Homomorphic Network Coding Signatures in the Standard Model

Nuttapong Attrapadung[1] and Benoît Libert[2]*

[1] Research Center for Information Security, AIST (Japan)
n.attrapadung@aist.go.jp
[2] Université catholique de Louvain, Crypto Group (Belgium)
benoit.libert@uclouvain.be

Abstract. Network coding is known to provide improved resilience to packet loss and increased throughput. Unlike traditional routing techniques, it allows network nodes to perform transformations on packets they receive before transmitting them. For this reason, packets cannot be authenticated using ordinary digital signatures, which makes it difficult to hedge against pollution attacks, where malicious nodes inject bogus packets in the network. To address this problem, recent works introduced signature schemes allowing to sign linear subspaces (namely, verification can be made w.r.t. any vector of that subspace) and which are well-suited to the network coding scenario. Currently known network coding signatures in the standard model are not homomorphic in that the signer is forced to sign all vectors of a given subspace at once. This paper describes the first homomorphic network coding signatures in the standard model: the security proof does not use random oracles and, at the same time, the scheme allows signing individual vectors *on-the-fly* and has constant per-packet overhead in terms of signature size. The construction is based on the dual encryption technique introduced by Waters (Crypto'09) to prove the security of hierarchical identity-based encryption schemes.

Keywords. Network coding, homomorphic signatures, provable security, standard model.

1 Introduction

Network coding [1, 18] is an attractive paradigm that offers an interesting alternative to traditional routing mechanisms. Instead of merely storing and forwarding packets in transit, intermediate nodes are allowed to modify them: typically, at each node, outgoing packets contain vectors that are calculated as linear combinations of vectors conveyed by incoming packets. In *random linear network coding*, packets are combined using coefficients which each node chooses at random, independently of its neighbors. Still, receiving nodes are able to recover

* This author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for his "chargé de recherches" fellowship and the BCRYPT Interuniversity Attraction Pole.

the original file from any set of, say $m > 1$, valid packets containing linearly independent vectors and without *a priori* knowing the coefficients chosen by intermediate nodes on the road. This technique has been shown (see [10] for instance) to provide many advantages such as an improved resilience to random packet loss or a substantially increased throughput in certain topologies.

Unfortunately, network coding is highly sensitive to *pollution attacks*, where malicious nodes inject invalid packets (*i.e.*, nodes outside the linear span of the received packets) in the network in order to prevent target nodes from recovering the original file. Since network nodes perform linear transformations over all their incoming packets, even a single faulty packet is likely to contaminate the entire network and eventually hinder the decoding process. To address this concern, intermediate good nodes need a method to verify the validity of incoming packets and sieve out bad ones. Obviously, the problem cannot be resolved by ordinary digital signatures since transmitted packets are modified by the network and cannot be merely signed by the source. For this reason, cryptographic approaches rely on techniques allowing to authenticate packets using homomorphic hash functions [16, 13, 25] or homomorphic signatures [9, 7, 12]. These primitives are designed in such a way that a signature (resp. a hash value) of a vector $\vec{v}$ can be obtained from signatures (resp. hash values) of several vectors that $\vec{v}$ is a linear combination of.

In contrast to information-theoretic approaches (like [14, 15]) that defend against network faults by introducing redundancies in packets, cryptographic techniques do not place restrictions on the adversary's behavior (e.g. by limiting his ability to eavesdrop the network or the fraction of nodes he can corrupt): as long as the receiver obtains sufficiently many correct packets, he can always recover the file regardless of the number of faults. On the other hand, these techniques typically require computational assumptions and sometimes appeal to idealizations such as the random oracle model [4]. This paper aims at making another step towards eliminating the latter.

Related Work. Homomorphic signatures were first suggested by Johnson, Molnar, Song and Wagner [20]. Their definition was adapted to the network coding scenario by Boneh, Freeman, Katz and Waters [7] who designed an efficient homomorphic NCS scheme in the random oracle model using bilinear maps. At the expense of losing the homomorphic property, they also showed how to build a network coding signature in the standard model. In [7], signature sizes were proved asymptotically optimal since a signature on any subspace necessarily grows with the dimension of that subspace. Recently, Gennaro, Katz, Krawczyk and Rabin gave a homomorphic signature [12] based on the RSA assumption (in the random oracle model) and showed how to work with small coefficients over the integers (instead of finite fields) in networks of bounded size. At the same time, Agrawal, Boneh, Boyen and Freeman [3] considered the situation of network nodes mixing packets from multiple distinct sources and described a multi-source network coding signature (without the homomorphic property) in the standard model.

In the secret-key setting, Agrawal and Boneh [2] considered how to improve upon the speed of network coding public key signatures and designed message authentication codes with homomorphic properties. Assuming that a bounded number of verifiers may collude, they also showed how intermediate nodes can verify the integrity of network-coded data. More recently, Li *et al.* [19] gave a MAC-based approach supporting in-network verification and resisting an arbitrary number of collusions.

Our Contribution. To the best of our knowledge, in the public key setting, known *homomorphic* network coding signatures [7, 12] all rely on random oracles in their security proof. Indeed, existing NCS schemes in the standard model (*i.e.*, the second scheme of [7] and the multi-source system in [3]) can only be used to sign all the base vectors of a subspace at once. This requires the source to be aware of the entire file before sending the first packet.

