Authenticating Computation on Groups: New Homomorphic Primitives and Applications[⋆]

Dario Catalano[1], Antonio Marcedone[2,⋆⋆], and Orazio Puglisi[1]

[1] Dipartimento di Matematica ed Informatica, Università di Catania, Italy,
{catalano, opuglisi}@dmi.unict.it
[2] Cornell University, USA and
Scuola Superiore di Catania, Università di Catania, Italy,
marcedone@cs.cornell.edu

Abstract. In this paper we introduce new primitives to authenticate computation on data expressed as elements in (cryptographic) groups. As for the case of homomorphic authenticators, our primitives allow to verify the correctness of the computation *without* having to know of the original data set. More precisely, our contributions are two-fold.
First, we introduce the notion of *linearly homomorphic authenticated encryption with public verifiability* and show how to instantiate this primitive (in the random oracle model) to support Paillier's ciphertexts. This immediately yields a very simple and efficient (publicly) verifiable computation mechanism for encrypted (outsourced) data based on Paillier's cryptosystem.
As a second result, we show how to construct linearly homomorphic signature schemes to sign elements in bilinear groups (LHSG for short). Such type of signatures are very similar to (linearly homomorphic) structure preserving ones, but they allow for more flexibility, as the signature is explicitly allowed to contain components which *are not* group elements. In this sense our contributions are as follows. First we show a very simple construction of LHSG that is secure against weak random message attack (RMA). Next we give evidence that RMA secure LHSG are interesting on their own right by showing applications in the context of on-line/off-line homomorphic and network coding signatures. This notably provides what seems to be the first instantiations of homomorphic signatures achieving on-line/off-line efficiency trade-offs. Finally, we present a generic transform that converts RMA-secure LHSG into ones that achieve full security guarantees.

1 Introduction

Homomorphic signatures allow to validate computation over authenticated data. More precisely, a signer holding a dataset $\{m_i\}_{i=1,\ldots,t}$ can produce corresponding

[⋆] Extended Abstract. The full version of this paper is available at https://eprint.iacr.org/2013/801.pdf
[⋆⋆] Part of the work done while visiting Aarhus University.

signatures $\sigma_i = \mathbf{Sign}(\mathrm{sk}, m_i)$ and store the signed dataset on a remote server. Later the server can (publicly) compute $m = f(m_1, \ldots, m_t)$ together with a (succinct) valid signature σ on it. A keynote feature of homomorphic signature is that the validity of σ can be verified *without* needing to know the original messages $m_1, \ldots, m_t$. Because of this flexibility homomorphic signatures have been investigated in several settings and flavors. Examples include homomorphic signatures for linear and polynomial functions [9,8], redactable signatures [26], transitive signatures and more [32,36]. In spite of this popularity, very few realizations of the primitive encompass the very natural case where the computation one wants to authenticate involves elements belonging to typical cryptographic groups (such as, for instance, groups of points over certain classes of elliptic curves, or groups of residues modulo a composite integer).

Our Contribution. In this paper we put forward new tools that allow to authenticate computation on elements in (cryptographic) groups. In this sense our contributions are two-fold. First, we define a new primitive that we call *Linearly Homomorphic Authenticated Encryption with Public Verifiability* (LAEPuV for short). Informally, this primitive allows to authenticate computation on (outsourced) encrypted data, with the additional benefit that the correctness of the computation can be publicly verified. The natural application of this primitive is the increasingly relevant scenario where a user wants to store (encrypted) data on the cloud in a way such that she can later delegate the cloud to perform computation on this data. As a motivating example, imagine that a teacher wants to use the cloud to store the grades of the homeworks of her students. To do so she can create a file identifier fid for each class (e.g. *Cryptography - Spring 2014*), sign each record tied with the corresponding fid and store everything offline. There are two problems with this solution. First, if the teacher wants to compute statistics (e.g. average grades) on these records she has to download all the data locally. Second, since data is stored in clear, outsourcing it offline might violate the privacy of students. *LAEPuV* solves both these issues, as it allows to delegate (basic) computations (i.e. linear functions) on encrypted data in a reliable and efficient way. In particular, it allows to verify the correctness of the computation *without* needing to download the original ciphertexts locally. Moreover, as for the case of homomorphic signatures, correctness of the computation can be publicly verified via a succinct tag whose size is *independent* of the size of the outsourced dataset.

We show an (efficient) realization of the primitive (in the random oracle model) supporting Paillier's ciphertexts. At an intuitive level our construction works by combining Paillier's encryption scheme with some appropriate additively homomorphic signature scheme. Slightly more in detail, the idea is as follows. One first decrypts a "masking" of the ciphertext C and then signs the masked plaintext using the linearly homomorphic signature. Thus we use the homomorphic signature to authenticate computations on ciphertexts by basically authenticating (similar) computations on the masked plaintexts. The additional advantage of this approach is that it allows to authenticate computation on Paillier's ciphertexts while preserving the possibility to re-randomize the ciphertexts. This means, in particular, that our scheme allows to authenticate computation also on randomized versions of the original ciphertexts[3].

This result allows to implement a very simple and efficient (publicly) verifiable computation mechanism for encrypted (outsourced) data based on Paillier's cryptosystem [34]. Previous (efficient) solutions for this problem rely on linearly homomorphic structure preserving signatures (LHSPS, for short) [30] and, as such, only supported cryptosystems defined over pairing-friendly groups. Since, no (linearly homomorphic) encryption scheme supporting exponentially large message spaces is known to exist in such groups, our construction appears to be the first one achieving this level of flexibility.

Beyond this efficiency gain, we stress that our approach departs from the LHSPS-based one also from a methodological point of view. Indeed, the latter authenticates computation by signing outsourced ciphertexts, whereas we sign (masked versions of) the corresponding plaintexts. This is essentially what buy us the possibility of relying on basic linearly homomorphic signatures, rather than on, seemingly more complicate, structure preserving ones. On the negative side, our solutions require the random oracle, whereas the only known LHSPS-based construction works in the standard model.

As additional byproduct of this gained flexibility, we show how to generalize our results to encompass larger classes of encryption primitives. In particular, we show that our techniques can be adapted to work using any encryption scheme, with some well defined homomorphic properties, as underlying encryption primitive. Interestingly, this includes many well known linearly homomorphic encryption schemes such as [25,33,29].

Signing elements in bilinear groups. As a second main contribution of this paper, we show how to construct a very simple linearly homomorphic signature scheme to sign elements in bilinear groups (LHSG for short). Such type of signatures are very similar to (linearly homomorphic) structure preserving ones, but they allow for more flexibility, as the signature is explicitly allowed to contain components which *are not* group elements (and thus signatures are not necessarily required to comply with the Groth-Sahai famework). More in detail, our scheme is proven secure against random message attack (RMA)[4] under a variant of the Computational Diffie-Hellman assumption introduced by Kunz-Jacques and Pointcheval in [28]. In this sense, our construction is less general (but also conceptually simpler) than the linearly homomorphic structure preserving signature recently given in [30]. Also, the scheme from [30] allows to sign vectors of arbitrary dimension, while ours supports vectors composed by one single component only.

[3] We stress however that this does not buy us privacy with respect to the functionality, i.e. the derived (authenticated) ciphertexts are not necessarily indistinguishable from freshly generated (authenticated) ones.

[4] Specifically, by random message security here we mean that the unforgeability guarantee holds only with respect to adversaries that are allowed to see signatures corresponding to messages randomly chosen by the signer

Interestingly, we show that this simple tool has useful applications in the context of *on-line/off-line* (homomorphic) signatures. Very informally, on-line/off-line signatures allow to split the cost of signing in two phases. An (expensive) offline phase that can be carried out *without* needing to know the message m to be signed and a much more efficient on-line phase that is done once m becomes available. In this sense, on-line/off-line homomorphic signature could bring similar efficiency benefits to protocols relying on homomorphic signatures. For instance, they could be used to improve the overall efficiency of linear network coding routing mechanisms employing homomorphic signatures to fight pollution attacks[5].

