate $\{(g_j, h_j)\}_{j=1}^3$. When the i-th ciphertext $(C_{1,i}, C_{2,i}, C_{3,i})$ has to be stored, the client derives $(\chi_{i+3}, \gamma_{i+3}, \delta_{i+3})$ and (g_{i+3}, h_{i+3}) by applying a PRF to the index i. This will be sufficient to sign vectors of the form $((C_{1,i}, C_{2,i}, C_{3,i})|\vec{E}_i)$.

In order to hide all partial information about the original dataset, the server may want to re-randomize the derived signature and ciphertext before returning them. This can be achieved by having the client include signatures on the vectors $(f, 1_\mathbb{G}, g, 1_\mathbb{G}, \ldots, 1_\mathbb{G})$, $(1_\mathbb{G}, h, g, 1_\mathbb{G}, \ldots, 1_\mathbb{G})$ in the outsourced dataset. Note that, in this case, the signature should be re-randomized as well. For this reason, our randomizable scheme described in Appendix B should be preferred.

Complete and careful security models for "verifiable computation on encrypted data" are beyond the scope of this paper. Here, they would naturally combine the properties of secure homomorphic encryption and authenticated computing. It should be intuitively clear that a malicious server cannot trick a client into accepting an incorrect result (*i.e.*, one which differs from the actual defined linear function it is supposed to compute over the defined signed ciphertext inputs) without defeating the security of the underlying homomorphic signature.

4.2 Extension to CCA1-Encrypted Data

In the application of Section 4.1, the underlying crypotosystem has to be additively homomorphic, which prevents it from being secure against adaptive chosen-ciphertext attacks. On the other hand, the method is compatible with security against *non-adaptive* chosen ciphertext attacks. One possibility is to apply the "lite" Cramer-Shoup technique (in its variant based on DLIN) as it achieves CCA1-security while remaining homomorphic. Unfortunately, the validity of ciphertexts is not publicly verifiable, which may be annoying in applications like cloud storage or universally verifiable e-voting systems. Indeed, servers may be willing to have guarantees that they are actually storing encryptions of some message instead of random group elements.

Consider the cryptosystem where ciphertexts $(C_1, C_2, C_3, C_4) = (f^r, h^t, g^{r+t}, \tilde{g}^m \cdot X_1^r \cdot X_2^t)$ are decrypted as $m = \log_{\tilde{g}}(C_4 \cdot C_1^{-x_1} C_2^{-x_2} C_3^{-z})$, where $X_1 = f^{x_1} g^z$ and $X_2 = h^{x_2} g^z$ are part of the public

key. In [52], such a system was made chosen-ciphertext secure using a *publicly* verifiable one-time simulation-sound proof that (f, h, g, C_1, C_2, C_3) forms a DLIN tuple. In the security proof, if the reduction is guaranteed not to leak $C_1^{-x_1}C_2^{-x_2}C_3^{-z}$ for an invalid triple (C_1, C_2, C_3) (*i.e.*, as long as the adversary is unable to generate a fake proof for this), the private key component z will remain perfectly hidden. Consequently, if the challenge ciphertext is computed by choosing $C_3^\star \in_R \mathbb{G}$ (so that $(f, h, g, C_1^\star, C_2^\star, C_3^\star)$ is not a DLIN tuple) and computing $C_4^\star = \tilde{g}^m \cdot C_1^{\star x_1} \cdot C_2^{\star x_2} \cdot C_3^{\star z}$, the plaintext m is independent of $\mathcal{A}$'s view. If we replace the one-time simulation-sound proofs by standard proofs of membership in the scheme of [52], we obtain a CCA1 homomorphic encryption scheme. Linearly homomorphic SPS schemes provide a simple and efficient way to do that.

The idea is to include in the public key the verification key of a one-time linearly homomorphic SPS — using the scheme of Section 3.1 — for $n = 3$ as well as signatures on the vectors $(f, 1_\mathbb{G}, g)$, $(1_\mathbb{G}, h, g) \in \mathbb{G}^3$. This will allow the sender to publicly derive a signature (z, r, u) on the vector $(C_1, C_2, C_3) = (f^r, h^t, g^{r+t})$. Each ciphertext thus consists of $(z, r, u, C_1, C_2, C_3, C_4)$. In the security proof, at each pre-challenge decryption query, the signature (z, r, u) serves as publicly verifiable evidence that (f, h, g, C_1, C_2, C_3) is a DLIN tuple. In the challenge phase, the reduction reveals another homomorphic signature $(z^\star, r^\star, u^\star)$ for a vector $(C_1^\star, C_2^\star, C_3^\star)$ that may be outside the span of $(f, 1_\mathbb{G}, g)$ and $(1_\mathbb{G}, h, g)$ but it does not matter since decryption queries are no longer allowed beyond this point.

We note that linearly homomorphic SPS can also be used to construct CCA1-secure homomorphic encryption schemes based on the Naor-Yung paradigm [56] in the standard model.

5 Non-Malleable Trapdoor Commitments to Group Elements from Linearly Homomorphic Structure-Preserving Signatures

As noted in [48, 49], some applications require to commit to group elements without knowing their discrete logarithms or destroying their algebraic structure by hashing them first. This section shows that, under a certain mild condition, linearly homomorphic SPS imply length-reducing non-malleable structure-preserving commitments to vectors of group elements.

As a result, we obtain the first length-reducing non-malleable structure-preserving trapdoor commitment. Our scheme is not *strictly*[6] structure-preserving (according to the terminology of [2]) because the commitment string lives in $\mathbb{G}_T$ rather than $\mathbb{G}$. Still, openings only consist of elements in $\mathbb{G}$, which makes it possible to generate efficient NIWI proofs that committed group elements satisfy certain properties. To our knowledge, the only known non-malleable commitment schemes whose openings only consist of group elements were described by Fischlin *et al.* [36]. However, these constructions cannot be length-reducing as they achieve universal composability [23, 25].

Our schemes are obtained by first constructing simulation-sound trapdoor commitments (SSTC) [40, 54] to group elements. SSTC schemes were first suggested by Garay, MacKenzie and Yang [40] as a tool for constructing universally composable zero-knowledge proofs [23]. MacKenzie and Yang subsequently gave a simplified security definition which suffices to provide non-malleability with respect to opening in the sense of the definition of re-usable non-malleable commitments [31].

In a SSTC, each commitment is labeled with a tag. The definition of [54] requires that, even if the adversary can see equivocations of commitments to possibly distinct messages for several tags $tag_1, \ldots, tag_q$, it will not be able to break the binding property for a new tag $tag \notin \{tag_1, \ldots, tag_q\}$.

Definition 5 ([54]). *A simulation-sound trapdoor commitment* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{FakeCom}, \mathsf{FakeOpen}, \mathsf{Verify})$ *is a tuple where* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Verify})$ *forms a commitment scheme and* $(\mathsf{FakeCom}, \mathsf{FakeOpen})$ *are PPT algorithms with the following properties*

[6] We recall that strictly structure-preserving commitments cannot be length-reducing, as shown by Abe *et al.* [2], so that our scheme is essentially the best we can hope for if we aim at short commitment stings.

Trapdoor: *for any tag and any message* Msg*, the following distributions are computationally indistinguishable:*

$$D_{fake} := \{(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ (\widetilde{\mathsf{com}}, \mathsf{aux}) \leftarrow \mathsf{FakeCom}(pk, tk, tag);\ \widetilde{\mathsf{dec}} \leftarrow \mathsf{FakeOpen}(\mathsf{aux}, tk, \widetilde{\mathsf{com}}, \mathsf{Msg}) : (pk, tag, \mathsf{Msg}, \widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}})\}$$

$$D_{real} := \{(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ (\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Com}(pk, tag, \mathsf{Msg}) : (pk, tag, \mathsf{Msg}, \mathsf{com}, \mathsf{dec})\}$$

Simulation-sound binding: *for any PPT adversary* $\mathcal{A}$*, the following probability is negligible*

$$\Pr[(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ (\mathsf{com}, tag, \mathsf{Msg}_1, \mathsf{Msg}_2, \mathsf{dec}_1, \mathsf{dec}_2) \leftarrow \mathcal{A}^{\mathcal{O}_{tk,pk}}(pk) : \mathsf{Msg}_1 \neq \mathsf{Msg}_2 \wedge \mathsf{Verify}(pk, tag, \mathsf{Msg}_1, \mathsf{com}, \mathsf{dec}_1) = \mathsf{Verify}(pk, tag, \mathsf{Msg}_2, \mathsf{com}, \mathsf{dec}_2) = 1 \wedge tag \notin Q],$$

where $\mathcal{O}_{tk,pk}$ *is an oracle that maintains an initially empty set* Q *and operates as follows:*

- *On input* (commit, tag)*, it runs* $(\widetilde{\mathsf{com}}, \mathsf{aux}) \leftarrow \mathsf{FakeCom}(pk, tk, tag)$*, stores* $(\widetilde{\mathsf{com}}, tag, \mathsf{aux})$*, returns* $\widetilde{\mathsf{com}}$ *and adds tag in* Q*.*
- *On input* $(\mathsf{decommit}, \widetilde{\mathsf{com}}, \mathsf{Msg})$*: if a tuple* $(\widetilde{\mathsf{com}}, tag, \mathsf{aux})$ *was previously stored, it computes* $\widetilde{\mathsf{dec}} \leftarrow \mathsf{FakeOpen}(\mathsf{aux}, tk, tag, \widetilde{\mathsf{com}}, \mathsf{Msg})$ *and returns* $\widetilde{\mathsf{dec}}$*. Otherwise,* $\mathcal{O}_{tk,pk}$ *returns* $\perp$*.*

While our SSTC to group elements will be proved secure in the above sense, a *non-adaptive* flavor of simulation-sound binding security is sufficient for the construction of non-malleable commitments. Indeed, Gennaro used [41] such a relaxed notion to achieve non-malleability from similar-looking multi-trapdoor commitments. In the non-adaptive notion, the adversary has to choose the set of tags $tag_1, \ldots, tag_\ell$ for which it wants to query the $\mathcal{O}_{tk,pk}$ oracle before seeing the public key pk.

5.1 Template of Linearly Homomorphic SPS Scheme

We first remark that *any* constant-size linearly homomorphic structure-preserving signature necessarily complies with the template below. Indeed, in order to have a linear homomorphism, each verification equation necessarily computes a product of pairings which should equal $1_{\mathbb{G}_T}$ in a valid signature. In each pairing of the product, one of the arguments must be a message or signature component while the second argument is either part of the public key or an encoding of the file identifier.

For simplicity, the template is described in terms of symmetric pairings but generalizations to asymmetric configurations are possible.

Keygen(λ, n): given λ and the dimension $n \in \mathbb{N}$ of the vectors to be signed, choose constants n_z, n_v, m. Among these, n_z and n_v will determine the signature length while m will be the number of verification equations. Then, choose $\{F_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_z\}}$, $\{G_{j,i}\}_{i\in\{1,\ldots,n\},\ j\in\{j,\ldots,m\}}$ in the group $\mathbb{G}$. The public key is $\mathsf{pk} = \big(\{F_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_z\}}, \{G_{j,i}\}_{i\in\{1,\ldots,n\},\ j\in\{j,\ldots,m\}}\big)$ while sk contains information about the representation of public elements w.r.t. specific bases.

Sign($\mathsf{sk}, \tau, (M_1, \ldots, M_n)$): Outputs a tuple $\sigma = \big(Z_1, \ldots, Z_{n_z}, V_1, \ldots, V_{n_v}\big) \in \mathbb{G}^{n_z+n_v}$.

SignDerive($\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell}$): parses each $\sigma^{(i)}$ as $\big(Z_1^{(i)}, \ldots, Z_{n_z}^{(i)}, V_1^{(i)}, \ldots, V_{n_v}^{(i)}\big)$ and computes

$$Z_\mu = \prod_{i=1}^{\ell} Z_\mu^{(i)\ \omega_i} \qquad V_\nu = \prod_{i=1}^{\ell} V_\nu^{(i)\ \omega_i} \qquad \mu \in \{1, \ldots, n_z\},\ \nu \in \{1, \ldots, n_v\}.$$

After a possible extra re-randomization step, it outputs $\big(Z_1, \ldots, Z_{n_z}, V_1, \ldots, V_{n_v}\big)$.

Verify($\mathsf{pk}, \sigma, \tau, (M_1, \ldots, M_n)$): given a signature $\sigma = \big(Z_1, \ldots, Z_{n_z}, V_1, \ldots, V_{n_v}\big) \in \mathbb{G}^{n_z+n_v}$, a tag τ and $(M_1, \ldots, M_n)$, return 0 if $(M_1, \ldots, M_n) = (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$. Otherwise, do the following.

1. For each $j \in \{1,\ldots,m\}$ and $\nu \in \{1,\ldots,n_v\}$, compute one-to-one[7] encodings $T_{j,\nu} \in \mathbb{G}$ of the tag τ as a group element.
2. Return 1 if and only if $c_j = 1_{\mathbb{G}_T}$ for $j = 1$ to m, where

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, Z_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, V_\nu) \cdot \prod_{i=1}^{n} e(G_{j,i}, M_i) \qquad j \in \{1,\ldots,m\}. \tag{3}$$

In the following, we say that a linearly homomorphic SPS is *regular* if, for each file identifier τ, any non-trivial vector $(M_1,\ldots,M_n) \neq (1_{\mathbb{G}},\ldots,1_{\mathbb{G}})$ has a valid signature.

5.2 Construction of Simulation-Sound Structure-Preserving Trapdoor Commitments

Let $\Pi^{\mathsf{SPS}} = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ be a linearly homomorphic SPS. We construct a simulation-sound trapdoor commitment as follows.

SSTC.Setup(λ, n): given the desired dimension $n \in \mathbb{N}$ of committed vectors, choose public parameters pp for the linearly homomorphic SPS scheme. Then, run $\Pi^{\mathsf{SPS}}.\mathsf{Keygen}(\lambda, n)$ to obtain a public key $\mathsf{pk} = \big(\{F_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_z\}}, \{G_{j,i}\}_{i\in\{1,\ldots,n\},\ j\in\{j,\ldots,m\}}\big)$, for some constants n_z, n_v, m, and a sk. The commitment key is $pk = \mathsf{pk}$ and the trapdoor tk consists of sk. Note that the public key defines a signature space $\mathbb{G}^{n_z+n_v}$, for constants n_z and n_v.