We show that RMA-secure LHSG naturally fit this more demanding on-line/off-line scenario. Specifically, we prove that if one combines a RMA-secure LHSG with (vector) Σ protocols with some specific homomorphic properties, one gets a fully fledged linearly homomorphic signature achieving a very efficient online phase. Moreover, since the resulting signature scheme supports vectors of arbitrary dimensions as underlying message space, our results readily generalize to the case of network coding signatures [7]. More concretely, by combining our RMA-secure scheme together with (a variant of) Schnorr's identification protocol we get what seem to be the first constructions of secure homomorphic and network coding signatures offering online/offline efficiency tradeoffs both for the message and the file identifier.

To complete the picture, we provide an efficient and *generic* methodology to convert RMA-secure LHSG into ones that achieve full security . We stress that while similar transforms were known for structure preserving signatures (e.g. [17]), to our knowledge this is the first such transform for the case of *linearly homomorphic* signatures in general.

Other Related Work. Authenticated Encryption (AE) allows to simultaneously achieve privacy and authentication. In fact AE is considered to be the standard for symmetric encryption, and many useful applications are based on this primitive. Formal definitions for (basic) AE where provided by Bellare and Namprempre in [6]. More closely related to our setting is the notion of homomorphic authenticated encryption recently proposed by Joo and Yun in [27]. With respect to ours, their definitions encompass a wider class of functionalities, but do not consider public verifiability.

Computationally Sound proofs. In the random oracle model, the problem of computing reliably on (outsourced) encrypted data can be solved in principle using Computationally Sound (CS) proofs [31]. The advantage of this solution, with respect to ours, is that it supports arbitrary functionalities. On the other hand, it is much less efficient as it requires the full machinery of the PCP theorem. Moreover, composition in CS proofs is quite complicate to achieve, whereas

[5] This is because the sender could preprocess many off-line computations at night or when the network traffic is low and then use the efficient online signing procedure to perform better when the traffic is high.

it comes for free in our solution, as the outputs of previous computations can always be used as inputs for new ones.

Linearly homomorphic signatures. The concept of homomorphic signature scheme was originally introduced in 1992 by Desmedt [18], and then refined by Johnson, Molnar, Song, Wagner in 2002 [26]. Linearly homomorphic signatures were introduced in 2009 by Boneh *et al.* [7] as a way to prevent pollution attacks in network coding. Following [7] many other works further explored the notion of homomorphic signatures by proposing new frameworks and realizations [23,3,9,8,14,4,15,21,5,13,16]. In the symmetric setting constructions of homomorphic message authentication codes have been proposed by [7,24,11,12].

Recently Libert *et al.* [30] introduced and realized the notion of *Linearly Homomorphic Structure Preserving signatures* (LHSPS for short). Informally LHSPS are like ordinary SPS but they come equipped with a linearly homomorphic property that makes them interesting even beyond their usage within the Groth Sahai framework. In particular Libert *et al.* showed that LHSPS can be used to enable simple verifiable computation mechanisms on encrypted data. More surprisingly, they observed that linearly homomorphic SPS (generically) yield efficient simulation sound trapdoor commitment schemes [22], which in turn imply non malleable trapdoor commitments [19] to group elements.

On-Line/Off-Line Signatures. On-Line/Off-Line digital signature were introduced by Even, Goldreich and Micali in [20]. In such schemes the signature process consists of two parts: a computationally intensive one that can be done Off-Line (i.e. when the message to be signed is not known) and a much more efficient online phase that is done once the message becomes available. There are two general ways to construct on-line/off-line signatures: using one time signatures [20] or using chameleon hash [35].
In [10] Catalano *et al.*, unified the two approaches by showing that they can be seen as different instantiations of the same paradigm.

2 Preliminaries and notation

We denote with $\mathbb{Z}$ the set of integers, with $\mathbb{Z}_p$ the set of integers modulo p. An algorithm $\mathcal{A}$ is said to be PPT if it's modelled as a probabilistic Turing machine that runs in polynomial time in its inputs. If S is a set, then $x \overset{\$}{\leftarrow} S$ denotes the process of selecting one element x from S uniformly at random. A function f is said to be negligible if for all polynomial p there exists $n_0 \in \mathbb{N}$ such that for each $n > n_0$, $|f(n)| < \frac{1}{p(n)}$.

Computational assumptions We start by recalling a couple of relevant computational assumptions. Let $\mathbb{G}$ be a finite (multiplicative) group of prime order p. The 2-out-of-3 Computational Diffie-Hellman assumption was introduced by Kunz-Jacques and Pointcheval in [28] as a relaxation of the standard CDH assumption. It is defined as follows.

Definition 1 (2-3CDH). *We say that the 2-out-of-3 Computational Diffie-Hellmann assumption holds in $\mathbb{G}$ if, given a random generator $g \in \mathbb{G}$, there exists no PPT $\mathcal{A}$ that on input (g, g^a, g^b) (for random $a, b \xleftarrow{\$} \mathbb{Z}_p$) outputs h, h^{ab} $(h \neq 1)$ with more than negligible probability.*

Also, we recall the Decisional Composite Residuosity Assumption, introduced by Paillier in [34].

Definition 2 (DCRA). *We say that the Decisional composite residuosity assumption (DCRA) holds if there exists no PPT $\mathcal{A}$ that can distinguish between a random element from $\mathbb{Z}^*_{N^2}$ and one from the set $\{z^N | z \in \mathbb{Z}^*_{N^2}\}$ (i.e. the set of the N-th residues modulo N^2), when N is the product of two random primes of proper size.*

3 (Publicly) Verifiable delegation of computation on outsourced ciphertext.

In this section, we introduce a new primitive that we call Linearly Homomorphic Authenticated Encryption with Public Verifiability (LAEPuV). Informally, this notion is inspired by the concept of homomorphic authenticated encryption, introduced by Joo and Yun [27]. Important differences are that our definition[6] focuses on linear functions and explicitly requires public verifiability.
Next, we provide an instantiation of this primitive supporting Paillier's scheme as the underlying encryption mechanism.
Additionally, in this and the following sections, we adopt the following conventions

- The set $\mathcal{F}$ of supported functionalities, is the set of linear combinations of elements of the group. Thus each function $f \in \mathcal{F}$ can be uniquely expressed as $f(m_1, \ldots, m_k) = \prod_{i=1}^{k} m_i^{\alpha_i}$, and therefore can be identified by a proper vector $(\alpha_1, \ldots, \alpha_k) \in \mathbb{Z}^k$.
- We identify each dataset by a string fid $\in \{0,1\}^*$, and use an additional argument $i \in \{1, \ldots, n\}$ for the signing/encryption algorithm to specify that the signed/encrypted message can be used only as the i-th argument for each function $f \in \mathcal{F}$.

Definition 3 (LAEPuV). *A LAEPuV scheme is a tuple of 5 PPT algorithms (***AKeyGen***,* **AEncrypt***,* **ADecrypt***,* **AVerify***,* **AEval***) such that:*

- **AKeyGen**$(1^\lambda, k)$ *takes as input the security parameter λ, and an upper bound k for the number of messages encrypted in each dataset. It outputs a secret key sk and a public key vk (used for function evaluation and verification); the public key implicitly defines a message space $\mathcal{M}$ which is also a group, a file identifier space $\mathcal{D}$ and a ciphertext space $\mathcal{C}$.*

[6] For lack of space the formal definition is provided in the full version of this paper .