SSTC.Com($pk, tag, (M_1,\ldots,M_n)$): to commit to $(M_1,\ldots,M_n) \in \mathbb{G}^n$ with respect to the tag $tag = \tau$, choose $\big(Z_1,\ldots,Z_{n_z},V_1,\ldots,V_{n_v}\big) \xleftarrow{R} \mathbb{G}^{n_z+n_v}$ in the signature space. Then, run step 1 of the verification algorithm and evaluate the right-hand-side member of (3). Namely, compute

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, Z_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, V_\nu) \cdot \prod_{i=1}^{n} e(G_{j,i}, M_i) \qquad j \in \{1,\ldots,m\} \tag{4}$$

where $\{T_{j,\nu}\}_{j,\nu}$ form an injective encoding of $tag = \tau$ as a set of group elements. The commitment string is $\mathsf{com} = (c_1,\ldots,c_m)$ whereas the decommitment is $\mathsf{dec} = \big(Z_1,\ldots,Z_{n_z},V_1,\ldots,V_{n_v}\big)$.

SSTC.FakeCom(pk, tk, tag): proceeds like $\mathsf{SSTC.Com}$ with $(\hat{M}_1,\ldots,\hat{M}_n) \xleftarrow{R} \mathbb{G}^n$. If $(\hat{\mathsf{com}}, \hat{\mathsf{dec}})$ denotes the resulting pair, the algorithm outputs $\widetilde{\mathsf{com}} = \hat{\mathsf{com}}$ and the auxiliary information aux, which consists of the pair $\mathsf{aux} = ((\hat{M}_1,\ldots,\hat{M}_n), \hat{\mathsf{dec}})$ for $tag = \tau$.

SSTC.FakeOpen($\mathsf{aux}, tk, tag, \widetilde{\mathsf{com}}, (M_1,\ldots,M_n)$): the algorithm parses $\widetilde{com}$ as $(\tilde{c}_1,\ldots,\tilde{c}_m)$ and aux as $\big((\hat{M}_1,\ldots,\hat{M}_n),(\hat{Z}_1,\ldots,\hat{Z}_{n_z},\hat{V}_1,\ldots,\hat{V}_{n_v})\big)$. It first generates a linearly homomorphic signature on $(M_1/\hat{M}_1,\ldots,M_n/\hat{M}_n)$ for the tag $tag = \tau$. Namely, using the trapdoor $tk = \mathsf{sk}$, compute a signature $\sigma' = (Z'_1,\ldots,Z'_{n_z},V'_1,\ldots,V'_{n_v}) \leftarrow \Pi^{\mathsf{SPS}}.\mathsf{Sign}\big(\mathsf{sk},\tau,(M_1/\hat{M}_n,\ldots,M_n/\hat{M}_n)\big)$. Since σ' is a valid signature and $\mathsf{aux} = \big((\hat{M}_1,\ldots,\hat{M}_n),(\hat{Z}_1,\ldots,\hat{Z}_{n_z},\hat{V}_1,\ldots,\hat{V}_{n_v})\big)$ satisfies

$$\tilde{c}_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \hat{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \hat{V}_\nu) \cdot \prod_{i=1}^{n} e(G_{j,i}, \hat{M}_i) \qquad j \in \{1,\ldots,m\}, \tag{5}$$

the fake opening algorithm can run $(\tilde{Z}_1,\ldots,\tilde{Z}_{n_z},\tilde{V}_1,\ldots,\tilde{V}_{n_v}) \leftarrow \mathsf{SignDerive}(\mathsf{pk},\tau,\{(1,\sigma'),(1,\hat{\sigma})\})$, where $\hat{\sigma} = (\hat{Z}_1,\ldots,\hat{Z}_{n_z},\hat{V}_1,\ldots,\hat{V}_{n_v})$, and output $\widetilde{\mathsf{dec}} = (\tilde{Z}_1,\ldots,\tilde{Z}_{n_z},\tilde{V}_1,\ldots,\tilde{V}_{n_v})$ which is a valid de-commitment to the vector $(M_1,\ldots,M_n)$ with respect to $tag = \tau$.

SSTC.Verify($pk, tag, (M_1,\ldots,M_n), \mathsf{com}, \mathsf{dec}$): parse com as $(c_1,\ldots,c_m) \in \mathbb{G}_T^m$ and the decommitment dec as $\big(Z_1,\ldots,Z_{n_z},V_1,\ldots,V_{n_v}\big) \in \mathbb{G}^{n_z+n_v}$ (if these values do not parse properly, return 0). Then, compute a one-to-one encoding $\{T_{j,\nu}\}_{j,\nu}$ of $tag = \tau$. Return 1 if relations (4) hold and 0 otherwise.

[7] This condition can be relaxed to have collision-resistant deterministic encodings. Here, we assume injectivity for simplicity.

In Appendix E, we extend the above construction so as to build simulation-sound trapdoor commitment to vectors from any linearly homomorphic signature that fits a certain template. As a result, we obtain a modular construction of constant-size non-malleable commitment to vectors which preserves the feasibility of efficiently proving properties about committed values.

Theorem 3. *Assuming that the underlying linearly homomorphic SPS is regular and secure against non-independent Type I adversaries, the above construction is a simulation-sound trapdoor commitment to group elements.* (The proof is given in Appendix D).

A standard technique (see, e.g., [40, 41]) to construct a re-usable non-malleable commitment from a SSTC scheme is as follows. To commit to Msg, the sender generates a key-pair $(\mathsf{VK}, \mathsf{SK})$ for a one-time signature and generates $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{SSTC.Commit}(pk, \mathsf{VK}, \mathsf{MSg})$ using VK as a tag. The non-malleable commitment string is the pair $(\mathsf{com}, \mathsf{VK})$ and the opening is given by (dec, σ), where σ is a one-time signature on com, so that the receiver additionally checks the validity of σ. This construction is known to provide independence (see Definition 8 in Appendix C) and thus non-malleability with respect to opening, as proved in [32, 44].

In our setting, we cannot compute σ as a signature of com, as it consists of $\mathbb{G}_T$ elements. However, we can rather sign the pair $(\mathsf{Msg}, \mathsf{dec})$ — whose components live in $\mathbb{G}$ — as long as it uniquely determines com. To this end, we can use the one-time structure-preserving of [1, Appendix C.1] since it allows signing messages of arbitrary length using a constant-size one-time public key. Like our scheme of Section 3.2, it relies on the SDP assumption and thus yields a non-malleable commitment based on this sole assumption. Alternatively, we can move σ in the commitment string (which thus consists of $(\mathsf{com}, \mathsf{VK}, \sigma)$), in which case the one-time signature does not need to be structure-preserving but it has to be strongly unforgeable (as can be observed from the definition of independent commitments [32] recalled in Appendix C) while the standard notion of unforgeability suffices in the former case.

Acknowledgements

The authors thank Dario Catalano for his comments and for pointing a necessary correction in the proof of Lemma 1.

References

1. M. Abe, K. Haralambiev, M. Ohkubo. Signing on Elements in Bilinear Groups for Modular Protocol Design. Cryptology ePrint Archive: Report 2010/133, 2010.
2. M. Abe, K. Haralambiev, M. Ohkubo. Group to Group Commitments Do Not Shrink. In *Eurocrypt'12*, *LNCS* 7237, pp. 301–317, 2012.
3. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, M. Ohkubo. Structure-Preserving Signatures and Commitments to Group Elements. In *Crypto'10*, *LNCS* 6223, pp. 209–236, 2010.
4. M. Abe, J. Groth, K. Haralambiev, M. Ohkubo. Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In *Crypto'11*, *LNCS* 6841, pp. 649–666, 2011.
5. M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, M. Ohkubo. Constant-Size Structure-Preserving Signatures: Generic Constructions and Simple Assumptions. In *Asiacrypt'12*, *LNCS* 7658, pp. 4–24, 2012.
6. M. Abe, B. David, M. Kohlweiss, R. Nishimaki, M. Ohkubo. Tagged One-Time Signatures: Tight Security and Optimal Tag Size. In *PKC'13*, *LNCS* 7778, pp. 312–331, 2013.
7. M. Abe, J. Groth, M. Ohkubo. Separating Short Structure-Preserving Signatures from Non-interactive Assumptions. In *Asiacrypt'11*, *LNCS* 7073, pp. 628–646, 2011.
8. J.-H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, a. shelat, B. Waters. Computing on Authenticated Data. In *TCC 2012*, *LNCS* 7194, pp. 1–20, 2012.
9. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song. Provable data possession at untrusted stores. In *ACM-CCS 2007*, pp. 598–609, 2007.
10. G. Ateniese, S. Kamara, J. Katz. Proofs of Storage from Homomorphic Identification Protocols. In *Asiacrypt'09*, *LNCS* 5912, pp. 319–333, 2009.
11. N. Attrapadung, B. Libert. Homomorphic Network Coding Signatures in the Standard Model. In *PKC'11*, *LNCS* 6571, pp. 17–34, 2011.

12. N. Attrapadung, B. Libert, T. Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. In *Asiacrypt'12*, *LNCS* 7658, pp. 367–385, 2012.
13. N. Attrapadung, B. Libert, T. Peters. Efficient Completely Context-Hiding Quotable Signatures and Linearly Homomorphic Signatures. In *PKC'13*, *LNCS* 7778, pp. 367–385, pp. 386–404, 2013.
14. M. Bellare, T. Ristenpart. Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme. In *Eurocrypt'09*, *LNCS* 5479, pp. 407–424, 2009.
15. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt'04*, LNCS 3027, pages 56–73, 2004.
16. D. Boneh, X. Boyen, H. Shacham. Short Group Signatures. In *Crypto'04*, *LNCS* 3152, pp. 41–55. Springer, 2004.
17. D. Boneh, D. Freeman, J. Katz, B. Waters. Signing a Linear Subspace: Signature Schemes for Network Coding. In *PKC'09*, *LNCS* 5443, pp. 68–87, 2009.
18. D. Boneh, D. Freeman. Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In *PKC'11*, *LNCS* 6571, pp. 1–16, 2011.
19. D. Boneh, D. Freeman. Homomorphic Signatures for Polynomial Functions. In *Eurocrypt'11*, *LNCS* 6632, pp. 149–168, 2011.
20. J. Camenisch, M. Dubovitskaya, K. Haralambiev. Efficient Structure-Preserving Signature Scheme from Standard Assumptions. In *Security and Cryptography for Networks 2012 (SCN 2012)*, *LNCS* 7485, pp. 76–94, 2012.
21. J. Camenisch, K. Haralambiev, M. Kohlweiss, J. Lapon, V. Naessens. Structure Preserving CCA Secure Encryption and Applications. In *Asiacrypt'11*, *LNCS* 7073, pp. 89–106, 2011.
22. J. Camenisch, T. Gross, T.-S. Heydt-Benjamin. Rethinking accountable privacy supporting services: extended abstract. In *Digital Identity Management 2008 (DIM'08)*, pp. 1–8, 2008.
23. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS'01* pp. 136–145, 2001.
24. R. Canetti, Y. Dodis, R. Pass, S. Walfish. Universally Composable Security with Global Setup. In *TCC'07*, *LNCS* 4392, pp. 61–85, 2007.
25. R. Canetti, M. Fischlin. Universally Composable Commitments. In *Crypto'01*, *LNCS* 2139, pp. 19–40, 2001.
26. D. Catalano, D. Fiore, B. Warinschi. Adaptive Pseudo-free Groups and Applications. In *Eurocrypt'11*, *LNCS* 6632, pp. 207–223, 2011.
27. D. Catalano, D. Fiore, B. Warinschi. Efficient Network Coding Signatures in the Standard Model. In *PKC'12*, *LNCS* 7293, pp. 680–696, 2012.
28. J. Cathalo, B. Libert, M. Yung. Group Encryption: Non-Interactive Realization in the Standard Model. In *Asiacrypt'09*, *LNCS* 5912, pp. 179–196, 2009.
29. M. Chase, M. Kohlweiss. A New Hash-and-Sign Approach and Structure-Preserving Signatures from DLIN. In *Security and Cryptography for Networks 2012 (SCN 2012)*, *LNCS* 7485, pp. 131–148, 2012.
30. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto'98*, LNCS 1462, pages 13–25, 1998.
31. I. Damgård, J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC'03*, pages 426–437, 2003.
32. G. Di Crescenzo, Y. Ishai, R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *STOC'98*, pp. 141–150, 1998.
33. Y. Desmedt. Computer security by redefining what a computer is. In *New Security Paradigms Workshop (NSPW) 1993*, pp. 160–166, 1993.
34. Y. Dodis, V. Shoup, S. Walfish. Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs. In *Crypto'08*, *LNCS* 5157, pp. 21–38, 2008.
35. D. Dolev, C. Dwork, M. Naor. Non-malleable cryptography. In *STOC'91*, pages 542–552. ACM Press, 1991.
36. M. Fischlin, B. Libert, M. Manulis. Non-interactive and Re-usable Universally Composable String Commitments with Adaptive Security. In *Asiacrypt'11*, *LNCS* 7073, pp. 468–485, 2011.
37. D. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *PKC'12*, *LNCS* 7293, pp. 697–714, 2012.
38. G. Fuchsbauer. Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. Cryptology ePrint Archive: Report 2009/320, 2009.
39. E. Fujisaki. New Constructions of Efficient Simulation-Sound Commitments Using Encryption and Their Applications. In *CT-RSA'12*, *LNCS* 7178, pp. 136–155, 2012.
40. J. Garay, P. MacKenzie, K. Yang Strengthening Zero-Knowledge Protocols Using Signatures. In *Eurocrypt'03*, *LNCS* 2656, pp. 177–194, 2003.
41. R. Gennaro. Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks. In *Crypto'04*, *LNCS* 3152, pp. 220–236, 2004.
42. R. Gennaro, C. Gentry, B. Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Crypto 2010*, *LNCS* 6223, pp. 465–482, 2010.
43. R. Gennaro, J. Katz, H. Krawczyk, T. Rabin. Secure Network Coding over the Integers. In *PKC'10*, *LNCS* 6056, pp. 142–160, 2010.
44. R. Gennaro and S. Micali. Independent Zero-Knowledge Sets. In *ICALP'06*, LNCS 4052, pages 34–45, 2006.