- **AEncrypt**(sk, fid, i, m) *is a probabilistic algorithm which takes as input the secret key, an element* $m \in \mathcal{M}$*, a dataset identifier fid, an index* $i \in \{1, \dots, k\}$ *and outputs a ciphertext* c*.*
- **AVerify**(vk, fid, c, f) *takes as input the public key vk, a ciphertext* $c \in \mathcal{C}$*, an identifier fid* $\in \mathcal{D}$ *and* $f \in \mathcal{F}$*. It return 1 (accepts) or 0 (rejects).*
- **ADecrypt**(sk, fid, c, f) *takes as input the secret key sk, a ciphertext* $c \in \mathcal{C}$*, an identifier fid* $\in \mathcal{D}$ *and* $f \in \mathcal{F}$ *and outputs* $m \in \mathcal{M}$ *or* $\perp$ *(if* c *is not considered valid).*
- **AEval**$(vk, f, \mathit{fid}, \{c_i\}_{i=1\dots k})$ *takes as input the public key vk, an admissible function* f *in its vector form* $(\alpha_1, \dots, \alpha_k)$*, an identifier fid, a set of* k *ciphertexts* $\{c_i\}_{i=1\dots k}$ *and outputs a ciphertext* $c \in \mathcal{C}$*. Note that this algorithm should also work if less than* k *ciphertexts are provided, as long as their respective coefficients in the function* f *are 0, but we don't explicitly account this to avoid heavy notation.*

The correctness conditions are the following:

- For any $(\mathrm{sk}, \mathrm{vk}) \leftarrow$ **AKeyGen**$(1^\lambda, k)$ honestly generated keypair, any $m \in \mathcal{M}$, any dataset identifier fid and any $i \in \{1, \dots, k\}$, with overwhelming probability

$$\mathbf{ADecrypt}(\mathrm{sk}, \mathrm{fid}, \mathbf{AEncrypt}(\mathrm{sk}, \mathrm{fid}, i, m), e_i) = m$$

  where e_i is the i-th vector of the standard basis of $\mathbb{Z}^k$.
- For any $(\mathrm{sk}, \mathrm{vk}) \leftarrow$ **AKeyGen**$(1^\lambda, k)$ honestly generated keypair, any $c \in C$

$$\mathbf{AVerify}(\mathrm{vk}, \mathrm{fid}, c, f) = 1 \iff \exists m \in \mathcal{M} : \mathbf{ADecrypt}(\mathrm{sk}, \mathrm{fid}, c, f) = m$$

- Let $(\mathrm{sk}, \mathrm{vk}) \leftarrow$ **AKeyGen**$(1^\lambda, k)$ be an honestly generated keypair, fid any dataset identifier, $c_1, \dots, c_k \in \mathcal{C}$ any tuple of ciphertexts such that $m_i =$ **ADecrypt**$(\mathrm{sk}, \mathrm{fid}, c_i, f_i)$. Then, for any admissible function $f = (\alpha_1, \dots, \alpha_k) \in \mathbb{Z}^k$, with overwhelming probability

$$\mathbf{ADecrypt}(\mathrm{sk}, \mathrm{fid}, \mathbf{AEval}(\mathrm{vk}, f, \mathrm{fid}, \{c_i\}_{i=1\dots k}), \sum_{i=0}^{k} \alpha_i f_i) = f(m_1, \dots, m_k)$$

Security definitions for LAEPuV are easy to derive, so, for lack of space, are omitted. We refer the reader to the full version of this paper.

3.1 An instantiation supporting Paillier's encryption

Let (**HKeyGen**, **HSign**, **HVerify**, **HEval**) be a secure[7] linearly homomorphic signature scheme whose message space is $\mathbb{Z}_N$ (where N is the product of two distinct (safe) primes). Moreover, let $\mathcal{H}$ be a family of collision resistant hash functions (whose images can be interpreted as elements of $\mathbb{Z}^*_{N^2}$). Then we can construct a LAEPuV scheme as follows.

[7] Again, for lack of space, security definition for linearly homomorphic signatures is provided in the full version of the paper .

AKeyGen$(1^\lambda, k)$: Choose two primes p, q of size $\lambda/2$, set $N \leftarrow pq$ and choose a random element $g \in \mathbb{Z}_{N^2}^*$ of order N. Run[8] **HKeyGen**$(1^\lambda, k, N)$ to obtain a signing key sk' and a verification key vk'. Pick a hash function $H \leftarrow \mathcal{H}$. Return $\mathrm{vk} \leftarrow (\mathrm{vk}', g, N, H)$ as the public verification key and $\mathrm{sk} = (\mathrm{sk}', p, q)$ as the secret signing key.

AEncrypt$(\mathrm{sk}, m, \mathrm{fid}, i)$: Choose random $\beta \leftarrow \mathbb{Z}_{N^2}^*$, compute $C \leftarrow g^m \beta^N \mod N^2$. Set $R \leftarrow H(\mathrm{fid}||i)$, and use the factorization of N to compute $(a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ such that $g^a b^N = RC \mod N^2$. Compute $\sigma \leftarrow$ **HSign**$(sk', \mathrm{fid}, i, a)$ and return $c = (C, a, b, \sigma)$.

AVerify$(\mathrm{vk}, \mathrm{fid}, c, f)$: Parse $c = (C, a, b, \sigma)$ and $\mathrm{vk} \leftarrow (\mathrm{vk}', g, N, H)$, then check that:

$$\mathbf{HVerify}(\mathrm{vk}', \mathrm{fid}, a, f, \sigma) = 1$$

$$g^a b^N = C \prod_{i=1}^{k} H(\mathrm{fid}||i)^{f_i} \mod N^2$$

If both the above equations hold output 1, else output 0.

ADecrypt$(\mathrm{sk}, \mathrm{fid}, c, f)$: If **AVerify**$(\mathrm{vk}, \mathrm{fid}, c, f) = 0$, return $\perp$. Otherwise, use the factorization of N to compute (m, β) such that $g^m \beta^N = C \mod N^2$ and return m.

AEval$(\mathrm{vk}, \alpha, \mathrm{fid}, c_1, \ldots, c_k)$: Parse $\alpha = (\alpha_1, \ldots, \alpha_k)$ and $c_i = (C_i, a_i, b_i, \sigma_i)$, set

$$C \leftarrow \prod_{i=i}^{k} C_i^{\alpha_i} \mod N^2, \quad a \leftarrow \sum_{i=i}^{k} a_i \alpha_i \mod N,$$

$$b \leftarrow \prod_{i=i}^{k} b_i^{\alpha_i} \mod N^2, \quad \sigma \leftarrow \mathbf{HEval}(vk', \mathrm{fid}, f, \{\sigma_i\}_{i=1,\ldots,k})$$

and return $c = (C, a, b, \sigma)$.

Remark 1. **(Supporting datasets of arbitrary size)**. In the construction above the number k of ciphertexts supported by each dataset needs to be fixed once and for all at setup time. This might be annoying in practical scenarios where more flexibility is preferable. We remark, that in the random oracle model, the scheme can be straightforwardly modified in order to remove this limitation. The idea would be to use the random oracle *also* in the underlying (homomorphic) signature scheme (see the full version of this paper for details) More precisely, rather than publishing the h_i as part of the public key, one computes different h_i's on the fly for each dataset by setting $h_i = H'(fid, i)$(where H' is some appropriate random oracle). Slightly more in detail, the elements from dataset fid are then authenticated by replacing the h_i with $h_{\mathrm{fid},i} = H'(\mathrm{fid}, i)$.

[8] Notice that the signature scheme must support $\mathbb{Z}_N$ as underlying message space. This is why we give N to the **HKeyGen** algorithm as additional parameter. Note that, this means that, in general, the signature algorithm cannot not use the factorization of N as part of its private key.

Using this simple trick brings the additional benefit that the public key can be reduced to constant size.

The security of the scheme is provided by the following theorems (whose proofs appear in the full version).

Theorem 1. *Assuming that the DCRA holds, if* $(\mathbf{HKeyGen}, \mathbf{HSign}, \mathbf{HVerify}, \mathbf{HEval})$ *is a secure linearly homomorphic signature scheme for messages in* $\mathbb{Z}_N$ *and* H *is a random oracle, the scheme described above is LH-IND-CCA secure.*

Theorem 2. *If* $\Sigma = (\mathbf{HKeyGen}, \mathbf{HSign}, \mathbf{HVerify}, \mathbf{HEval})$ *is a secure linearly homomorphic signature scheme for messages in* $\mathbb{Z}_N$ *then the scheme described above is LH-Uf-CCA secure.*

Remark 2. **(Instantiating the underlying signature scheme)**. As a concrete instantiation of the linearly homomorphic signature scheme ($\mathbf{HKeyGen}$, $\mathbf{HSign}, \mathbf{HVerify}$, $\mathbf{HEval}$),one can use use a simple variant of the (Strong) RSA based scheme from [15] adapted to use $\mathbb{Z}_N$ as underlying message space, see the full version for details.

3.2 A General Result

In this section we show how to generalize our results to support arbitrary encryption schemes satisfying some well defined homomorphic properties.
In such schemes, the message, randomness and ciphertext spaces are assumed to be finite groups, respectively denoted with $\mathcal{M}, \mathcal{R}, \mathcal{C}$ (the key spaces are treated implicitly). To adhere with the notation used in the previous section, we will denote the operation over $\mathcal{M}$ additively and the ones over $\mathcal{R}$ and $\mathcal{C}$ multiplicatively. We assume $\mathcal{T}$ to be an IND-CPA secure public key encryption scheme satisfying the following additional properties:

- We require the group operation and the inverse of an element to be efficiently computable over all groups, as well as efficient sampling of random elements. The integer linear combinations are thus defined and computed by repeatedly applying these operations.
- For any $m_1, m_2 \in \mathcal{M}, r_1, r_2 \in \mathcal{R}$, any valid public key pk it holds

$$\mathbf{Enc}_{\mathrm{pk}}(m_1, r_1) \cdot \mathbf{Enc}_{\mathrm{pk}}(m_2, r_2) = \mathbf{Enc}_{\mathrm{pk}}(m_1 + m_2, r_1 \cdot r_2)$$

- For any honest key pair (pk, sk) and any $c \in \mathcal{C}$ there exists $m \in \mathcal{M}$ and $r \in \mathcal{R}$ such that $\mathbf{Enc}_{\mathrm{pk}}(m, r) = c$ (i.e. the encryption function is surjective over the group $\mathcal{C}$). Moreover, we assume that such m and r are efficiently computable given the secret key.

Now, let $(\mathbf{HKeyGen}, \mathbf{HSign}, \mathbf{HVerify}, \mathbf{HEval})$ be a secure linearly homomorphic signature scheme for elements in $\mathcal{M}$, let $\mathcal{H}$ be a family of collision resistant hash functions $H_K : \{0,1\}^* \to \mathcal{C}$ and let $\mathcal{T} = \{\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}\}$ be an encryption scheme as above.

We construct a LAEPuV scheme as follows:

AKeyGen$(1^\lambda, k)$: Run **HKeyGen**$(1^\lambda, k)$ to obtain a signing key sk' and a verification key vk' and **Gen**(1^λ) to obtain a public key $\overline{\mathrm{pk}}$ and a secret key $\overline{\mathrm{sk}}$. Pick a hash function $H \leftarrow \mathcal{H}$. Return $\mathrm{vk} \leftarrow (\mathrm{vk}', \overline{\mathrm{pk}}, H)$ as the public verification key and $\mathrm{sk} = (\mathrm{sk}', \overline{\mathrm{sk}})$ as the secret key.

AEncrypt$(\mathrm{sk}, m, \mathrm{fid}, i)$: Choose random $r \leftarrow \mathcal{R}$, compute $C \leftarrow \mathbf{Enc}_{\overline{\mathrm{pk}}}(m, r)$ and compute, using the secret key sk, $\overline{m}$ and $\overline{r}$ such that $\mathbf{Enc}_{\overline{\mathrm{pk}}}(\overline{m}, \overline{r}) = H(\mathrm{fid}||i)$. Compute $\sigma \leftarrow \mathbf{HSign}(sk', \mathrm{fid}, i, m + \overline{m})$ and return $c = (C, m + \overline{m}, r \cdot \overline{r}, \sigma)$.

AVerify$(\mathrm{vk}, \mathrm{fid}, c, f)$: Parse $c = (C, a, b, \sigma)$ and $\mathrm{vk} \leftarrow (\mathrm{vk}', \overline{\mathrm{pk}})$, then check that:

$$\mathbf{HVerify}(\mathrm{vk}', \mathrm{fid}, a, f, \sigma) = 1$$

$$\mathbf{Enc}_{\overline{\mathrm{pk}}}(a, b) = C \prod_{i=1}^{k} H(\mathrm{fid}||i)^{f_i}$$

If both the above equations hold output 1, else output 0.

ADecrypt$(\mathrm{sk}, \mathrm{fid}, c, f)$: Parse $c = (C, a, b, \sigma)$. If **AVerify**$(\mathrm{vk}, \mathrm{fid}, c, f) = 0$, return $\perp$. Otherwise, use the secret key $\overline{\mathrm{sk}}$ to compute $m \leftarrow \mathbf{Dec}_{\overline{\mathrm{sk}}}(C)$

AEval$(\mathrm{vk}, \alpha, \mathrm{fid}, c_1, \ldots, c_k)$: Parse $\alpha = (\alpha_1, \ldots, \alpha_k)$ and $c_i = (C_i, a_i, b_i, \sigma_i)$, set

$$C \leftarrow \prod_{i=i}^{k} C_i^{\alpha_i}, \quad a \leftarrow \sum_{i=i}^{k} a_i \alpha_i,$$

$$b \leftarrow \prod_{i=i}^{k} b_i^{\alpha_i}, \quad \sigma \leftarrow \mathbf{HEval}(vk', \mathrm{fid}, f, \{\sigma_i\}_{i=1,\ldots,k})$$

and return $c = (C, a, b, \sigma)$.

Theorem 3. *Assuming $\mathcal{T}$ is an is IND-CPA secure public key encryption scheme satisfying the conditions detailed above,* (**HKeyGen**, **HSign**, **HVerify**, **HEval**) *is a secure linearly homomorphic signature scheme supporting $\mathcal{M}$ as underlying message space and H is a random oracle, then the scheme described above has indistinguishable encryptions.*

Theorem 4. *If $\Sigma =$* (**HKeyGen**, **HSign**, **HVerify**, **HEval**) *is a secure linearly homomorphic signature scheme for messages in $\mathcal{M}$ then the scheme described above is unforgeable.*

Proofs of theorems 3 and 4 are almost identical to the (corresponding) proofs of theorems 1 and 2 and are thus omitted.

4 Linearly homomorphic signature scheme to sign elements in bilinear groups

Here we introduce the notion of linearly homomorphic signature scheme to sign elements in bilinear groups. This essentially adapts the definition from [21] to support a bilinear group as underlying message space. The formal definition is given in the full version of the paper.

4.1 A random message secure construction

Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p such that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map and $\mathcal{S} = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$ a standard signature with message space $\mathcal{M}$. The scheme works as follows:

HKeyGen$(1^\lambda, 1, k)$: Choose a random generator $g \in \mathbb{G}$ and run $\mathbf{KeyGen}(1^\lambda)$ to obtain a signing key sk_1 and a verification key vk_1. Pick random $w \xleftarrow{\$} \mathbb{Z}_p$ and set $W \leftarrow g^w$. Select random group elements $h_1, \dots, h_k, \xleftarrow{\$} \mathbb{G}$.
Set $\mathrm{vk} \leftarrow (\mathrm{vk}_1, g, W, h_1, \dots, h_k)$ as the public verification key and $\mathrm{sk} = (\mathrm{sk}_1, w)$ as the secret signing key.