45. J. Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *Asiacrypt'06*, *LNCS* 4284, pp. 444–459, Springer, 2006.
46. J. Groth, R. Ostrovsky. Cryptography in the Multi-String Model. In *Crypto'07*, LNCS 4622, pp. 323–341, 2007.
47. J. Groth, A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt'08*, LNCS 4965, pp. 415–432, 2008.
48. J. Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive: Report 2009/007, 2009.
49. J. Groth. Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments. In *Asiacrypt'11*, *LNCS* 7073, pp. 431–448, 2011.
50. D. Hofheinz, E. Kiltz. Programmable Hash Functions and Their Applications. In *Crypto'08*, *LNCS* 5157, pp. 21–38, 2008.
51. D. Hofheinz, T. Jager. Tightly Secure Signatures and Public-Key Encryption. In *Crypto'12*, *LNCS* 7417, pp. 590–607, 2012.
52. B. Libert, M. Yung. Non-Interactive CCA2-Secure Threshold Cryptosystems with Adaptive Security: New Framework and Constructions. In *TCC 2012*, LNCS 7194, pp. 75–93, Springer, 2012.
53. R. Johnson, D. Molnar, D. Song, D. Wagner. Homomorphic Signature Schemes. In *CT-RSA'02*, LNCS 2271, pp. 244–262, 2002.
54. P. MacKenzie, K. Yang. On Simulation-Sound Trapdoor Commitments. In *Eurocrypt'04*, *LNCS* 3027, pp. 382–400, 2004.
55. T. Malkin, I. Teranishi, Y. Vahlis, M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC'11*, 89–106, 2011.
56. M. Naor, M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC'90*, ACM Press, 1990.
57. R. Nishimaki, E. Fujisaki, K. Tanaka. A Multi-trapdoor Commitment Scheme from the RSA Assumption. In *ACISP 2010*, *LNCS*, 6168, pp. 182-199, 2010.
58. Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group Signatures with Message-Dependent Opening. In *5th International Conference on Pairing-Based Cryptography (Pairing 2012)*, LNCS 7708, pp. 270–294, 2013.
59. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Crypto'84*, *LNCS* 196, pp. 47–53, 1984.
60. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, pp. 114–127, 2005.

A Deferred Proofs for the Scheme in Section 3.2

A.1 Proof of Lemma 1

Proof. Let us assume that an *independent* adversary $\mathcal{A}$ can produce a Type II forgery with non-negligible advantage ε. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ solving a SDP instance (g_z, g_r, h_z, h) with probability at least $\varepsilon/(8(q-1)(L+1))$. Algorithm $\mathcal{B}$ chooses $(w_0, w_1, \ldots, w_L) \in \mathbb{G}^{L+1}$ in the same way as in the security proof of Waters signatures [60]. Namely, for any string $\tau \in \{0,1\}^L$, the hash value $H_{\mathbb{G}}(\tau) = w_0 \cdot \prod_{i=1}^{L} w_i^{\tau[i]}$ can be expressed as $H_{\mathbb{G}}(\tau) = g_r^{J(\tau)} \cdot h^{K(\tau)}$ for certain integer-valued functions $J, K : \{0,1\}^L \to \mathbb{Z}_p$ that remain internal to the simulation. They are further defined using the methodology of programmable hash functions [50] so that, for any distinct $\tau, \tau_1, \ldots, \tau_q$, we have $J(\tau) = 0 \bmod p$ and $J(\tau_i) \neq 0 \bmod p$ for each $i \in \{1, \ldots, q\}$ with non-negligible probability $\zeta = 1/(8 \cdot q \cdot (L+1))$.

Remaining public key components are defined by setting $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h^{\delta_i}$, with $\chi_i, \gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$ for $i = 1$ to n, as in the real key generation algorithm.

Since $\mathcal{A}$ is a Type II forger, it is expected to produce a forgery $(\tau^\star, \vec{M}^\star, \sigma^\star)$ for a tag $\tau^\star$ that was used by $\mathcal{B}$ in some signing query but for which $\vec{M}^\star \notin \mathrm{span}(\vec{M}_1, \ldots, \vec{M}_{n-1})$, where $\vec{M}_1, \ldots, \vec{M}_{n-1}$ are the vectors of $\mathbb{G}^n$ that were associated with $\tau^\star$. We denote by $\tau_1, \ldots, \tau_q$ the distinct adversarially-chosen tags involved in $\mathcal{A}$'s queries during the game. Note that, since $\mathcal{A}$ is a Type II adversary, we will have $\tau^\star \in \{\tau_1, \ldots, \tau_q\}$ at the end of the game. We also assume w.l.o.g. that exactly $n-1$ signing queries are made for each tag $\tau \in \{\tau_1, \ldots, \tau_q\}$ during the game (otherwise, $\mathcal{B}$ can simulate signing queries for itself). During its interaction with $\mathcal{A}$, the reduction $\mathcal{B}$ answers Sign, SignDerive and Reveal queries as follows.

Signing queries: At each signing query $\big(\tau_j, \vec{M} = (M_1, \ldots, M_n)\big)$ involving the j-th distinct tag τ_j, $\mathcal{B}$ evaluates the function $J(\tau_j)$ and considers the following situations.

– If $J(\tau_j) \neq 0$, $\mathcal{B}$ picks $\rho, \theta \xleftarrow{R} \mathbb{Z}_p$ and computes

$$\Theta_1 = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot (h_z)^{\frac{K(\tau_j)}{J(\tau_j)} \cdot \theta} \cdot, \qquad \Theta_2 = h^{\rho} \cdot (h_z)^{\frac{-\theta}{J(\tau_j)}},$$

which can be written $(\Theta_1, \Theta_2) = \big(h_z^{-\theta \cdot \alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\tilde{\rho}}, h^{\tilde{\rho}}\big)$ if we define $\tilde{\rho} = \rho - \frac{\theta \cdot \beta_z}{J(\tau_j)}$. Using (Θ_1, Θ_2), $\mathcal{B}$ obtains a valid signature on the vector $(M_1, \ldots, M_n)$ by computing

$$z = g_r^{\theta} \cdot \prod_{i=1}^{n} M_i^{-\chi_i} \qquad r = g_z^{-\theta} \cdot \prod_{i=1}^{n} M_i^{-\gamma_i} \qquad u = \Theta_1 \cdot \prod_{i=1}^{n} M_i^{-\delta_i} \qquad v = \Theta_2$$

The signature $\sigma = (z, r, u, v)$ is not directly sent to $\mathcal{A}$ but assigned to a new handle h and stored in an entry $(\mathsf{h}, (\tau_j, \vec{M}), \sigma)$ of the table T.

– If $J(\tau_j) = 0$, $\mathcal{B}$ picks $\rho \xleftarrow{R} \mathbb{Z}_p$ and computes

$$z = \prod_{i=1}^{n} M_i^{-\chi_i} \qquad r = \prod_{i=1}^{n} M_i^{-\gamma_i} \qquad u = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot \prod_{i=1}^{n} M_i^{-\delta_i} \qquad v = h^{\rho},$$

which corresponds to a valid signature (z, r, u, v) on $(M_1, \ldots, M_n)$ for which $\theta = 1$. Again, $\mathcal{B}$ chooses a handle h and stores $\big(\mathsf{h}, (\tau, \vec{M}), (z, r, u, v)\big)$ in the table T.

Derivation queries: Whenever $\mathcal{A}$ queries $\big((\mathsf{h}_1, \ldots, \mathsf{h}_k), (\tau, \vec{M}'), \{\beta_i\}_{i=1}^k\big)$ to the $\mathsf{SignDerive}$ oracle, $\mathcal{B}$ returns $\perp$ if not all handles $\mathsf{h}_1, \ldots, \mathsf{h}_k$ correspond to queries involving τ. Otherwise, let $\vec{M}_1, \ldots, \vec{M}_k$ be the queried vectors. If $\vec{M}' \neq \prod_{i=1}^k \vec{M}_i^{\beta_i}$, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ answers the query in the same way as the real $\mathsf{SignDerive}$ oracle, by updating the table T.

Reveal queries: When $\mathcal{A}$ supplies a handle h, $\mathcal{B}$ returns $\perp$ if no entry of the form $(\mathsf{h}, (\tau, \vec{M}), .)$ exists in T. Otherwise, $\mathcal{B}$ returns the previously generated signature σ and adds $\big((\tau, \vec{M}), \sigma\big)$ in the list Q.

Forgery: Eventually, $\mathcal{A}$ outputs a Type II forgery $(\tau^\star, \vec{M}^\star, \sigma^\star)$, where $\vec{M}^\star = (M_1^\star, \ldots, M_n^\star)$ and $\sigma^\star = (z^\star, r^\star, u^\star, v^\star) \in \mathbb{G}^4$ satisfies the verification equation. At this point, $\mathcal{B}$ evaluates $J(\tau^\star)$ and reports failure if $J(\tau^\star) \neq 0$ or if the set $\{\tau_1, \ldots, \tau_q\}$ contains at least two tags τ_{j_1}, τ_{j_2} such that $J(\tau_{j_1}) = J(\tau_{j_2}) = 0$. The same analysis as in [60] shows that, with probability $1/(8(q-1)(L+1))$, we have $J(\tau^\star) = 0$ and $J(\tau_j) \neq 0$ for each $\tau_j \in \{\tau_1, \ldots, \tau_q\} \backslash \{\tau^\star\}$. We thus find that $\mathcal{B}$'s probability not to abort during the entire game is at least $1/(8(q-1)(L+1))$.

If $\mathcal{B}$ does not fail, we have $H_{\mathbb{G}}(\tau^\star) = h^{K(\tau^\star)}$, so that $\mathcal{B}$ can compute

$$z^\dagger = \prod_{i=1}^{n} M_i^{\star -\chi_i} \qquad r^\dagger = \prod_{i=1}^{n} M_i^{\star -\gamma_i} \qquad u^\dagger = v^{\star -K(\tau^\star)} \cdot \prod_{i=1}^{n} M_i^{\star -\delta_i} \qquad v^\dagger = v^\star. \tag{6}$$

We see that $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ forms a valid signature on $(M_1^\star, \ldots, M_n^\star)$ whose last component $v^\dagger$ coincides with that of $\mathcal{A}$'s forgery. Since $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ and $(z^\star, r^\star, u^\star, v^\star)$ both satisfy the verification equations, the triple

$$(z^\ddagger, r^\ddagger, u^\ddagger) = \Big(\frac{z^\star}{z^\dagger}, \frac{r^\star}{r^\dagger}, \frac{u^\star}{u^\dagger}\Big)$$

necessarily satisfies $e(g_z, z^\ddagger) \cdot e(g_r, r^\ddagger) = e(h_z, z^\ddagger) \cdot e(h, u^\ddagger) = 1_{\mathbb{G}_T}$. We are thus left with proving that $z^\ddagger \neq 1_{\mathbb{G}}$ with all but negligible probability.

To do this, the key observation is that, in the desirable event

$$J(\tau^\star) = 0 \qquad \wedge \qquad \bigwedge_{\tau_j \neq \tau^\star} J(\tau_j) \neq 0, \tag{7}$$

the only information that $\mathcal{B}$ reveals about $(\chi_1, \ldots, \chi_n)$ is contained in the z-components of signatures involving $\tau^\star$ if $\mathcal{A}$ is a Type II adversary. Indeed, for each signing query $(\tau, \vec{M})$ such that $\tau \neq \tau^\star$, $\mathcal{B}$ introduces in the signature a fresh random exponent $\theta \in_R \mathbb{Z}_p$ that does not appear anywhere else. This allows $\mathcal{B}$ not to leak anything about $(\chi_1, \ldots, \chi_n)$ during these queries.

More precisely, let us first consider what an unbounded Type II adversary $\mathcal{A}$ can see. Throughout the game, $\mathcal{A}$ makes $n(q-1)+(n-1)$ signing queries since at most $n-1$ independent queries are allowed for the tag $\tau^\star$. Let us index these queries as $\{\big(\tau_j, \vec{M}_k = (M_{k,1}, \ldots, M_{k,n})\big)\}_{j,k}$, with $j \in \{1, \ldots, q\}$, and let $\{(z_{j,k}, r_{j,k}, u_{j,k}, v_{j,k})\}_{j,k}$ denote the answers in which $\mathcal{B}$ introduces $n(q-1)$ variables $\{\theta_{j,k}\}_{j \neq j^\star, k \in \{1,\ldots,n\}}$ in the exponent. Together with private key elements $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$, we have a total of $3n + n(q-1) = 2n + nq$ unknowns. Each signature $(z_{j,k}, r_{j,k}, u_{j,k}, v_{j,k})$ provides $\mathcal{A}$ with at most one new linearly independent equation —recall that $(z_{j,k}, v_{j,k})$ uniquely determines $r_{j,k}, u_{j,k}$ while $v_{j,k}$ does not depend on $\theta_{j,k}$ or $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$—in addition to the $2n$ linear equations resulting from the public key elements $\{(g_i, h_i)\}_{i=1}^n$.

Overall, a Type II adversary $\mathcal{A}$ thus obtains $2n + nq - 1$ linear equations which is insufficient to solve a system of $2n + nq$ unknowns. Since $(M_1^\star, \ldots, M_n^\star)$ is linearly independent of the vectors $\vec{M}_{j^\star,1}, \ldots, \vec{M}_{j^\star,n-1}$ associated with $\tau^\star$, for $\mathcal{A}$, predicting the value $z^\ddagger$ of (8) is equivalent to finding the missing piece equation that would determine $(\chi_1, \ldots, \chi_n)$. With probability $1 - 1/p$, we thus have $z^\ddagger \neq z^\star$ as claimed. □

A.2 Proof of Lemma 2

Proof. Let $\mathcal{A}$ be a Type I forger with non-negligible advantage ε. We show that it implies an algorithm $\mathcal{B}$ solving a SDP instance (g_z, g_r, h_z, h) with probability at least $\varepsilon/(8q(L+1))$.

Algorithm $\mathcal{B}$ begins by choosing $(w_0, w_1, \ldots, w_L) \in \mathbb{G}^{L+1}$ as in the security proof of Waters signatures [60]. This is done in such a way that, for any $\tau \in \{0,1\}^L$, the hash value $H_{\mathbb{G}}(\tau)$ can be written $H_{\mathbb{G}}(\tau) = g_r^{J(\tau)} \cdot h^{K(\tau)}$ for the same functions $J, K : \{0,1\}^L \to \mathbb{Z}_p$ as in the proof of Lemma 1. For any distinct $\tau, \tau_1, \ldots, \tau_q$, we will thus have $J(\tau) = 0 \bmod p$ and $J(\tau_i) \neq 0 \bmod p$ for each $i \in \{1, \ldots, q\}$ with non-negligible probability $\zeta = 1/(8 \cdot q \cdot (L+1))$.

Other public key components are defined by setting $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h^{\delta_i}$, with $\chi_i, \gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$ for $i = 1$ to n. During the game, $\mathcal{A}$'s queries are handled as follows.

Signing queries: At each signing query $\big(\tau_j, \vec{M} = (M_1, \ldots, M_n)\big)$ involving the j-th distinct tag τ_j, $\mathcal{B}$ aborts in the event that $J(\tau_j) = 0 \bmod p$. Otherwise, $\mathcal{B}$ picks $\rho, \theta \xleftarrow{R} \mathbb{Z}_p$ and computes

$$\Theta_1 = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot (h_z)^{\frac{K(\tau_j)}{J(\tau_j)} \cdot \theta} \cdot, \qquad \Theta_2 = h^\rho \cdot (h_z)^{\frac{-\theta}{J(\tau_j)}}.$$

Note that the above pair can be written $(\Theta_1, \Theta_2) = \big(h_z^{-\theta \cdot \alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\tilde{\rho}}, h^{\tilde{\rho}}\big)$, where $\tilde{\rho} = \rho - \frac{\theta \cdot \beta_z}{J(\tau_j)}$. Using (Θ_1, Θ_2), $\mathcal{B}$ obtains a well-formed signature on $(M_1, \ldots, M_n)$ by computing

$$z = g_r^\theta \cdot \prod_{i=1}^n M_i^{-\chi_i} \qquad r = g_z^{-\theta} \cdot \prod_{i=1}^n M_i^{-\gamma_i} \qquad u = \Theta_1 \cdot \prod_{i=1}^n M_i^{-\delta_i} \qquad v = \Theta_2.$$

The signature $\sigma = (z, r, u, v)$ is not directly returned to $\mathcal{A}$ but associated with a new handle h and stored in an entry $(\mathsf{h}, (\tau_j, \vec{M}), \sigma)$ of the table T.

Derivation queries: When $\mathcal{A}$ queries $\big((\mathsf{h}_1, \ldots, \mathsf{h}_k), (\tau, \vec{M}'), \{\beta_i\}_{i=1}^k\big)$ to the signature derivation oracle, $\mathcal{B}$ returns $\perp$ if not all handles $\mathsf{h}_1, \ldots, \mathsf{h}_k$ correspond to queries involving τ. Otherwise, let $\vec{M}_1, \ldots, \vec{M}_k$ be the queried vectors. If $\vec{M}' \neq \prod_{i=1}^k \vec{M}_i^{\beta_i}$, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ answers exactly like the real $\mathsf{SignDerive}$ oracle and updates the table T.

Reveal queries: When $\mathcal{A}$ queries the Reveal oracle with a handle h, $\mathcal{B}$ returns $\perp$ if no entry of the form $(\mathsf{h}, (\tau, \vec{M}), .)$ exists in T. Otherwise, $\mathcal{B}$ returns the previously computed signature σ —just like the actual Reveal oracle— and adds $\big((\tau, \vec{M}), \sigma\big)$ in the list Q.

Forgery: Eventually, $\mathcal{A}$ outputs a Type II forgery $(\tau^\star, \vec{M}^\star, \sigma^\star)$, where $\vec{M}^\star = (M_1^\star, \ldots, M_n^\star)$ and $\sigma^\star = (z^\star, r^\star, u^\star, v^\star) \in \mathbb{G}^4$ is a tuple satisfying the verification equation. At this step, $\mathcal{A}$ computes $J(\tau^\star)$ and aborts if $J(\tau^\star) \neq 0$. However, the same analysis as in [60] shows that, with probability $1/(8q(L+1))$, we have $J(\tau^\star) = 0$ and $J(\tau_j) \neq 0$ for each $j \in \{1, \ldots, q\}$.

If $\mathcal{B}$ does not fail, we have $H_{\mathbb{G}}(\tau^\star) = h^{K(\tau^\star)}$ and $\mathcal{B}$ can thus compute

$$z^\dagger = \prod_{i=1}^{n} M_i^{\star - \chi_i} \qquad r^\dagger = \prod_{i=1}^{n} M_i^{\star - \gamma_i} \qquad u^\dagger = v^{\star - K(\tau^\star)} \cdot \prod_{i=1}^{n} M_i^{\star - \delta_i} \qquad v^\dagger = v^\star. \tag{8}$$

The 4-uple $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ forms a valid signature on $(M_1^\star, \ldots, M_n^\star)$ whose last component is identical to that of $\mathcal{A}$'s forgery. Since $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ and $(z^\star, r^\star, u^\star, v^\star)$ both satisfy the verification equations, we find that

$$(z^\ddagger, r^\ddagger, u^\ddagger) = \Big(\frac{z^\star}{z^\dagger}, \frac{r^\star}{r^\dagger}, \frac{u^\star}{u^\dagger}\Big)$$

necessarily gives a non-trivial solution to the SDP instance with overwhelming probability.

Indeed, the same arguments as in the proof of Lemma 1 show that we can only have $z^\ddagger \neq 1_{\mathbb{G}}$ with probability $1/p$. The reason is that, in each signing query, $\mathcal{B}$ introduces a new blinding exponent θ that does not appear anywhere else. For this reason, $\mathcal{B}$ never leaks any information about $(\chi_1, \ldots, \chi_n)$ at any time and the element $z^\dagger$ is thus completely undetermined in $\mathcal{A}$'s view. □

B A Fully Randomizable Linearly Homomorphic SPS

In certain situations, one may want derived signatures to have the same distribution as original signatures on the same messages.

B.1 Privacy Definition

Ahn *et al.* [8] formalized a strong privacy property requiring that derived signatures be statistically indistinguishable from original ones, even when these are given.

In [12], Attrapadung *et al.* extended the definition of [8] —which only considers honestly generated signatures— to any original signature satisfying the verification algorithm.

Definition 6 ([12]). *A linearly homomorphic signature* $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ *is said* completely context hiding *if, for all public/private key pairs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\lambda)$*, for any message set* $\mathcal{S} = \{(\tau, \vec{M}_1), \ldots, (\tau, \vec{M}_{n-1})\}$*, any coefficients* $\{\omega_i\}_{i=1}^{n-1}$ *and any* $(\tau, \vec{M})$ *such that* $\vec{M} = \prod_{i=1}^{n-1} \vec{M}_i^{\omega_i}$*, for all* $\{\sigma_i\}_{i=1}^{n-1}$ *such that* $\mathsf{Verify}(\mathsf{pk}, \tau, \vec{M}_i, \sigma_i) = 1$*, the following distributions are statistically close*

$$\Big\{\big(\mathsf{sk},\ \{\sigma_i\}_{i=1}^{n-1},\ \mathsf{Sign}(\mathsf{sk}, \tau, \vec{M})\big)\Big\}_{\mathsf{sk}, \mathcal{S}, \vec{M}}, \qquad \Big\{\big(\mathsf{sk},\ \{\sigma_i\}_{i=1}^{n-1},\ \mathsf{SignDerive}\big(\mathsf{pk}, \tau,\ \{(\omega_i, \sigma_i)\}_{i=1}^{n-1}\big)\big)\Big\}_{\mathsf{sk}, \mathcal{S}, \vec{M}}.$$

In [8] Ahn *et al.* showed that, if a scheme is strongly context hiding, then Definition 1 can be simplified by removing the $\mathsf{SignDerive}$ and Reveal oracles and only providing the adversary with an ordinary signing oracle.

B.2 A Completely Context-Hiding Construction

We show that our scheme of Section 3.2 can be modified so as to become *strongly* context-hiding in the sense of [8]. Namely, signatures produced by the $\mathsf{SignDerive}$ algorithm should be statistically indistinguishable from signatures freshly generated by Sign, even when the original signatures are given.

The difficulty is that, in the scheme of Section 3.2, we cannot re-randomize the underlying θ without knowing $h_z^{\alpha_r}$. To address this problem, it is tempting to include in each signature a randomization component of the form $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\zeta}, h^{\zeta})$, for some $\zeta \in \mathbb{Z}_p$, which can be seen as a signature on the vector $(1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$. Unfortunately, the security proof ceases to go through as the reduction finds itself unable to generate a well-formed pair $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\zeta}, h^{\zeta})$ at some step of its interaction with the adversary. Our solution actually consists in committing to the signature components that cannot be re-randomized and provide evidence that committed group elements satisfy the verification equations. This is achieved using Groth-Sahai non-interactive arguments on a perfectly witness indistinguishable Groth-Sahai CRS, as in the linearly homomorphic construction of Attrapadung *et al.* [13]. A slight difference with [13], however, is that signature components $(H_{\mathbb{G}}(\tau)^{-\rho}, h^{-\rho})$ are no longer used and replaced by the technique of Malkin *et al.* [55], which yields slightly shorter signatures.

$\mathsf{Keygen}(\lambda, n)$: given a security parameter λ and the dimension $n \in \mathbb{N}$ of the subspace to be signed, choose bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^{\lambda}$. Then, do the following.

1. Choose $h \xleftarrow{R} \mathbb{G}$ and $\alpha_z, \alpha_r, \beta_z \xleftarrow{R} \mathbb{Z}_p$. Define $g_z = h^{\alpha_z}$, $g_r = h^{\alpha_r}$ and $h_z = h^{\beta_z}$.
2. For each $i \in \{1, \ldots, n\}$, pick $\chi_i, \gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$ and compute $g_i = g_z^{\chi_i} \cdot g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} \cdot h^{\delta_i}$.
3. Generate $L+1$ Groth-Sahai common reference strings by choosing $f_1, f_2 \xleftarrow{R} \mathbb{G}$ and defining vectors $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$ and $\vec{f}_{3,i} \xleftarrow{R} \mathbb{G}^3$, for each $i \in \{0, \ldots, L\}$.

The public key consists of

$$\mathsf{pk} = \Big(g_z,\ g_r,\ h_z,\ h,\ \{g_i, h_i\}_{i=1}^n,\ \mathbf{f} = \big(\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L\big)\Big)$$

while the private key is $\mathsf{sk} = \big(h_z^{\alpha_r},\ \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n\big)$.

$\mathsf{Sign}(\mathsf{sk}, \tau, (M_1, \ldots, M_n))$: to sign a vector $(M_1, \ldots, M_n) \in \mathbb{G}^n$ using $\mathsf{sk} = \big(h_z^{\alpha_r},\ \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n\big)$ with the file identifier τ, conduct the following steps.

1. Choose $\theta \xleftarrow{R} \mathbb{Z}_p$ and compute

$$z = g_r^{\theta} \cdot \prod_{i=1}^{n} M_i^{-\chi_i} \qquad r = g_z^{-\theta} \cdot \prod_{i=1}^{n} M_i^{-\gamma_i} \qquad u = h_z^{-\theta \cdot \alpha_r} \cdot \prod_{i=1}^{n} M_i^{-\delta_i}$$

2. Using the bits $\tau[1] \ldots \tau[L]$ of $\tau \in \{0,1\}^L$, define the vector $\vec{f}_{\tau} = \vec{f}_{3,0} \cdot \prod_{i=1}^{L} \vec{f}_{3,i}^{\,\tau[i]}$ so as to assemble a Groth-Sahai CRS $\mathbf{f}_{\tau} = (\vec{f}_1, \vec{f}_2, \vec{f}_{\tau})$.
3. Using $\mathbf{f}_{\tau}$, compute Groth-Sahai commitments

$$\begin{aligned}
\vec{C}_z &= (1_{\mathbb{G}}, 1_{\mathbb{G}}, z) \cdot \vec{f}_1^{\,\nu_{z,1}} \cdot \vec{f}_2^{\,\nu_{z,2}} \cdot \vec{f}_{\tau}^{\,\nu_{z,3}}, \\
\vec{C}_r &= (1_{\mathbb{G}}, 1_{\mathbb{G}}, r) \cdot \vec{f}_1^{\,\nu_{r,1}} \cdot \vec{f}_2^{\,\nu_{r,2}} \cdot \vec{f}_{\tau}^{\,\nu_{r,3}} \\
\vec{C}_u &= (1_{\mathbb{G}}, 1_{\mathbb{G}}, u) \cdot \vec{f}_1^{\,\nu_{u,1}} \cdot \vec{f}_2^{\,\nu_{u,2}} \cdot \vec{f}_{\tau}^{\,\nu_{u,3}}
\end{aligned}$$

to z, r and u, respectively. Using the randomness of these commitments, generate proofs $\vec{\pi}_1 = (\pi_{1,1}, \pi_{1,2}, \pi_{1,3}) \in \mathbb{G}^3$ and $\vec{\pi}_2 = (\pi_{2,1}, \pi_{2,2}, \pi_{2,3}) \in \mathbb{G}^3$ that (z, r, u) satisfy the verification equations $1_{\mathbb{G}_T} = e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^{n} e(g_i, M_i)$ and $1_{\mathbb{G}_T} = e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^{n} e(h_i, M_i)$. These proofs are obtained as

$$\begin{aligned}
\vec{\pi}_1 = (\pi_{1,1}, \pi_{1,2}, \pi_{1,3}) &= \big(g_z^{-\nu_{z,1}} \cdot g_r^{-\nu_{r,1}},\ g_z^{-\nu_{z,2}} \cdot g_r^{-\nu_{r,2}},\ g_z^{-\nu_{z,3}} \cdot g_r^{-\nu_{r,3}}\big) \\
\vec{\pi}_2 = (\pi_{2,1}, \pi_{2,2}, \pi_{2,3}) &= \big(h_z^{-\nu_{z,1}} \cdot h^{-\nu_{u,1}},\ h_z^{-\nu_{z,2}} \cdot h^{-\nu_{u,2}},\ h_z^{-\nu_{z,3}} \cdot h^{-\nu_{u,3}}\big)
\end{aligned}$$

and satisfy the verification equations

$$\prod_{i=1}^{n} E\big(g_i, (1_{\mathbb{G}}, 1_{\mathbb{G}}, M_i)\big)^{-1} = \quad E\big(g_z, \vec{C}_z\big) \cdot E\big(g_r, \vec{C}_r\big) \cdot E(\pi_{1,1}, \vec{f}_1) \cdot E(\pi_{1,2}, \vec{f}_2) \cdot E(\pi_{1,3}, \vec{f}_\tau) \quad (9)$$

$$\prod_{i=1}^{n} E\big(h_i, (1_{\mathbb{G}}, 1_{\mathbb{G}}, M_i)\big)^{-1} = \quad E\big(h_z, \vec{C}_z\big) \cdot E\big(h, \vec{C}_u\big) \cdot E(\pi_{2,1}, \vec{f}_1) \cdot E(\pi_{2,2}, \vec{f}_2) \cdot E(\pi_{2,3}, \vec{f}_\tau).$$

The signature consists of

$$\sigma = (\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2) \in \mathbb{G}^{15}. \quad (10)$$

SignDerive(pk, τ, $\{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell}$): given pk, a file identifier τ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parse each signature $\sigma^{(i)}$ as a tuple of the form $\sigma^{(i)} = (\vec{C}_{z,i}, \vec{C}_{r,i}, \vec{C}_{u,i}, \vec{\pi}_{1,i}, \vec{\pi}_{2,i}) \in \mathbb{G}^{15}$ for $i = 1$ to ℓ. Otherwise, the derivation process proceeds in two steps.