HSign$(\mathrm{sk}, m, \mathrm{fid}, i)$: This algorithm stores a list $\mathcal{L}$ of all previously returned dataset identifiers fid (together with the related secret information r and public information σ, τ defined below) and works as follows

If fid $\notin \mathcal{L}$, then choose $r \xleftarrow{\$} \mathbb{Z}_p$, set $\sigma \leftarrow g^r$, $\tau \leftarrow \mathbf{Sign}(\mathrm{sk}, \mathrm{fid}, \sigma)$

else if fid $\in \mathcal{L}$, then retrieve the associated r, σ, τ from memory.

Then set $M \leftarrow m^w, V \leftarrow (h_i M)^r$ (if a signature for the same fid and the same index i was already issued, then abort). Finally output $\pi \leftarrow (\sigma, \tau, V, M)$ as a signature for m w.r.t. the function e_i (where e_i is the i-th vector of the canonical basis of $\mathbb{Z}^n$).

HVerify$(\mathrm{vk}, \pi, m, \mathrm{fid}, f)$: Parse the signature π as (σ, τ, V, M) and f as $(f_1, \dots, f_k)$. Then check that:

$$\mathbf{Verify}(\mathrm{vk}, \tau, (\mathrm{fid}, \sigma)) = 1 \qquad e(M, g) = e(m, W) \qquad e(V, g) = e(\prod_{i=1}^{k} h_i^{f_i} M, \sigma)$$

If all the above equations hold output 1, else output 0.

HEval $(\mathrm{vk}, \alpha, \pi_1, \dots, \pi_k)$: Parse α as $(\alpha_1, \dots, \alpha_k)$ and π_i as $(\sigma_i, \tau_i, V_i, M_i)$, $\forall i = 1, \dots, k$. Then, compute $V \leftarrow \prod_{i=1}^{k} V_i^{\alpha_i}$, $M \leftarrow \prod_{i=1}^{k} M_i^{\alpha_i}$ and output $\pi = (\sigma_1, \tau_1, V, M)$ (or $\perp$ if the σ_i are not all equal).

The security of the scheme follows from the following theorem (whose proof is deferred to the full version of this paper).

Theorem 5. *If the 2-3CDH assumption holds and $\mathcal{S}$ is a signature scheme unforgeable under adaptive chosen message attack then the scheme described above is a LHSG scheme secure against a random message attack .*

Remark 3. If the application considered allows the fid to be a group element and not simply a string, we can replace the signature $\mathcal{S}$ with a Structure preserving Signature satisfying the same hypothesis of theorem 5. This allows to obtain the first example of a linearly homomorphic structure preserving signature scheme (LHSPS) where all parts of the signature are actually elements of the group. This is in contrast with the construction from [30], where the fid is inherently used as a bit string. In addition, if the identifier can be chosen at random by the signer and not by the adversary, we can even define σ to be the identifier itself and thus further improve efficiency. In practical instantiation it's possible to use the SPS of [1].

5 Applications to On-Line/Off-Line Homomorphic Signatures

In this section, we show a general construction to build (efficient) on-line/off-line homomorphic (and network coding) signature schemes by combining a LHSG unforgeable against a random message attack (like the one described in section 4.1) with a certain class of sigma protocols. The intuitive idea is that in order to sign a certain message m, one can choose a Σ-Protocol whose challenge space contains m, then sign the first message of the Σ-Protocol with a standard signature (this can be done off-line) and use knowledge of the witness of the protocol to later compute the response (third message) of the protocol associated to the challenge m. This is secure because, if an adversary could produce a second signature with respect to the same first message, by the special soundness of the Σ-Protocol, he would be able to recover the witness itself. We show how, if both the signature scheme and the Σ-Protocol have specific homomorphic properties, this construction can be extended to build (linearly) homomorphic signatures as well.
Informally the properties we require from the underlying sigma protocol are: (1) it is linearly homomorphic, (2) its challenge space can be seen as a vector space and (3) the third message of the protocol can be computed in a very efficient way (as it is used in the online phase of the resulting scheme). In what follows, we first adapt the definition of linearly homomorphic signature (LHSG) to the On-line/Off-line case. Then, we formally define the properties required by the sigma protocol, and we describe (and prove secure) our construction.

Linearly Homomorphic On-line/Off-line signatures First, we remark that the only difference between a LHSG and a LHOOS is in the signing algorithm. When signing m the latter can use some data prepared in advance (by running a dedicated algorithm **OffSign**) to speed up the signature process. The definitions of unforgeability are therefore analogous to the ones of traditional LHSG schemes and are omitted to avoid repetition[9].

Definition 4 (LHOOS). *A Linearly Homomorphic On-line/Off-line signature scheme is a tuple of PPT algorithms (***KeyGen***,* **OffSign***,* **OnSign***,* **Verify***,* **Eval***) such that:*

– **KeyGen**$(1^\lambda, n, k)$ *takes as input the security parameter λ, an integer n denoting the length of vectors to be signed and an upper bound k for the number of messages signed in each dataset. It outputs a secret signing key sk and a public verification key vk; the public key implicitly defines a message space that can be seen as a vector space of the form $\mathcal{M} = \mathbb{F}^n$ (where $\mathbb{F}$ is a field), a file identifier space $\mathcal{D}$ and a signature space Σ.*

[9] We stress, however, that those definitions are stronger than the ones traditionally introduced for network coding (i.e. the adversary is more powerful and there are more types of forgeries), and therefore our efficient instantiation perfectly integrates in that framework.

- **OffSign**(sk) *takes as input the secret key and outputs some information* I.
- **OnSign**$(sk, \mathit{fid}, I, \mathbf{m}, i)$ *takes as input the secret key, an element* $\mathbf{m} \in \mathcal{M}$, *an index* $i \in \{1, \dots, k\}$, *a dataset identifier fid and an instance of* I *output by* **OffSign**. *This algorithm must ensure that all the signatures issued for the same fid are computed using the same information* I *(i.e. by associating each fid with one specific* I *and storing these couples on a table). It outputs a signature* σ.
- **Verify** $(vk, \sigma, \mathbf{m}, \mathit{fid}, f)$ *takes as input the public key vk, a signature* $\sigma \in \Sigma$, *a message* $\mathbf{m} \in \mathcal{M}$, *a dataset identifier fid* $\in \mathcal{D}$ *and a function* $f \in \mathbb{Z}^k$; *it outputs* 1 *(accept) or 0 (reject).*
- **Eval**$(vk, \mathit{fid}, f, \{\sigma_i\}_{i=1\dots k})$ *takes as input the public key vk, a dataset identifier fid, an admissible function* f *in its vector form* $(\alpha_1, \dots, \alpha_k)$, *a set of* k *signatures* $\{\sigma_i\}_{i=1\dots k}$ *and outputs a signature* $\sigma \in \Sigma$. *Note that this algorithm should also work if less than k signatures are provided, as long as their respective coefficients in the function* f *are 0, but we don't to explicitly account this to avoid heavy notation.*

The correctness conditions of our scheme are the following:

- Let $(\mathrm{sk}, \mathrm{vk}) \leftarrow \mathbf{KeyGen}(1^\lambda, n, k)$ be an honestly generated keypair, $\mathbf{m} \in \mathcal{M}$, fid any dataset identifier and $i \in 1, \dots, k$. If $\sigma \leftarrow \mathbf{Sign}(\mathrm{sk}, \mathrm{fid}, \mathbf{OffSign}(sk), \mathbf{m}, i)$, then with overwhelming probability
$$\mathbf{Verify}(\mathrm{vk}, \sigma, \mathbf{m}, \mathrm{fid}, e_i) = 1,$$
where e_i is the i^{th} vector of the standard basis of $\mathbb{Z}^k$.
- Let $(\mathrm{sk}, \mathrm{vk}) \leftarrow \mathbf{KeyGen}(1^\lambda, n, k)$ be an honestly generated keypair, $\mathbf{m}_1, \dots, \mathbf{m}_k \in \mathcal{M}$ any tuple of messages signed (or derived from messages originally signed) w.r.t the same fid (and therefore using the same offline information I), and let $\sigma_1, \dots, \sigma_k \in \Sigma$, $f_1, \dots, f_k \in \mathcal{F}$ such that for all $i \in \{1, \dots, k\}$, $\mathbf{Verify}(\mathrm{vk}, \sigma_i, \mathbf{m}_i, \mathrm{fid}, f_i) = 1$. Then, for any admissible function $f = (\alpha_1, \dots, \alpha_k) \in \mathbb{Z}^k$, with overwhelming probability
$$\mathbf{Verify}(\mathrm{vk}, \mathbf{Eval}(\mathrm{vk}, \mathrm{fid}, f, \{\sigma_i\}_{i=1\dots k}), f(\mathbf{m}_1, \dots, \mathbf{m}_k), \mathrm{fid}, \sum_{i=0}^{k} \alpha_i f_i) = 1$$