1. Compute

$$\vec{C}_z = \prod_{i=1}^{\ell} \vec{C}_{z,i}^{\,\omega_i} \qquad \vec{C}_r = \prod_{i=1}^{\ell} \vec{C}_{r,i}^{\,\omega_i} \qquad \vec{C}_u = \prod_{i=1}^{\ell} \vec{C}_{u,i}^{\,\omega_i} \qquad \vec{\pi_1} = \prod_{i=1}^{\ell} \vec{\pi}_{1,i}^{\,\omega_i} \qquad \vec{\pi_2} = \prod_{i=1}^{\ell} \vec{\pi}_{2,i}^{\,\omega_i}$$

2. Re-randomize the above commitments and proofs using their homomorphic property and return the re-randomized version $\sigma = (\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$.

Verify(pk, σ, τ, $(M_1, \ldots, M_n)$): given a pair $(\tau, (M_1, \ldots, M_n))$ and a purported signature σ parse the latter as $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$. Then, return 1 if and only if $(M_1, \ldots, M_n) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and equations (9) are satisfied.

We believe this construction to be of interest even if we disregard its structure-preserving property. Indeed, if we compare it with the only known completely context-hiding linearly homomorphic signature in the standard model [13], its signatures are shorter by one group element. Moreover, we can prove the security under the sole DLIN assumption whereas the scheme of [13] requires an additional assumption.

The scheme is clearly completely context hiding because signatures only consist of perfectly randomizable commitments and NIWI arguments. As for the unforgeability of the scheme, the proof of the following theorem is along the lines of [55, Theorem 5].

Theorem 4. *The above scheme provides unforgeability against independent adversaries if the DLIN assumption holds in* $\mathbb{G}$.

Proof. Since the scheme is completely context-hiding, we work with a simpler security definition where the adversary only interacts with a signing oracle. This suffices to guarantee security in the sense of Definition 2, as implied by the result of Ahn *et al.* [8]. The proof proceeds via a sequence of games. In each game, we denote by X_i the probability that the adversary $\mathcal{A}$ wins and by Y_i the probability that the simulator outputs 1.

Game_{real} : This is the real game. When the adversary $\mathcal{A}$ terminates, the simulator outputs 1 if $\mathcal{A}$ is successful. We thus have $\Pr[X_{real}] = \Pr[Y_{real}] = \mathbf{Adv}(\mathcal{A})$.

Game_0 : This game is identical to Game_{real} but we modify the generation of the public key. Namely, the vectors $(\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^{L})$ are chosen by setting $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$ and $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$, with $f_1, f_2 \xleftarrow{R} \mathbb{G}$. As for $\{\vec{f}_{3,i}\}_{i=0}^{L}$, they are obtained as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\,\xi_{0,1}} \cdot \vec{f}_2^{\,\xi_{0,2}} \cdot (1,1,g)^{\xi_{0,3}} \cdot (1,1,g)^{\mu\cdot\zeta-\rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\,\xi_{i,1}} \cdot \vec{f}_2^{\,\xi_{i,2}} \cdot (1,1,g)^{\xi_{i,3}} \cdot (1,1,g)^{-\rho_i}, \qquad\qquad i \in \{1,\ldots,L\} \end{aligned} \quad (11)$$

with $\mu \xleftarrow{R} \{0,\ldots,L\}$, $\xi_{0,1},\xi_{1,1},\ldots,\xi_{L,1} \xleftarrow{R} \mathbb{Z}_p$, $\xi_{0,2},\xi_{1,2},\ldots,\xi_{L,2} \xleftarrow{R} \mathbb{Z}_p$, $\xi_{0,3},\xi_{1,3},\ldots,\xi_{L,3} \xleftarrow{R} \mathbb{Z}_p$ and $\rho_0,\rho_1,\ldots,\rho_L \xleftarrow{R} \{0,\ldots,\zeta-1\}$, with $\zeta = 2q$ and where q is the number of distinct tags across all signing queries. Note that this change is only conceptual since $\{\vec{f}_{3,i}\}_{i=0}^{L}$ have the same distribution as in Game_{real}. We thus have $\Pr[X_0] = \Pr[Y_0] = \mathbf{Adv}(\mathcal{A})$.

Game_1 : In this game, we first raise an event F_1, which causes the simulator $\mathcal{B}$ to abort and output a random bit if it does *not* occur. Let $\tau_1,\ldots,\tau_q$ be the distinct tags successively involved in $\mathcal{A}$'s queries throughout the game and let $\tau^\star$ be the tag involved in $\mathcal{A}$'s forgery. We know that, for a Type II forger, $\tau^\star \in \{\tau_1,\ldots,\tau_q\}$ whereas $\tau^\star \notin \{\tau_1,\ldots,\tau_q\}$ for a Type I adversary. For each string $\tau \in \{0,1\}^L$, we consider the function $J(\tau) = \mu\cdot\zeta - \rho_0 - \sum_{i=1}^{L}\rho_i\tau[i]$. We also define F_1 to be the event that

$$J(\tau^\star) = 0 \qquad \wedge \qquad \bigwedge_{\tau_j \in \{\tau_1,\ldots,\tau_q\}\setminus\{\tau^\star\}} J(\tau_j) \neq 0.$$

We note that the exponents $\rho_0,\rho_1,\ldots,\rho_L$ are independent of $\mathcal{A}$'s view: as a consequence, the simulator could equivalently define $\{\vec{f}_{3,i}\}_{i=0}^{L}$ first and only choose $\{\rho_i\}_{i=0}^{L}$ – together with values $\{\xi_{3,i}\}_{i=0}^{L}$ explaining the $\{\vec{f}_{3,i}\}_{i=0}^{L}$ – at the end of the game, when $\tau^\star,\tau_1,\ldots,\tau_q$ have been defined. In the case of a Type I attack, the same analysis as [60] (after the simplification of Bellare and Ristenpart [14]) shows that $\Pr[Y_1 \wedge F_1] \geq \mathbf{Adv}(\mathcal{A})^2/(27\cdot q\cdot(L+1))$, so that $\Pr[Y_1] \geq \frac{\mathbf{Adv}(\mathcal{A})^2}{27\cdot q\cdot(L+1)} + \frac{1}{2}$.

This follows from the fact that, for any set of queries, a lower bound on the probability of event F_1 is $1/(2q(L+1))$. In the case of Type II attacks, a lower bound on the probability of F_1 for any set of queries is given by $\eta \geq 1/(2(q-1)(L+1)) > 1/(2q(L+1))$. Indeed, after re-ordering, the set of queried tags can be written $\{\tau^\star,\tau_1,\ldots,\tau_{q-1}\}$ and, from the known results [60, 50] on the programmability of Waters' hash function, we know that the probability, taken over the choice of $(\mu,\rho_0,\ldots,\rho_L)$, to have $J(\tau^\star) = 0$ and $\wedge_{j=1}^{q-1}J(\tau_j) \neq 0$ for any distinct $\tau^\star,\tau_1,\ldots,\tau_q$ is at least $1/(2(q-1)(L+1)) > 1/(2q(L+1))$.

Game_2 : In this game, we modify the distribution of the public key. Namely, $\vec{f}_1 = (f_1,1,g)$ and $\vec{f}_2 = (1,f_2,g)$ are chosen as before but, instead of generating the vectors $\{\vec{f}_{3,i}\}_{i=0}^{L}$ as previously, we choose them as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\,\xi_{0,1}} \cdot \vec{f}_2^{\,\xi_{0,2}} \cdot (1,1,g)^{\mu\cdot\zeta-\rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\,\xi_{i,1}} \cdot \vec{f}_2^{\,\xi_{i,2}} \cdot (1,1,g)^{-\rho_i}, \qquad\qquad i \in \{1,\ldots,L\} \end{aligned} \tag{12}$$

which amounts to setting $\xi_{0,3} = \xi_{1,3} = \ldots = \xi_{L,3} = 0$. This change should not significantly affect $\mathcal{A}$'s behavior if the DLIN assumption holds. More precisely, if $\mathcal{B}$ outputs 1 with noticeably different probabilities in Game_2 and Game_2, we construct a DLIN distinguisher $\mathcal{B}^{\mathrm{DLIN}}$ that takes as input $(g,f_1,f_2,f_1^{\delta_1},f_2^{\delta_2},Z)$, where $\delta_1,\delta_2 \xleftarrow{R} \mathbb{Z}_p$, and decides if $Z = g^{\delta_1+\delta_2}$ or $Z \in_R \mathbb{G}$. To this end, $\mathcal{B}^{\mathrm{DLIN}}$ uses the random self-reducibility of DLIN and creates $L+1$ independent DLIN instances out of its given instance. Namely, it picks $\varphi_i,\phi_i,\psi_i \xleftarrow{R} \mathbb{Z}_p$, for $i \in \{0,\ldots,L\}$ and sets

$$\begin{aligned} \vec{f}_{3,0} &= \big((f_1^{\delta_1})^{\varphi_0}\cdot f_1^{\phi_0},\ (f_2^{\delta_2})^{\varphi_0}\cdot f_2^{\psi_0},\ Z^{\varphi_0}\cdot g^{\phi_0+\psi_0}\cdot(1,1,g)^{\mu\cdot\zeta-\rho_0}\big) \\ \vec{f}_{3,i} &= \big((f_1^{\delta_1})^{\varphi_i}\cdot f_1^{\phi_i},\ (f_2^{\delta_2})^{\varphi_i}\cdot f_2^{\psi_i},\ Z^{\varphi_i}\cdot g^{\phi_i+\psi_i}\cdot(1,1,g)^{-\rho_i}\big), \qquad i \in \{1,\ldots,L\} \end{aligned}$$

If $Z \in_R \mathbb{G}$, $\{\vec{f}_{3,i}\}_{i=0}^{L}$ is distributed as in Game_1. If $Z = g^{\delta_1+\delta_2}$, the distribution of $\{\vec{f}_{3,i}\}_{i=0}^{L}$ is the same as in (12), which means that $\mathcal{B}^{\mathrm{DLIN}}$ is playing Game_2 with $\mathcal{A}$. For this reason, we can write $|\Pr[Y_2] - \Pr[Y_1]| \leq \mathbf{Adv}(\mathcal{B}^{\mathrm{DLIN}})$.

Game_3 : In this game, we modify the treatment of signing queries. We note that, for a given message $(\tau,\vec{M} = (M_1,\ldots,M_n))$, there is an exponential number of witnesses $(z,r,u) \in \mathbb{G}^3$ satisfying the

verification equations

$$e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^{n} e(g_i, M_i) = 1_{\mathbb{G}_T} \tag{13}$$
$$e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^{n} e(h_i, M_i) = 1_{\mathbb{G}_T}.$$

Specifically, each $z \in_R \mathbb{G}$ determines a unique pair (r, u) for which (13) holds. However, in Game_3, the simulator $\mathcal{B}$ answers all signing queries using the witness (z, r, u) such that

$$z = \prod_{i=1}^{n} M_i^{-\chi_i} \qquad r = \prod_{i=1}^{n} M_i^{-\gamma_i} \qquad u = \prod_{i=1}^{n} M_i^{-\delta_i}$$

Note that this amounts to choosing $\theta = 0$ at step 1 of the signing algorithm. Still, $\mathcal{B}$ has a valid witness for the statement to be proved. It thus assembles a Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_\tau)$ by computing $\vec{f}_\tau = \vec{f}_{3,0} \cdot \prod_{i=1}^{L} \vec{f}_{3,i}^{\tau[i]}$. Using $\mathbf{f}$, it computes Groth-Sahai commitments $\vec{C}_z, \vec{C}_r, \vec{C}_u$ to z, r and u. Using the randomness of these commitments, it faithfully generates proofs $\vec{\pi}_1$ and $\vec{\pi}_2$ satisfying the verification equations (9).

We argue that this change does not affect $\mathcal{A}$'s view whatsoever. Indeed, if event F_1 occurs we have $J(\tau^\star) = 0$ and $J(\tau_j) \neq 0$ for each $\tau_j \neq \tau^\star$. Moreover, when $J(\tau_j)$, the Groth-Sahai CRS $(\vec{f}_1, \vec{f}_2, \vec{f}_{\tau_j})$ is a perfectly hiding Groth-Sahai CRS. This means that $\vec{C}_z$, $\vec{C}_r$, $\vec{C}_u$ are perfectly hiding commitments and proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are perfectly witness indistinguishable proofs. In other words, although the proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are always generated using the witnesses (z, r, u) for which $\theta = 0$, their distribution does not depend on which specific witness is used.

In contrast, in the case of Type II attacks, signing queries involving $\tau^\star$, $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi_2})$ reveal the underlying (z, r, u) in the information theoretic sense since $(\vec{f}_1, \vec{f}_2, \vec{f}_{\tau^\star})$ is a perfectly binding CRS when $J(\tau^\star) = 0$. However, at most $n-1$ signing queries on linearly independent vectors $\vec{M}_j$ are made for the tag $\tau^\star$, so that $\mathcal{A}$ only obtains $n-1$ linearly independent equations in the exponent. As a consequence, $\mathcal{A}$ does not obtain a sufficient amount of information to recognize that $\theta = 0$ in the underlying signatures. For this reason, we find that $\Pr[Y_3] = \Pr[Y_2]$.

In Game_3, we show that a successful forger $\mathcal{A}$ implies an algorithm $\mathcal{B}$ solving a given SDP instance (g_z, g_r, h_z, h) with non-negligible advantage, which contradicts the DLIN assumption.