Vector and Homomorphic *Σ*-protocols Informally, a Σ-Protocol can be described as a tuple of four algorithms (**Σ-Setup**, **Σ-Com**, **Σ-Resp**, **Σ-Verify**), where the first one generates a statement/witness couple, **Σ-Com** and **Σ-Resp** generate the first and third message of the protocol, and **Σ-Verify** is used by the verifier to decide on the validity of the proof (a more formal and detailed description is given in the full version of this paper). This notion can be extended to the vector case[10]. For this purpose we adapt the notion of Homomorphic Identification Protocol originally introduced in [2] to the Sigma protocol framework.

[10] The intuition is that it should be more efficient to run a vector Σ-Protocol once than a standard Σ-Protocol multiple times in parallel)

Given a language L and an integer $n \in \mathbb{N}$, we can consider the language $L^n = \{(x_1, \dots, x_n) \mid x_i \in L \ \forall i = 1, \dots, n\}$. A natural witness for a tuple (vector) in this language is the tuple of the witnesses of each of its components for the language L. As before we can consider the relation $\mathcal{R}^n$ associated to L^n, where $(\boldsymbol{x}, \boldsymbol{w}) = (x_1, \dots, x_n, w_1, \dots, w_n) \in \mathcal{R}^n$ if $(x_1, \dots, x_n)$ is part of L^n and w_i is a witness for x_i. A *vector* Σ-Protocol for $\mathcal{R}^n$ is a three round protocol defined similarly as above with the relaxation that the special soundness property is required to hold in a weaker form. Namely, we require the existence of an efficient extractor algorithm $\mathbf{\Sigma_n}$**-Ext** such that $\forall \boldsymbol{x} \in L^n$, $\forall\ R, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{c'}, \boldsymbol{s'}$ such that $(c, s) \neq (c', s')$, $\mathbf{\Sigma_n\text{-Verify}}(\boldsymbol{x}, R, \boldsymbol{c}, \boldsymbol{s}) = 1$ and $\mathbf{\Sigma_n\text{-Verify}}(\boldsymbol{x}, R, \boldsymbol{c'}, \boldsymbol{s'}) = 1$, outputs $(x, w) \leftarrow \mathbf{\Sigma_n\text{-Ext}}(\boldsymbol{x}, R, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{c'}, \boldsymbol{s'})$ where x is one of the components of $\boldsymbol{x}$ and $(x, w) \in \mathcal{R}$.
Another important requirement for our construction to work is the following property.

Definition 5. *A Σ-Protocol $\Sigma = (\mathbf{\Sigma\text{-Setup}}, \mathbf{\Sigma\text{-Com}}, \mathbf{\Sigma\text{-Resp}}, \mathbf{\Sigma\text{-Verify}})$ for a relation $\mathcal{R}$ is called* group homomorphic *if*

- *The outputs of the* $\mathbf{\Sigma}$**-Com** *algorithm and the challenge space of the protocol can be seen as elements of two groups $(\mathbb{G}_1, \odot)$ and $(\mathbb{G}_2, \otimes)$ respectively*
- *There exists a PPT algorithm* **Combine** *such that, for all $(x, w) \in \mathcal{R}$ and all $\alpha \in \mathbb{Z}^n$, if transcripts $\{(R_i, c_i, s_i)\}_{i=1,\dots,n}$ are such that $\mathbf{\Sigma\text{-Verify}}(x, R_i, c_i, s_i) = 1$ for all i, then*

$$\mathbf{\Sigma\text{-Verify}}\left(x, \bigodot_{i=1}^{n} R_i^{\alpha_i}, \bigotimes_{i=1}^{n} c_i^{\alpha_i}, \mathbf{Combine}(x, \alpha, \{(R_i, c_i, s_i)\}_{i=1,\dots,n})\right) = 1$$

Although it is given for the standard case, this property can easily be extended to vector Σ-Protocols: in particular, the group $\mathbb{G}_2$ can be seen as the group of vectors of elements taken from another group $\mathbb{G}$. To sum up, we define a class of vector Σ-Protocols having all the properties required by our construction:

Definition 6 (1-n (vector) Σ-Protocol). *Let $(\mathbb{G}_1, \odot)$, $(\mathbb{G}_2, \otimes)$ be two computational groups. A 1-n vector sigma protocol consists of four PPT algorithm $\Sigma_n = (\mathbf{\Sigma_n\text{-Setup}}, \mathbf{\Sigma_n\text{-Com}}, \mathbf{\Sigma_n\text{-Resp}}, \mathbf{\Sigma_n\text{-Verify}})$ defined as follows:*

$\mathbf{\Sigma_n\text{-Setup}}(1^\lambda, n, \mathcal{R}^n) \rightarrow (\mathbf{x}, \mathbf{w})$. *It takes as input a security parameter λ, a vector size n and a relation $\mathcal{R}^n$ over a language L^n. It returns a vector of statements and witnesses $(x_1, \dots, x_n, w_1, \dots, w_n)$. The challenge space is required to be* **ChSp**$\subseteq \mathbb{G}_2^n$.

$\mathbf{\Sigma_n\text{-Com}}(\mathbf{x}) \rightarrow (R, r)$. *It's a PPT algorithm run by the prover to get the first message R to send to the verifier and some private state to be stored. We require that $R \in \mathbb{G}_1$.*

$\mathbf{\Sigma_n\text{-Resp}}(\mathbf{x}, \mathbf{w}, r, \mathbf{c}) \rightarrow s$. *It's a deterministic algorithm run by the prover to compute the last message of the protocol. It takes as input the statements and witnesses $(\mathbf{x}, \mathbf{w})$ the challenge string $\mathbf{c} \in$***ChSp** *(sent as second message of the protocol) and some state information r. It outputs the third message of the protocol, s.*

$\mathbf{\Sigma_n}$**-Verify**$(\mathbf{x}, R, \mathbf{c}, s) \rightarrow \{0,1\}$. *It's the verification algorithm that on input the message R, the challenger* $\mathbf{c} \in$**ChSp** *and a response s, outputs* 1 *(accept) or* 0 *(reject).*

We require this protocol to be group homomorphic and to satisfy the completeness and special honest verifier zero knowledge properties. Moreover, the protocol must guarantee either the vector special soundness outlined above or a stronger soundness property that we define below.

Roughly speaking, this property requires that the extractor, upon receiving the witnesses for all but one statements of the vector $\boldsymbol{x}$, has to come up with a witness for the remaining one.