Recall that, when the adversary $\mathcal{A}$ terminates, it outputs $(\tau^\star, \vec{M}^\star, \sigma^\star)$, where $\vec{M}^\star = (M_1^\star, \ldots, M_n^\star)$ and $\sigma^\star = (\vec{C}_z^\star, \vec{C}_r^\star, \vec{C}_u^\star, \vec{\pi}_1^\star, \vec{\pi}_2^\star) \in \mathbb{G}^{15}$ satisfies the verification equations. At this point, if the event F_1 introduced in Game_1 occurs, we must have $J(\tau^\star) = 0$, which means that $\vec{f}_{\tau^\star} = \vec{f}_{3,0} \cdot \prod_{i=1}^{L+1} \vec{f}_{3,i}^{\tau^\star[i]}$ is in $\mathrm{span}(\vec{f}_1, \vec{f}_2)$. This implies that $\vec{C}_z^\star$, $\vec{C}_r^\star$ and $\vec{C}_u^\star$ are perfectly binding commitments. Moreover, using $(\log_g(f_1), \log_(f_2))$, $\mathcal{B}$ can extract the underlying group elements $(z^\star, r^\star, u^\star) \in \mathbb{G}^3$ by performing BBS decryptions of ciphertexts $(\vec{C}_z^\star, \vec{C}_r^\star, \vec{C}_u^\star)$. Since $(\vec{\pi}_1^\star, \vec{\pi}_2^\star)$ are valid proofs for a perfectly sound Groth-Sahai CRS, the extracted elements $(z^\star, r^\star, u^\star)$ necessarily satisfy

$$1_{\mathbb{G}_T} = e(g_z, z^\star) \cdot e(g_r, r^\star) \cdot \prod_{i=1}^{n} e(g_i, M_i^\star) = e(h_z, z^\star) \cdot e(h, u^\star) \cdot \prod_{i=1}^{n} e(h_i, M_i^\star). \tag{14}$$

Having extracted $(z^\star, r^\star, u^\star)$, $\mathcal{B}$ also computes

$$z^\dagger = \prod_{i=1}^{n} M_i^{\star -\chi_i} \qquad r^\dagger = \prod_{i=1}^{n} M_i^{\star -\gamma_i} \qquad u^\dagger = \prod_{i=1}^{n} M_i^{\star -\delta_i}, \tag{15}$$

so that $(z^\dagger, r^\dagger, u^\dagger)$ also satisfies (14). Since $(z^\dagger, r^\dagger, u^\dagger)$ and $(z^\star, r^\star, u^\star)$ both satisfy (14), the triple

$$(z^\ddagger, r^\ddagger, u^\ddagger) = \left(\frac{z^\star}{z^\dagger}, \frac{r^\star}{r^\dagger}, \frac{u^\star}{u^\dagger}\right)$$

necessarily satisfies $e(g_z, z^\ddagger) \cdot e(g_r, r^\ddagger) = e(h_z, z^\ddagger) \cdot e(h, u^\ddagger) = 1_{\mathbb{G}_T}$. To conclude the proof, we argue that $z^\ddagger \neq 1_{\mathbb{G}}$ with all but negligible probability.

To do this, we remark that, if the event F_1 defined in Game_1 occurs, the only information that $\mathcal{B}$ leaks about $(\chi_1, \ldots, \chi_n)$ resides in the unique signing query involving $\tau^\star$ if the case of Type II attacks. Indeed, for all signing queries $(\tau, \vec{M})$ involving tags τ such that $\tau \neq \tau^\star$, we have $J(\tau) \neq 0$ so that $(\vec{f_1}, \vec{f_2}, \vec{f_\tau})$ is a perfectly hiding Groth-Sahai CRS, for which proofs $(\vec{\pi}_1, \vec{\pi}_2)$ and commitments are perfectly witnesses indistinguishable. In other words, the signatures $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ for which $J(\tau) \neq 0$ leak nothing about $(\chi_1, \ldots, \chi_n)$. In contrast, in the case of Type II attacks, signing queries involving $\tau^\star$, $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi_2})$ reveal the underlying (z, r, u) in the information theoretic sense. However, at most $n-1$ linearly independent vectors $\vec{M}_j$ are signed w.r.t. $\tau^\star$, so that $\mathcal{A}$ only obtains $n-1$ linearly independent equations in the exponent for the unknowns $(\chi_1, \ldots, \chi_n)$. As a consequence, we can apply the same arguments as in the proof of Theorem 1 and Lemma 1. With probability $1-1/p$, we thus have $z^\ddagger \neq z^\star$.

To recap, we know that $\mathcal{B}$ outputs 1 with probability 1/2 if it aborts. If it does not abort, it outputs 1 whenever $\mathcal{A}$ outputs a successful forgery. We thus find

$$\Pr[Y_3] = \mathbf{Adv}^{\mathrm{SDP}}(\mathcal{B}) \cdot \left(1 - \frac{1}{p}\right)^{-1} + \frac{1}{2}.$$

When putting the above altogether, we find

$$\frac{\mathbf{Adv}(\mathcal{A})^2}{27 \cdot q \cdot (L+1)} \leq \mathbf{Adv}^{\mathrm{SDP}}(\mathcal{B}) \cdot \left(1 - \frac{1}{p}\right)^{-1} + \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B}).$$

Since any SDP algorithm $\mathcal{B}_0$ yields a DLIN distinguisher $\mathcal{B}_1$ such that $\mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B}_0) \geq 2 \cdot \mathbf{Adv}^{\mathrm{SDP}}(\mathcal{B}_1)$, we find

$$\mathbf{Adv}(\mathcal{A}) \leq \sqrt{27 \cdot q \cdot (L+1) \cdot \left[1 + \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right)^{-1}\right] \cdot \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})}$$

and the announced result follows □

C Definitions for Trapdoor Commitments

Formally, a non-interactive commitment scheme $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Verify})$ is a triple of probabilistic polynomial-time (PPT) algorithms where, on input of a security parameter λ, Setup outputs a public key pk; Com takes as input a message Msg, a public key pk and outputs a commitment/de-commitment pair $(\mathsf{com}, \mathsf{dec}) \xleftarrow{R} \mathsf{Com}(pk, \mathsf{Msg})$, and $\mathsf{Verify}(pk, \mathsf{Msg}, \mathsf{com}, \mathsf{dec})$ is deterministic and outputs 0 or 1. The correctness property guarantees that Verify always outputs 1 whenever $(\mathsf{com}, \mathsf{dec})$ is obtained by committing to Msg using honestly generated parameters.

The *binding* property demands that, given pk, no PPT adversary should be able to produce a commitment that can be opened to two distinct messages. More precisely, for any PPT adversary $\mathcal{A}$, the following advantage function should be negligible as a function of λ.

$$\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{CMT}}(\mathcal{A}) := \Pr[\ \mathsf{Verify}(pk, \mathsf{Msg}_0, \mathsf{com}, \mathsf{dec}_0) = \mathsf{Verify}(pk, \mathsf{Msg}_1, \mathsf{com}, \mathsf{dec}_1) = 1 \wedge \mathsf{Msg}_0 \neq \mathsf{Msg}_1 \ : \ pk \xleftarrow{R} \mathsf{Setup}(\lambda);\ (\mathsf{com}, \mathsf{Msg}_0, \mathsf{dec}_0, \mathsf{Msg}_1, \mathsf{dec}_1) \xleftarrow{R} \mathcal{A}(pk)\ ]$$

A commitment is also said *hiding* if commitment to distinct messages have computationally indistinguishable distributions. Formally, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following advantage term is negligible as a function of λ.

$$\mathbf{Adv}^{\text{hide}}_{\mathsf{CMT}}(\mathcal{A}) := \Big|\Pr[\ b = b' \ :\ pk \xleftarrow{R} \mathsf{Setup}(\lambda);\ b \xleftarrow{R} \{0,1\};\ (\mathsf{Msg}_0, \mathsf{Msg}_1, st) \xleftarrow{R} \mathcal{A}_1(pk);$$
$$(\mathsf{com}, \mathsf{dec}) \xleftarrow{R} \mathsf{Com}(pk, m_b);\ b' \xleftarrow{R} \mathcal{A}_2(\mathsf{com}, st)\ ] - \frac{1}{2}\Big|$$

A trapdoor commitment is a perfectly hiding commitment for which a trapdoor tk makes it possible to break the binding property and open a commitment to any arbitrary value. However, this should remain infeasible without the trapdoor. More formally, a trapdoor commitment uses two additional algorithms $(\mathsf{FakeCom}, \mathsf{FakeOpen})$ that proceed as follows.

Definition 7. *A trapdoor commitment is a tuple* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{FakeCom}, \mathsf{FakeOpen}, \mathsf{Verify})$ *of efficient algorithms where* Com *and* Verify *proceed as in an ordinary commitment and other algorithms proceed as follows.*

Setup: *is a randomized algorithm that takes as input a security parameter λ. It produces a public key pk and a trapdoor tk.*

FakeCom: *is a randomized algorithm that takes as input a public key pk and the trapdoor tk. It outputs a fake commitment string $\widetilde{\mathsf{com}}$ and some auxiliary information* aux.

FakeOpen: *takes as input a fake commitment produced by* $\mathsf{FakeCom}$ *and the corresponding auxiliary information* aux. *It also takes as input a message* Msg *and the trapdoor tk and outputs a fake de-commitment* $\widetilde{\mathsf{dec}}$ *such that* $\mathsf{Verify}(pk, \mathsf{Msg}, \widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}}) = 1$. *Moreover, the two distributions*

$$D_{fake} := \{(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ (\widetilde{\mathsf{com}}, \mathsf{aux}) \leftarrow \mathsf{FakeCom}(pk, tk);$$
$$\widetilde{\mathsf{dec}} \leftarrow \mathsf{FakeOpen}(\mathsf{aux}, tk, \widetilde{\mathsf{com}}, \mathsf{Msg}) : (pk, \mathsf{Msg}, \widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}})\}$$

and

$$D_{real} := \{(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ (\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Com}(pk, \mathsf{Msg}) : (pk, \mathsf{Msg}, \mathsf{com}, \mathsf{dec})\}$$

should be indistinguishable.

We now recall the definition of independence for commitment schemes, which is known (see, *e.g.*, [44] for a proof) to imply re-usable non-malleability with respect to opening.

Definition 8 ([32]). *A trapdoor commitment scheme* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{FakeCom}, \mathsf{FakeOpen}, \mathsf{Verify})$ *provides ℓ-independence if, for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ and any pair of ℓ-tuples $(\mathsf{Msg}_1, \ldots, \mathsf{Msg}_\ell)$, $(\mathsf{Msg}'_1, \ldots, \mathsf{Msg}_\ell)'$, the following probability is a negligible function of the security parameter λ:*

$$\Pr[(pk, tk) \leftarrow \mathsf{Setup}(\lambda);\ R_1, \ldots, R_\ell \xleftarrow{R} \{0,1\}^{\mathsf{poly}(\lambda)};$$
$$(\widetilde{\mathsf{com}}_i, \mathsf{aux}_i) \leftarrow \mathsf{FakeCom}(pk, tk, R_i)$$
$$(st, \mathsf{com}^\star) \leftarrow \mathcal{A}_1(pk, \widetilde{\mathsf{com}}_1, \ldots, \widetilde{\mathsf{com}}_\ell)\ \textit{with}\ \mathsf{com}^\star \notin \{\widetilde{\mathsf{com}}_i\}_{i=1}^{\ell}$$
$$\mathsf{dec}_i \leftarrow \mathsf{FakeOpen}(\mathsf{aux}_i, tk, \widetilde{\mathsf{com}}_i, \mathsf{Msg}_i)\ \ \forall i \in \{1, \ldots, \ell\}$$
$$\mathsf{dec}'_i \leftarrow \mathsf{FakeOpen}(\mathsf{aux}_i, tk, \widetilde{\mathsf{com}}_i, \mathsf{Msg}'_i)\ \ \forall i \in \{1, \ldots, \ell\}$$
$$(\mathsf{Msg}^\star_1, \mathsf{dec}^\star_1) \leftarrow \mathcal{A}_2(st, pk, \mathsf{Msg}_1, \mathsf{dec}_1, \ldots, \mathsf{Msg}_\ell, \mathsf{dec}_\ell)$$
$$(\mathsf{Msg}^\star_2, \mathsf{dec}^\star_2) \leftarrow \mathcal{A}_2(st, pk, \mathsf{Msg}'_1, \mathsf{dec}'_1, \ldots, \mathsf{Msg}'_\ell, \mathsf{dec}'_\ell) :$$
$$\mathsf{Msg}^\star_1 \neq \mathsf{Msg}^\star_2 \wedge \mathsf{Verify}(pk, \mathsf{Msg}^\star_1, \mathsf{com}^\star, \mathsf{dec}^\star_1) = 1 \wedge \mathsf{Verify}(pk, \mathsf{Msg}^\star_2, \mathsf{com}^\star, \mathsf{dec}^\star_2) = 1]$$

A trapdoor commitment is independent *if it provides ℓ-independence for any arbitrary $\ell \in \mathsf{poly}(\lambda)$.*

It is known (see, *e.g.*, [54]) that, when a SSTC scheme and a secure one-time signature are combined to build an ordinary commitment scheme, the simulation-sound binding property and the security of the one-time signature imply the notion of independence.

D Proof of Theorem 3

Proof. We first observe that the commitment satisfies the trapdoor property if the homomorphic SPS is regular. Indeed, in the distribution D_{fake}, the commitment $\widetilde{\mathsf{com}}$ is obtained as

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \hat{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \hat{V}_\nu) \cdot \prod_{i=1}^{n} e(G_{j,i}, \hat{M}_i) \qquad j \in \{1, \ldots, m\} \tag{16}$$

where $(\hat{M}_1, \ldots, \hat{M}_n) \in_R \mathbb{G}^n$ and for a uniformly random tuple $(\hat{Z}_1, \ldots, \hat{Z}_{n_z}, \hat{V}_1, \ldots, \hat{V}_{n_v}) \in_R \mathbb{G}^{n_z+n_v}$. We also know that, for any $(M_1, \ldots, M_n) \neq (\hat{M}_1, \ldots, \hat{M}_n)$, the vector $(M_1/\hat{M}_1, \ldots, M_n/\hat{M}_n)$ has a valid signature $\sigma' = (Z'_1, \ldots, Z'_{n_z}, V'_1, \ldots, V'_{n_v})$, so that there exists

$$\widetilde{\mathsf{dec}} = (\tilde{Z}_1, \ldots, \tilde{Z}_{n_z}, \tilde{V}_1, \ldots, \tilde{V}_{n_v}) = (Z'_1 \cdot \hat{Z}_1, \ldots, \hat{Z}'_{n_z} \cdot \hat{Z}_{n_z}, V'_1 \cdot \hat{V}_1, \ldots, V'_{n_v} \cdot \hat{V}_{n_v})$$

that explains $\widetilde{\mathsf{com}}$ as a commitment to $(M_1, \ldots, M_n)$. Moreover, since $(\hat{Z}_1, \ldots, \hat{Z}_{n_v}, \hat{V}_1, \ldots, \hat{V}_{n_v})$ was chosen uniformly in $\mathbb{G}^{n_z+n_v}$, $\widetilde{\mathsf{dec}}$ is uniform among values $(\tilde{Z}_1, \ldots, \tilde{Z}_{n_z}, \tilde{V}_1, \ldots, \tilde{V}_{n_v})$ such that

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \tilde{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \tilde{V}_\nu) \cdot \prod_{i=1}^{n} e(G_{j,i}, M_i) \qquad j \in \{1, \ldots, m\}. \tag{17}$$

In other words, the joint distribution of $(\widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}})$ is the same as if it were obtained by choosing $(\tilde{Z}_1, \ldots, \tilde{Z}_{n_z}, \tilde{V}_1, \ldots, \tilde{V}_{n_v}) \xleftarrow{R} \mathbb{G}^{n_v+n_z}$ and computing $\{c_j\}_{j=1}^m$ as per (17).