Definition 7 (Strong (Vector) Special Soundness). *Let* $\Sigma = (\mathbf{\Sigma}$**-Setup**, $\mathbf{\Sigma}$**-Com**, $\mathbf{\Sigma}$**-Resp**, $\mathbf{\Sigma}$**-Verify**) *be a* 1-n Σ*-Protocol for a relation* $\mathcal{R}^n$*. We say that* Σ *has the* Strong Special Soundness *property if there exists an efficient extractor algorithm* $\mathbf{\Sigma_n}$**-Ext** *such that* $\forall\, \boldsymbol{x} \in L^n, \forall j^* \in \{1,\ldots,n\}$*,* $\forall\ R, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{c'}, \boldsymbol{s'}$ *such that* $c_{j^*} \neq c'_{j^*}$*,* $\mathbf{\Sigma_n}$**-Verify**$(\boldsymbol{x}, R, \boldsymbol{c}, \boldsymbol{s}) = 1$ *and* $\mathbf{\Sigma_n}$**-Verify**$(\boldsymbol{x}, R, \boldsymbol{c'}, \boldsymbol{s'}) = 1$*, outputs* $w_{j^*} \leftarrow \mathbf{\Sigma_n}$**-Ext**$(\boldsymbol{x}, R, \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{c'}, \boldsymbol{s'}, \{w_j\}_{j \neq j^*})$ *such that* $(x_{j^*}, w_{j^*}) \in \mathcal{R}$*.*

In the full version of this paper we show that a simple variant of the well known identification protocol by Schnorr is a 1-n Σ-Protocol (with Strong Vector Special Soundness).

5.1 A Linearly Homomorphic On-Line/Off-Line Signature

Suppose $\mathcal{S} = ($**KeyGen**, **Sign**, **Verify**, **Eval**$)$ is a randomly secure LHSG (even one that only allows to sign scalars), $\Sigma_n = (\mathbf{\Sigma_n}$**-Setup** ,$\mathbf{\Sigma_n}$**-Com** ,$\mathbf{\Sigma_n}$**-Resp** ,$\mathbf{\Sigma_n}$**-Verify**) is a 1-$n$ Σ-Protocol and $\mathcal{H} = ($**CHGen**, **CHEval**, **CHFindColl**$)$ defines a family of chameleon hash functions. Moreover, suppose that the LHSG's message space is the same as the group $\mathbb{G}_1$ of the outputs of $\mathbf{\Sigma_n}$**-Com**. Our generic construction uses the challenge space of the Σ-Protocol as a message space and works as follows:

ON/OFFKeyGen $(1^\lambda, k, n)$: It runs $(\mathrm{vk}_1, sk_1) \leftarrow$ **KeyGen**$(1^\lambda, 1, k)$, $(\mathbf{x}, \mathbf{w}) \leftarrow \mathbf{\Sigma_n}$**-Setup** $(1^\lambda, n, \mathcal{R}^n)$ and $(\mathrm{hk}, \mathrm{ck}) \leftarrow$ **CHGen**(1^λ). It outputs $\mathrm{vk} \leftarrow (vk_1, \mathbf{x}, \mathrm{hk})$, $\mathrm{sk} \leftarrow (sk_1, \mathbf{w}, \mathrm{ck})$.

OFFSign (sk): This algorithm runs the $\mathbf{\Sigma_n}$**-Com** algorithm k times to obtain $(R_i, r_i) \leftarrow \mathbf{\Sigma_n}$**-Com** $(\mathbf{x})$, chooses a random string fid' from the dataset identifiers' space and randomness ρ' and sets $\overline{\mathrm{fid}} \leftarrow$ **CHEval**$(\mathrm{hk}, \mathrm{fid}', \rho')$. Then it signs each R_i using the LHSG signing algorithm $\overline{\sigma_i} \leftarrow$ **Sign**$(sk_1, R_i, \overline{\mathrm{fid}}, i)$ and outputs $I_{\mathrm{fid}'} = \{(i, r_i, R_i, \overline{\sigma_i}, \mathrm{fid}', \rho')\}_{i=1,\ldots,k}$.

ONSign $(\mathrm{vk}, \mathrm{sk}, \mathbf{m}, \mathrm{fid}, I_{\mathrm{fid}'}, i)$: It parses $I_{\mathrm{fid}'}$ as $\{(i, r_i, R_i, \overline{\sigma_i}, \mathrm{fid}', \rho')\}_{i=1,\ldots,k}$, computes $s \leftarrow \mathbf{\Sigma_n}$**-Resp** $(\mathbf{x}, \mathbf{w}, r_i, \mathbf{m})$, $\rho \leftarrow$ **CHFindColl**$(\mathrm{ck}, \mathrm{fid}', \rho', \mathrm{fid})$ and outputs $\sigma \leftarrow (R_i, \overline{\sigma}_i, s, \rho)$. As explained in the definition, this algorithm must ensure that all the messages signed with respect to the same fid are computed from the same information $I_{\mathrm{fid}'}$

ON/OFFVerify $(\mathrm{vk}, \sigma, \mathbf{m}, \mathrm{fid}, f)$: It parses σ as $(R, \overline{\sigma}, s, \rho)$ and vk as $(\mathrm{vk}_1, \mathbf{x}, \mathrm{hk})$. Then it checks that

$$\mathbf{Verify}(\mathrm{vk}_1, \overline{\sigma}, R, \mathbf{CHEval}(\mathrm{fid}, \rho), f) = 1 \quad \text{and} \quad \mathbf{\Sigma_n\text{-}Verify}(\mathbf{x}, R, \mathbf{m}, s) = 1.$$

If both the above equations hold it returns 1, else it returns 0.

ON/OFFEval $(\mathrm{vk}, \alpha, \sigma_1, \ldots, \sigma_k)$: it parses σ_i as $(R_i, \overline{\sigma}_i, s_i, \rho)$ for each $i = 1, \ldots, k$ and vk as $(\mathrm{vk}_1, \mathbf{x})$. Then it computes:

$$R \leftarrow R_1^{\alpha_1} \odot \cdots \odot R_k^{\alpha_k}, \quad \overline{\sigma} \leftarrow \mathbf{Eval}(\mathrm{vk}_1, \alpha, \overline{\sigma}_1, \ldots, \overline{\sigma}_k),$$

$$s \leftarrow \mathbf{Combine}\left(\boldsymbol{x}, \alpha, \{(R_i, c_i, s_i)\}_{i=1,\ldots,k}\right).$$

Finally it returns $(R, \overline{\sigma}, s, \rho)$ (as a signature for the message $\mathbf{m}_1^{\alpha_1} \otimes \cdots \otimes \mathbf{m}_k^{\alpha_k}$).

Remark 4. The construction presented above applies to any LHSG. However, if the LHGS itself is obtained as described in section 4.1, the use of the chameleon hash function could be avoided by substituting the signature scheme $\mathcal{S}$ used for the fid with an on-line/off-line one. This improves efficiency.

Theorem 6. *If $\mathcal{S}$ = (**KeyGen**, **Sign**, **Verify**, **Eval**) is a random message secure LHSG, Σ_n = (**Σ_n-Setup** ,**Σ_n-Com** ,**Σ_n-Resp** ,**Σ_n-Verify**) is a 1-n Σ-Protocol for a non trivial relation $\mathcal{R}^n$, and $\mathcal{H}$ implements a family of chameleon hash functions then the LHOOS described above is secure against a chosen message attack .*

For lack of space, again, this proof is omitted. The security obtained by this construction can be strengthened by assuming additional properties on the underlying LHSG scheme: if $\mathcal{S}$ is strongly secure against a random message attack, then we can prove that the resulting construction is strongly secure (against a CMA) as well.

Theorem 7. *If $\mathcal{S}$ = (**KeyGen**, **Sign**, **Verify**, **Eval**) is a LHSG scheme strongly unforgeable against a random message attack and Σ_n = (**Σ_n-Setup**, **Σ_n-Com**, **Σ_n-Resp**, **Σ_n-Verify**) is a 1-n Σ-Protocol for a non trivial relation $\mathcal{R}^n$, then the on-line/off-line scheme described above is **strongly** unforgeable against chosen message attacks.*

The proof is straightforward and similar to the previous one and is omitted.

6 From random message security to chosen message security

In this section we present a general transform to construct an LHSG secure against chosen message attack from one secure under random message attack. Our transformation requires the RMA secure scheme to satisfy some additional, but reasonable, requirements (a slightly more generic transformation is given in the full version of this paper). In particular we require it to be *almost deterministic*. Informally, this means that given a file identifier fid $\in \mathcal{D}$ and a signature on a message m with respect to fid, the signature of any other $m' \in \mathcal{M}$ w.r.t. to any admissible function $f \in \mathcal{F}$ and the same fid is uniquely determined.