We now turn to the simulation-sound binding property and show that, if there exists a PPT adversary $\mathcal{A}$ that breaks this property with non-negligible advantage ε, there exits a non-independent Type I forger $\mathcal{B}$ against the signature scheme.

Concretely, our adversary $\mathcal{B}$ obtains a public key pk from its own challenger and sends the commitment key $pk = \mathsf{pk}$ to $\mathcal{A}$. Whenever $\mathcal{A}$ sends a query (commit, tag) to the $\mathcal{O}_{tk,pk}$ oracle, $\mathcal{B}$ faithfully runs the $\mathsf{SSTC.FakeCom}$ algorithm and thus computes $\widetilde{\mathsf{com}} = \{\tilde{c}_j\}_{j=1}^m$ according to (16) for randomly chosen $(\hat{M}_1, \ldots, \hat{M}_n) \xleftarrow{R} \mathbb{G}^n$, $\hat{\mathsf{dec}} = (\hat{Z}_1, \ldots, \hat{Z}_{n_z}, \hat{V}_1, \ldots, \hat{V}_{n_v}) \xleftarrow{R} \mathbb{G}^{n_z+n_v}$ and retains the information $\mathsf{aux} = ((\hat{M}_1, \ldots, \hat{M}_n), \hat{\mathsf{dec}})$. When the oracle $\mathcal{O}_{tk,pk}$ subsequently receives a query of the form $(\mathsf{decommit}, \widetilde{com}, (M_1, \ldots, M_n))$, the reduction $\mathcal{B}$ invokes its own signing oracle on the input $(tag, (M_1/\hat{M}_1, \ldots, M_n/\hat{M}_n))$. Upon receiving the resulting signature $(Z'_1, \ldots, Z'_{n_z}, V'_1, \ldots, V'_{n_v})$, $\mathcal{B}$ computes and returns $\widetilde{\mathsf{dec}} = (\hat{Z}_1 \cdot Z'_1, \ldots, \hat{Z}_{n_z} \cdot Z'_{n_z}, \hat{V}_1 \cdot V'_1, \ldots, \hat{V}_{n_v} \cdot V'_{n_v})$.

Eventually, the adversary $\mathcal{A}$ outputs a commitment of its own $\mathsf{com}^\star = (c_1^\star, \ldots, c_m^\star)$ along with valid openings $\mathsf{dec} = (Z_1, \ldots, Z_{n_z}, V_1, \ldots, V_{n_v})$, $\mathsf{dec}' = (Z'_1, \ldots, Z'_{n_z}, V'_1, \ldots, V'_{n_v})$ to distinct vectors $(M_1, \ldots, M_n) \neq (M'_1, \ldots, M'_n)$ for some tag $tag^\star$ that has never been used in *any* query to $\mathcal{O}_{tk,sk}$. Since both openings successfully pass the verification test, we find that

$$\big(Z_1/Z'_1, \ldots, Z_{n_z}/Z'_{n_z}, \ldots, V_1/V'_1, \ldots, V_{n_v}/V'_{n_v}\big)$$

forms a valid homomorphic signature on the vector $(M_1/M'_1, \ldots, M_n/M'_n) \neq (1_\mathbb{G}, \ldots, 1_\mathbb{G})$ for the identifier $\tau^\star = tag^\star$. By construction, $\tau^\star$ was never the input of a signing query made by $\mathcal{B}$ to its own oracle. Consequently, $\mathcal{B}$ is indeed a Type I non-independent forger with advantage ε. □

E Non-Interactive Simulation-Sound Trapdoor Commitments from Linearly Homomorphic Signatures in Groups of Public Order

MacKenzie and Yang [54] showed that simulation-sound trapdoor commitments imply digital signatures. In the converse direction, constructions of SSTCs are only known for signature schemes

admitting efficient Σ protocols. In fact, as noted by Fujisaki [39], all known constructions of non-interactive simulation-sound or multi-trapdoor [41] commitments build on signature schemes for which an efficient Σ protocol allows proving knowledge of a signature.

The idea is to commit to a message m by using m as the challenge of a Σ protocol for proving knowledge of a signature $\sigma = \mathsf{Sig}(sk, tag)$ on the tag. The commitment is given by the first message a of the Σ protocol transcript (a, m, z), which is obtained by simulating a proof of knowledge of a valid signature σ on the message tag. The commitment is subsequently opened by revealing z. By the special soundness of the Σ protocol, unless the sender actually knows a valid signature on tag, it can only open a given commitment a to one message m.

While simple, the above construction (which extends to give identity-based trapdoor commitments, as noted in [24]) does not readily extend to commit to vectors. Fujisaki [39] gave an alternative construction based on encryption schemes. However, this construction is interactive. Groth and Ostrovsky [46] finally defined the notion of simulation-extractable commitments by additionally requiring adversarially-generated commitments to be extractable instead of simply binding. A consequence of this strengthened property is that, just like UC commitments [25], simulation-extractable commitments cannot be length-reducing any longer.

This section shows that ordinary (*i.e.*, non-structure-preserving) linearly homomorphic signatures also make it possible to construct non-interactive simulation-sound (and thus non-malleable) commitments if they satisfy a certain template. Moreover, they make it possible to commit to vectors while preserving the ability of efficiently proving properties about committed vectors. We notably obtain efficient constructions based on the Diffie-Hellman and strong Diffie-Hellman [15] assumptions.

E.1 Definition and Template

We first consider a definition of unforgeability which is obtained by simplifying Definition 2 and removing the $\mathsf{SignDerive}$ and Reveal oracles. As we will see, this simplified definition will be sufficient for the construction of simulation-sound trapdoor commitments. On the other hand, unlike the definition used in [17–19], Definition 9 allows the adversary to choose the file identifiers in his signing queries.

Definition 9. *A linearly homomorphic signature scheme $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ is secure if no probabilistic polynomial time (PPT) adversary has non-negligible advantage (as a function of the security parameter $\lambda \in \mathbb{N}$) in the following game:*

1. *The adversary $\mathcal{A}$ chooses an integer $n \in \mathbb{N}$ and sends it to the challenger who runs $\mathsf{Keygen}(\lambda, n)$ and obtains $(\mathsf{pk}, \mathsf{sk})$ before sending pk to $\mathcal{A}$.*
2. *On a polynomial number of occasions, $\mathcal{A}$ chooses a tag $\tau \in \mathcal{T}$ and a vector $\vec{v}$. The challenger returns $\sigma = \mathsf{Sign}(\mathsf{sk}, \tau, \vec{v})$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs an identifier $\tau^\star$, a signature $\sigma^\star$ and a vector $\vec{y} \in \mathbb{Z}_N^n$. The adversary $\mathcal{A}$ is deemed successful if $\mathsf{Verify}(\mathsf{pk}, \tau^\star, \vec{y}^\star, \sigma^\star) = 1$ and either of the following holds:*
   - *(Type I): $\tau^\star \neq \tau_i$ for any i and $\vec{y}^\star \neq \vec{0}$.*
   - *(Type II): $\tau^\star = \tau_i$ for some $i \in \{1, \ldots, q\}$ and $\vec{y}^\star \notin V_i$, where V_i denotes the subspace spanned by all vectors $\vec{v}_1, \ldots, \vec{v}_{k_i}$ that have been queried for τ_i.*

Note that, in some cases, it may be sufficient to use a *non-adaptive* definition of unforgeability where the adversary has to declare all the file identifier $\tau_1, \ldots, \tau_q$ involved in signing queries at the very beginning of the attack (before seeing the public key pk).

Again, we say that the adversary is *independent* if

- For any given tag τ, it is restricted to only query signatures on linearly independent vectors.
- Each pair $(\vec{\tau}, \vec{m})$ is queried at most once.

Let $\Pi = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ be a linearly homomorphic signature over $\mathbb{Z}_p^n$, for some large prime $p > 2^\lambda$. We assume that Π uses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of public orders p^k and p, respectively, for some $k \in \mathbb{N}$. We also assume that each signature σ lives in $\mathbb{G}_1$. The verification algorithm takes as input a purported signature $\sigma \in \mathbb{G}_1$, a file identifier τ and a vector $\vec{m}$. It returns 1 if and only if

$$F(\sigma, \vec{m}, \mathsf{pk}, \tau) = 1_{\mathbb{G}_2}, \tag{18}$$

where F is a function ranging over the group $\mathbb{G}_2$ and satisfying certain linearity properties. Namely, for each pk produced by Keygen and each τ, we require that

$$F(\sigma_1 \cdot \sigma_2, \vec{m}_1 + \vec{m}_2, \mathsf{pk}, \tau) = F(\sigma_1, \vec{m}_1, \mathsf{pk}, \tau) \cdot F(\sigma_2, \vec{m}_2, \mathsf{pk}, \tau)$$

for any vectors $\vec{m}_1, \vec{m}_2 \in \mathbb{Z}_p^n$ and any $\sigma_1, \sigma_2 \in \mathbb{G}_1$. As a consequence, we also have

$$F(\sigma, \vec{m}, \mathsf{pk}, \tau)^\omega = F(\sigma^\omega, \omega \cdot \vec{m}, \mathsf{pk}, \tau)$$

for any $\omega \in \mathbb{Z}_p$ and any $\sigma \in \mathbb{G}_1$. Finally, the derivation algorithm $\mathsf{SignDerive}$ proceeds by computing $\mathsf{SignDerive}(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell) = \prod_{i=1}^\ell \sigma^{(i)^{\omega_i}}$.

We remark that the above template only captures schemes in groups of public order, so that constructions based on the Strong RSA assumption [26, 27] or on lattices [18, 19] are not covered. The reason is that, when working over the integers, messages and signature components may increase at each homomorphic operation. This makes it harder to render trapdoor openings indistinguishable from original de-commitments.

E.2 Simulation-sound Trapdoor Commitments from Linearly Homomorphic Signatures

From a linearly homomorphic signature scheme $\Pi = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ satisfying the template of Appendix E.1, we construct a non-interactive length-reducing SSTC as follows.

SSTC.Setup(λ, n): given the required dimension $n \in \mathbb{N}$ of committed vectors, run $\Pi.\mathsf{Keygen}(\lambda, n)$ to obtain a public key pk and a private key sk. The commitment key is $pk = \mathsf{pk}$ and the trapdoor tk consists of the private key sk of Π.

SSTC.Com($pk, tag, \vec{m}$): to commit to a vector $\vec{m} \in \mathbb{Z}_p^n$, choose $\sigma \xleftarrow{R} \mathbb{G}_1$ in the signature space. Compute and output

$$c = F(\sigma, \vec{m}, \mathsf{pk}, tag)$$

by evaluating F as in the left-hand-side member of the verification equation (18). The commitment string is $\mathsf{com} = c$ whereas the decommitment is $\mathsf{dec} = \sigma$.

SSTC.FakeCom(pk, tk, tag): proceeds identically to $\mathsf{SSTC.Com}$ but using a randomly chosen vector $\vec{m}_{fake} \xleftarrow{R} \mathbb{Z}_p^n$. If $(\hat{\mathsf{com}}, \hat{\mathsf{dec}})$ denotes the resulting commitment/decommitment pair, the algorithms sets $\widetilde{\mathsf{com}} = \hat{\mathsf{com}}$ and $\mathsf{aux} = (\vec{m}_{fake}, \hat{\mathsf{dec}})$.

SSTC.FakeOpen($\mathsf{aux}, tk, tag, \widetilde{\mathsf{com}}, \vec{m}$): the algorithm parses $\widetilde{com}$ as $\tilde{c} \in \mathbb{G}_2$ and aux as $(\vec{m}_{fake}, \hat{\mathsf{dec}})$, where $\hat{\mathsf{dec}} = \hat{\sigma} \in \mathbb{G}_1$. It first generates a linearly homomorphic signature on the difference vector $\vec{m} - \vec{m}_{fake} \in \mathbb{Z}_p^n$ for the tag $tag = \tau$. Namely, using the trapdoor $tk = \mathsf{sk}$, compute

$$\sigma' \leftarrow \Pi.\mathsf{Sign}\big(\mathsf{sk}, \tau, \vec{m} - \vec{m}_{fake}\big).$$

Finally, it computes $\widetilde{\sigma} = \mathsf{SignDerive}(\mathsf{pk}, \tau, \{(1, \hat{\sigma}), (1, \sigma')\}) = \hat{\sigma} \cdot \sigma' \in \mathbb{G}_1$ and returns $\widetilde{\mathsf{dec}} = \widetilde{\sigma}$.

SSTC.Verify($pk, tag, \vec{m}, \mathsf{com}, \mathsf{dec}$): parse the commitment com as $c \in \mathbb{G}_2$ and the opening dec as $\sigma \in \mathbb{G}_1$. If these cannot be parsed properly, return 0. Otherwise, return 1 if $c = F(\sigma, \vec{m}, \mathsf{pk}, tag)$ and 0 otherwise.

For completeness, we prove the following result in a similar way to the proof of Theorem 3.

Theorem 5. *The above construction is a secure SSTC assuming that Π is both regular and unforgeable against non-independent Type I attacks.*

Proof. The proof is very similar to the proof of Theorem 3. We first show that the commitment is a trapdoor commitment if Π is a regular homomorphic signature. Indeed, in the distribution D_{fake}, the commitment is obtained as

$$\widetilde{\mathsf{com}} = F(\hat{\sigma}, \vec{m}_{fake}, \mathsf{pk}, tag) \tag{19}$$

where $\vec{m}_{fake} \in_R \mathbb{Z}_p^n$ and $\hat{\sigma} \in_R \mathbb{G}_1$. Since Π is regular, we also know that, for any $\vec{m} \neq \vec{m}_{fake}$, the vector $\vec{m} - \vec{m}_{fake}$ has a valid signature $\sigma' \in \mathbb{G}_1$. As a consequence, there exists

$$\widetilde{\mathsf{dec}} = \tilde{\sigma} = \mathsf{SignDerive}(\mathsf{pk}, \tau, \{(1, \hat{\sigma}), (1, \sigma')\}) = \hat{\sigma} \cdot \sigma'$$

such that $\widetilde{\mathsf{com}} = F(\tilde{\sigma}, \vec{m}, \mathsf{pk}, tag)$, so that $\widetilde{\mathsf{com}}$ can be explained as a commitment to $\vec{m}$. Moreover, since $\hat{\sigma}$ was chosen uniformly in $\mathbb{G}_1$, the obtained de-commitment $\tilde{\sigma}$ is uniform among values such that

$$\widetilde{\mathsf{com}} = F(\tilde{\sigma}, \vec{m}, \mathsf{pk}, tag)$$

Said otherwise, $(\widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}})$ has the same distribution as if it were obtained by choosing $\widetilde{dec} = \tilde{\sigma} \xleftarrow{R} \mathbb{G}_1$ and computing $\widetilde{\mathsf{com}} = F(\tilde{\sigma}, \vec{m}, \mathsf{pk}, tag)$.

To establish the simulation-sound binding property, we show that, if there exists a PPT adversary $\mathcal{A}$ that breaks this property with advantage ε, the homomorphic signature scheme Π can be broken by a non-independent Type I forger $\mathcal{B}$ with the same advantage ε.

Algorithm $\mathcal{B}$ takes as input a linearly homomorphic signature public key pk and sends $pk = \mathsf{pk}$ to the simulation-binding adversary $\mathcal{A}$. When $\mathcal{A}$ sends a query (commit, tag) to the $\mathcal{O}_{tk,pk}$ oracle, $\mathcal{B}$ runs the $\mathsf{SSTC.FakeCom}$ algorithm and computes $\widetilde{\mathsf{com}} = F(\hat{\sigma}, \vec{m}_{fake}, \mathsf{pk}, tag)$ for randomly chosen $\hat{\sigma} \xleftarrow{R} \mathbb{G}_1$ and $\vec{m}_{fake} \xleftarrow{R} \mathbb{Z}_p^n$. It retains the state information $\mathsf{aux} = (\vec{m}_{fake}, \hat{\sigma})$. For each invocation of the oracle $\mathcal{O}_{tk,pk}$ for an input of the form $(\mathsf{decommit}, \widetilde{com}, \vec{m})$, $\mathcal{B}$ sends the query $(tag, \vec{m} - \vec{m}_{fake})$ to its own signing oracle. Upon receiving the latter's response σ', $\mathcal{B}$ computes and returns $\widetilde{\mathsf{dec}} = \sigma' \cdot \hat{\sigma}$.

Eventually, $\mathcal{A}$ comes up with a commitment of its own $\mathsf{com}^\star$ with valid openings $\mathsf{dec} = \sigma$, $\mathsf{dec}' = \sigma'$ to distinct vectors $\vec{m} \neq \vec{m}'$ for a tag $tag^\star$ that it never submitted to $\mathcal{O}_{tk,sk}$. Since $\vec{m} \neq \vec{m}'$ and dec and dec' are valid openings of $\mathsf{com}^\star$ to $\vec{m}$ and $\vec{m}'$, respectively, the triple

$$\left(\tau^\star, \sigma/\sigma', \vec{m} - \vec{m}'\right)$$

forms a valid Type I forgery for the linearly homomorphic scheme Π. □

E.3 Instantiations

Construction from the Diffie-Hellman Assumption. Previously, non-malleable commitments based on the CDH assumption were — implicitly or explicitly — described in [34, 57] but it is not immediate how to extend them to commit to vectors in a modular way.

In [12], Attrapadung *et al.* described a linearly homomorphic signature which is notably secure against Type I independent adversaries — as implicitly proved by [12, Lemma 8] — under the computational Diffie-Hellman (CDH) assumption.

Keygen(λ, n): given a security parameter $\lambda \in \mathbb{N}$ and an integer $n \in \mathsf{poly}(\lambda)$, choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Choose $\alpha \xleftarrow{R} \mathbb{Z}_p$, $g, v \xleftarrow{R} \mathbb{G}$ and $u_0, u_1, \ldots, u_L \xleftarrow{R} \mathbb{G}$, for some $L \in \mathsf{poly}(\lambda)$. These elements $(u_0, \ldots, u_L) \in \mathbb{G}^{L+1}$ will be used to implement a programmable hash function $H_\mathbb{G} : \{0,1\}^L \to \mathbb{G}$ such that any L-bit string $\tau = \tau[1] \ldots \tau[L] \in \{0,1\}^L$ is mapped to the hash value $H_\mathbb{G}(\tau) = u_0 \cdot \prod_{i=1}^{L} u_i^{\tau[i]}$. Pick $g_i \xleftarrow{R} \mathbb{G}$ for $i = 1$ to n. Finally, define the identifier space $\mathcal{T} := \{0,1\}^L$. The private key is $\mathsf{sk} := \alpha$ and the public key consists of

$$\mathsf{pk} := \Big((\mathbb{G}, \mathbb{G}_T),\ g,\ g^\alpha,\ v,\ \{g_i\}_{i=1}^n,\ \{u_i\}_{i=0}^L \Big).$$

Sign($\mathsf{sk}, \tau, \vec{m}$): given a vector $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, a file identifier $\tau \in \{0,1\}^L$ and the private key $\mathsf{sk} = \alpha \in \mathbb{Z}_p$, return $\perp$ if $\vec{m} = \vec{0}$. Otherwise, choose $r, s \xleftarrow{R} \mathbb{Z}_p$. Then, compute a signature $\sigma = (\sigma_1, \sigma_2, s) \in \mathbb{G}^2 \times \mathbb{Z}_p$ as

$$\sigma_1 = (g_1^{m_1} \cdots g_n^{m_n} \cdot v^s)^\alpha \cdot H_\mathbb{G}(\tau)^r, \qquad \sigma_2 = g^r.$$

SignDerive($\mathsf{pk}, \tau, \{(\beta_i, \sigma_i)\}_{i=1}^\ell$): given pk, a file identifier τ and ℓ tuples (β_i, σ_i), parse each signature σ_i as $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, s_i)$ for $i = 1$ to ℓ. Then, choose $\tilde{r} \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\sigma_1 = \prod_{i=1}^{\ell} \sigma_{i,1}^{\beta_i} \cdot H_\mathbb{G}(\tau)^{\tilde{r}} \qquad \sigma_2 = \prod_{i=1}^{\ell} \sigma_{i,2}^{\beta_i} \cdot g^{\tilde{r}} \qquad s = \sum_{i=1}^{\ell} \beta_i \cdot s_i$$

and output (σ_1, σ_2, s).

Verify($\mathsf{pk}, \tau, \vec{m}, \sigma$): given pk, a signature $\sigma = (\sigma_1, \sigma_2, s)$ and a message $(\tau, \vec{m})$, where $\tau \in \{0,1\}^L$ and $\vec{m}$ is a vector $(m_1, \ldots, m_n) \in (\mathbb{Z}_p)^n$, return 0 if $\vec{m} = \vec{0}$. Otherwise, return 1 if

$$e(\sigma_1, g) = e(g_1^{m_1} \cdots g_n^{m_n} \cdot v^s, g^\alpha) \cdot e(H_\mathbb{G}(\tau), \sigma_2).$$

and 0 otherwise.

This scheme can be seen as a specific instantiation of the template where the group $\mathbb{G}_1$ is a product $\mathbb{G}_1 = \mathbb{G}^2 \times \mathbb{Z}_p$, which is a group for the operation $(\cdot, \cdot, +)$, and $\mathbb{G}_2 = \mathbb{G}_T$. Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ thus have order p^3 and p, respectively. As for the linear function F, it can be instantiated as

$$F\big((\sigma_1, \sigma_2, s), \vec{m}, \mathsf{pk}, \tau\big) := e(\sigma_1, g^{-1}) \cdot e(H_\mathbb{G}(\tau), \sigma_2) \cdot e(g_1^{m_1} \cdots g_n^{m_n} \cdot v^s, g^\alpha)$$

As a result, we obtain a new non-interactive simulation-sound trapdoor commitment to vectors under the CDH assumption. We note that the scheme can be optimized by removing the terms v^s and s, so as to have $(\sigma_1, \sigma_2) = \big((\prod_{i=1}^n g_i^{m_i})^\alpha \cdot H_\mathbb{G}(\tau)^r, g^r\big)$ and

$$F\big((\sigma_1, \sigma_2), \vec{m}, \mathsf{pk}, \tau\big) := e(\sigma_1, g^{-1}) \cdot e(H_\mathbb{G}(\tau), \sigma_2) \cdot e(g_1^{m_1} \cdots g_n^{m_n}, g^\alpha)$$

Indeed, in the proof of Lemma 8 in [12], we observe that, if the signature scheme only needs to be secure against Type I attacks, the terms $(v^s, s) \in \mathbb{G} \times \mathbb{Z}_p$ can be eliminated.

Unlike the CDH-based construction of [39], the above commitment scheme is non-interactive and allows committing to vectors with a constant-size commitment string. Unlike the solution consisting in committing to a short string obtained by hashing the vector, our solution makes it possible for the sender to prove properties (using Σ protocols or Groth-Sahai proofs) about committed vectors in an efficient way.

We also remark that, for vectors of dimension $n = 1$, we obtain a simplification of existing multi-trapdoor (or identity-based) trapdoor commitments [34, 57] based on the Waters signature: instead of starting from a Σ protocol for proving knowledge of a Waters signature, we obtain a more

efficient scheme by building the commitment algorithm on the verification equation of the underlying signature: recall that the verification equation of Waters signatures (σ_1, σ_2) returns 1 if and only if it holds that $e(\sigma_1, g) = e(g^\alpha, h) \cdot e(H_{\mathbb{G}}(M), \sigma_2)$, where $M \in \{0,1\}^L$ is the message and g^α, h are part of the public key. Now, to commit to a message $m \in \mathbb{Z}_p$ the sender can pick random $\theta_1, \theta_2 \in \mathbb{G}$ and compute $\mathsf{com} = e(g^\alpha, h)^m \cdot e(g, \theta_1) \cdot e(H_{\mathbb{G}}(\tau), \theta_2) \in \mathbb{G}_T$ and $\mathsf{dec} = (\theta_1, \theta_2)$. It is easy to see that a signature (σ_1, σ_2) on τ allows trapdoor opening com. Moreover, the resulting scheme gives shorter commitment string and a faster verification algorithm than in [24, 57].

Construction from the Strong Diffie-Hellman Assumption. As mentioned earlier, in the application to non-malleable commitments, simulation-sound trapdoor commitments only need to be secure against adversaries that choose beforehand (before receiving the public key) on which tags they will see equivocations of commitments produced by $\mathsf{FakeCom}$. In this case, we only need the underlying linearly homomorphic signature to be secure against non-adaptive Type I independent adversaries. The construction of Catalano, Fiore and Warinschi [27] is an example of such system. In [27], it was implicitly[8] proved that the scheme is secure against non-adaptive (independent) Type I adversaries under the strong Diffie-Hellman assumption [15].

$\mathsf{Keygen}(\lambda, n)$: given a security parameter $\lambda \in \mathbb{N}$ and an integer $n \in \mathsf{poly}(\lambda)$, choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Choose $\alpha \xleftarrow{R} \mathbb{Z}_p$, $g, v \xleftarrow{R} \mathbb{G}$ and $g_i \xleftarrow{R} \mathbb{G}$ for $i = 1$ to n. Finally, define the identifier space $\mathcal{T} := \mathbb{Z}_p$. The private key is $\mathsf{sk} := \alpha$ and the public key consists of

$$\mathsf{pk} := \Big((\mathbb{G}, \mathbb{G}_T),\ g,\ g^\alpha,\ v,\ \{g_i\}_{i=1}^n \Big).$$

$\mathsf{Sign}(\mathsf{sk}, \tau, \vec{m})$: given a vector $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, a file identifier $\tau \in \mathbb{Z}_p$ and the private key $\mathsf{sk} = \alpha \in \mathbb{Z}_p$, choose $s \xleftarrow{R} \mathbb{Z}_p$. Then, compute a signature $\sigma = (\sigma_1, s) \in \mathbb{G} \times \mathbb{Z}_p$ where

$$\sigma_1 = \big(g_1^{m_1} \cdots g_n^{m_n} \cdot v^s \big)^{\frac{1}{\alpha + \tau}}.$$

$\mathsf{SignDerive}(\mathsf{pk}, \tau, \{(\beta_i, \sigma_i)\}_{i=1}^\ell)$: given pk, a file identifier τ and ℓ tuples (β_i, σ_i), parse each signature σ_i as $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, s_i)$ for $i = 1$ to ℓ. Then, choose $\tilde{r} \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\sigma_1 = \prod_{i=1}^{\ell} \sigma_{i,1}^{\beta_i} \qquad\qquad s = \sum_{i=1}^{\ell} \beta_i \cdot s_i$$

and output (σ_1, s).

$\mathsf{Verify}(\mathsf{pk}, \tau, \vec{m}, \sigma)$: given the public key pk, a signature $\sigma = (\sigma_1, s)$ and a message $(\tau, \vec{m})$, where $\tau \in \mathbb{Z}_p$ and $\vec{m} = (m_1, \ldots, m_n) \in (\mathbb{Z}_p)^n$, return 1 if and only if

$$e(\sigma_1, g^\tau \cdot g^\alpha) = e(g_1^{m_1} \cdots g_n^{m_n} \cdot v^s, g). \tag{20}$$

This construction can also be seen as a special case of our template where $\mathbb{G}_1 = \mathbb{G} \times \mathbb{Z}_p$ is a group for the operation $(\cdot, +)$ and $\mathbb{G}_2 = \mathbb{G}_T$ is a multiplicative group. Here, we thus have $|\mathbb{G}_1| = p^2$ and $|\mathbb{G}_2| = p$. The linear function F is now defined as

$$F\big((\sigma_1, s), \vec{m}, \mathsf{pk}, \tau\big) := e(\sigma_1, g^\tau \cdot g^\alpha) \cdot e(g_1^{m_1} \cdots g_n^{m_n} \cdot v^s, g^{-1}).$$

The linearly homomorphic signature of [27] thus implies a non-interactive non-adaptive simulation-sound trapdoor commitment to vectors based on the strong Diffie-Hellman assumption. Again, the scheme can be simplified by removing the term v^s since the underlying signature only needs to be secure against non-adaptive Type I attacks. In the case $n = 1$, the resulting non-malleable commitment is a variant of the one of [41, Section 4.2].

[8] Catalano *et al.* [27] consider a model where the file identifiers are always chosen by the challenger at each signing query in the security game. However, the security proof of [27, Lemma 1] does not require the file identifiers to be uniformly distributed and it goes through if they are chosen by the adversary at the outset of the game instead of being chosen by the reduction.