Remark 5. We stress that while we present our theorems in the context of linearly homomorphic signatures (LHSG), if they are applied to linearly homomorphic structure preserving signatures, the structure preserving property is preserved.

Let $\mathcal{S} = (\mathbf{HKeyGen}, \mathbf{HSign}, \mathbf{HVerify}, \mathbf{HEval})$ be a LHSG which is RMA-secure and almost deterministic. The transformation below shows how to produce a new LHSG $\mathcal{T} = (\mathbf{TKeyGen}, \mathbf{TSign}, \mathbf{TVerify}, \mathbf{TEval})$ which is secure under CMA.

- $\mathbf{TKeyGen}(1^\lambda, n, k)$ takes as input the security parameter λ, the vector size n and an upper bound k for the number of messages signed in each dataset. It runs two times the **HKeyGen** algorithm to obtain $(\mathrm{sk}_1, \mathrm{vk}_1) \leftarrow \mathbf{HKeyGen}(1^\lambda, n, k)$ and $(\mathrm{sk}_2, \mathrm{vk}_2) \leftarrow \mathbf{HKeyGen}(1^\lambda, n, k)$.
  It outputs $\mathrm{sk} = (\mathrm{sk}_1, \mathrm{sk}_2)$ as the secret signing key and $\mathrm{vk} = (\mathrm{vk}_1, \mathrm{vk}_2)$ as the public verification key. The message space $\mathcal{M}$ is the same of $\mathcal{S}$.
- $\mathbf{TSign}(\mathrm{sk}, \mathbf{m}, \mathrm{fid}, i)$ It chooses random $\mathbf{m}_1 = (m_{1,1}, \ldots, m_{1,n}) \overset{\$}{\leftarrow} \mathcal{M}$ and computes $\mathbf{m}_2 \leftarrow \left(\frac{m_1}{m_{1,1}}, \ldots, \frac{m_n}{m_{1,n}}\right)$ (where $\mathbf{m} = (m_1, \ldots, m_n)$).
  Then it computes $\sigma_1 \leftarrow \mathbf{HSign}(\mathrm{sk}_1, \mathbf{m}_1, i, \mathrm{fid})$, $\sigma_2 \leftarrow \mathbf{HSign}(\mathrm{sk}_2, \mathbf{m}_2, i, \mathrm{fid})$ and outputs $\sigma = (\mathrm{fid}, \mathbf{m}_1, \sigma_1, \sigma_2)$.
- $\mathbf{TVerify}(\mathrm{vk}, \sigma, \mathbf{m}, \mathrm{fid}, f)$ parses σ as $(\mathrm{fid}, \mathbf{m}_1, \sigma_1, \sigma_2)$, computes $\mathbf{m}_2 \leftarrow \left(\frac{m_1}{m_{1,1}}, \ldots, \frac{m_n}{m_{1,n}}\right)$ and checks that the following equations hold:
  $$\mathbf{HVerify}(\mathrm{vk}_i, m_i, \sigma_i, \mathrm{fid}, f) = 1 \quad \text{for} \quad i = 1, 2.$$
- $\mathbf{Eval}(\mathrm{vk}, \mathrm{fid}, f, \{\sigma^{(i)}\}_{i=1\ldots k})$ parses $\sigma^{(i)}$ as $(\mathrm{fid}^{(i)}, \mathbf{m}_1^{(i)}, \sigma_1^{(i)}, \sigma_2^{(i)})$ and f as $(\alpha_1, \ldots, \alpha_k)$, then checks that $\mathrm{fid} = \mathrm{fid}^{(i)}$ for all i and, if not, aborts. Finally it sets
  $$\sigma_1 \leftarrow \mathbf{HEval}(\mathrm{vk}_1, \mathrm{fid}, \{\sigma_1^{(i)}\}_{i=1\ldots k}, f),$$
  $$\sigma_2 \leftarrow \mathbf{HEval}(\mathrm{vk}_2, \mathrm{fid}, \{\sigma_2^{(i)}\}_{i=1\ldots k}, f),$$
  $$\mathbf{m}_1 = \left(\prod_{i=1}^{k} (m_{1,1}^{(i)})^{\alpha_i}, \ldots, \prod_{i=1}^{k} (m_{1,n}^{(i)})^{\alpha_i}\right)$$
  and returns
  $$\sigma \leftarrow (\mathrm{fid}, \mathbf{m}_1, \sigma_1, \sigma_2)$$

Theorem 8. *Suppose $\mathcal{S}$ is a LHSG secure against a random message attack with almost deterministic signatures. Moreover assume that the underlying message space is a group where one can efficiently solve systems of group equations. Then the scheme $\mathcal{T}$ described above is a LHSG secure against a chosen message attack.*

Again, the proof of the above theorem is deferred to the full version of this paper.

Acknowledgements. We would like to thank Dario Fiore for helpful discussions and one of the anonymous reviewers of Crypto 2014 for suggesting us the generalized scheme presented in section 3.2.

References

1. Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, August 2011.
2. Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of storage from homomorphic identification protocols. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 319–333. Springer, December 2009.
3. Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 17–34. Springer, March 2011.
4. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 367–385. Springer, December 2012.
5. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 386–404. Springer, February / March 2013.
6. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.
7. Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87. Springer, March 2009.
8. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, May 2011.
9. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16. Springer, March 2011.
10. Dario Catalano, Mario Di Raimondo, Dario Fiore, and Rosario Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. In Ronald Cramer, editor, *PKC 2008: 11th International Conference on Theory and Practice of Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 101–120. Springer, March 2008.
11. Dario Catalano and Dario Fiore. Practical homomorphic macs for arithmetic circuits. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 336–352. Springer, May 2013.

12. Dario Catalano, Dario Fiore, Rosario Gennaro, and Luca Nizzardo. Generalizing homomorphic macs for arithmetic circuits. In Hugo Krawczyk, editor, *PKC 2014: 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 538–555. Springer, mar 2014.
13. Dario Catalano, Dario Fiore, Rosario Gennaro, and Konstantinos Vamvourellis. Algebraic (trapdoor) one-way functions and their applications. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 680–699. Springer, mar 2013.
14. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Adaptive pseudo-free groups and applications. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 207–223. Springer, May 2011.
15. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 680–696. Springer, May 2012.
16. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay; Rosario Gennaro, editor, *Advances in Cryptology – CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 371–389. Springer, August 2014.
17. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300. Springer, April 2012.
18. Yvo Desmedt. Computer security by redefining what a computer is. NSPW, 1993.
19. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552. ACM Press, May 1991.
20. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.
21. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 697–714. Springer, May 2012.
22. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194. Springer, May 2003.
23. Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 142–160. Springer, May 2010.
24. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In *Advances in Cryptology – ASIACRYPT 2013*, Lecture Notes in Computer Science, pages 301–320. Springer, December 2013.
25. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

26. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262. Springer, February 2002.
27. Chihong Joo and Aaram Yun. Homomorphic authenticated encryption secure against chosen-ciphertext attack. Cryptology ePrint Archive, Report 2013/726, 2013. http://eprint.iacr.org/.
28. Sébastien Kunz-Jacques and David Pointcheval. About the security of MTI/C0 and MQV. In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 156–172. Springer, September 2006.
29. Benoit Libert and Marc Joye. Efficient cryptosystems from 2 ^{k} -th power residue symbols. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 76–92. Springer, May 2013.
30. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti; Juan A. Garay, editor, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 289–307. Springer, August 2013.
31. Silvio Micali. Cs proofs. In *FOCS*, pages 436–453. IEEE Computer Society, 1994.
32. Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer, February 2002.
33. David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 59–66. ACM Press, November 1998.
34. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999.
35. Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer, August 2001.
36. Xun Yi. Directed transitive signature scheme. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 129–144. Springer, February 2007.