This paper describes the first homomorphic NCS scheme with a security proof outside the random oracle methodology. Our construction is based on the dual encryption paradigm, introduced by Waters [24] and developed in [17], the purpose of which was initially to build fully secure (hierarchical) identity-based encryption [22, 6] schemes. We pinpoint an intuitive connection between NCS schemes and the spatial encryption primitive of Boneh and Hamburg [8], where the receiver's ability to decrypt is made contingent on his knowledge of a private key for a subspace containing the vector assigned to the ciphertext. We explain that such a scheme can be turned into a (not necessarily homomorphic) NCS scheme when the file identifier can be suitably tied up to the signed subspace. The homomorphic property is then achieved by carefully re-using the signer's random coins across all vectors of the same linear subspace: by deriving these coins from the file identifier using a pseudorandom function, the signer can start transmitting packets before the file to be sent is completely known.

In order to prove security in the sense of the definition of Boneh *et al.* [7], we use groups of composite order and apply the technique of Lewko and Waters [17] in the context of signatures. One difficulty to deal with is that, unlike previous homomorphic NCS schemes [7, 12], the system uses a randomized signing algorithm and signatures on distinct vectors must be generated using partially identical randomness in order to be linearly combinable. We thus have to take special precautions to prevent malicious nodes from re-randomizing signatures and wrongly accuse the signer of flooding the network with signatures that cannot be combined.

Since we work in groups of composite order N, vector coordinates and network coefficients must be chosen in a ring $\mathbb{Z}_N$ instead of a prime field as in [7]. Nevertheless, the scheme has counterparts in prime order groups. While Freeman's framework [11] does not seem to apply (given that it does not apply to the Lewko-Waters techniques [17], as mentioned in [11]) to generically transform the scheme into an instantiation in prime-order groups, the system can be adapted in asymmetric pairing-friendly groups of prime order in the same way as the Lewko-Waters IBE [17]. It is also translatable in groups with symmetric pairings using the techniques of [24]. In the paper, we chose to describe it in composite order groups for simplicity.

Organization. In the following, we first review the notion of network coding signatures in section 2. Our homomorphic scheme and its proof are detailed in sections 3.1 and 3.2, respectively.

2 Background and Definitions

2.1 Network Coding

This section briefly recalls the idea of linear network coding. Consider a network with one source node and a subset of nodes called "target nodes". The purpose is to have the source transmit a file to all target nodes, where a file is represented as a matrix containing the m row vectors $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{Z}_N^k$ over a ring or a field $\mathbb{Z}_N$. The source node first creates m augmented vectors $\vec{w}_1, \ldots, \vec{w}_m \in \mathbb{Z}_N^n$, with $n = k + m$, by setting

$$\begin{pmatrix} —\vec{w}_1— \\ —\vec{w}_2— \\ \vdots \\ —\vec{w}_m— \end{pmatrix} = \left(\begin{array}{c|c} —\vec{v}_1— & 1\,0 \cdots\ 0 \\ —\vec{v}_2— & 0\,1 \cdots\ 0 \\ \vdots & \ddots \\ —\vec{v}_m— & 0\,0 \cdots\ 1 \end{array}\right). \tag{1}$$

The source then sends these augmented vectors to its neighbor nodes.

We notice that the span of row vectors of the above matrix will generate a vector subspace $V \subset \mathbb{Z}_N^n$ of dimension m with the basis $\vec{w}_1, \ldots, \vec{w}_m$. As defined in [7], when the basis is in the above form (in the right-hand side of Equation (1)), it is called a *properly augmented basis*.

Each honest intermediate node in the network processes the incoming packets as follows. Upon receiving vectors $\vec{y}_1, \ldots, \vec{y}_\ell \in \mathbb{Z}_N^n$ on its ℓ incoming edges, it computes a new vector for each outgoing edge as a linear combination of the vectors it received. Namely, at the j^{th} outgoing edge, the vector $\vec{z}_j \in \mathbb{Z}_N^n$ will have the form $\vec{z}_j = \sum_{i=1}^{\ell} \alpha_{i,j} \vec{y}_i$, for some (typically random) coefficients $(\alpha_{1,j}, \ldots, \alpha_{\ell,j}) \in \mathbb{Z}_N^\ell$.

A target node will recover the file using a set of vectors from its incoming edges. This can be done if they consist of m vectors $\{\vec{y}_i = (\vec{x}_i || \vec{u}_i)\}_{i=1}^m$ where $\vec{u}_1, \ldots, \vec{u}_m$ are linearly independent (here, $\vec{x}_i \in \mathbb{Z}_N^k, \vec{u}_i \in \mathbb{Z}_N^m$). The original file is then recovered as

$$\begin{pmatrix} —\vec{v}_1— \\ —\vec{v}_2— \\ \vdots \\ —\vec{v}_m— \end{pmatrix} = \begin{pmatrix} —\vec{u}_1— \\ —\vec{u}_2— \\ \vdots \\ —\vec{u}_m— \end{pmatrix}^{-1} \begin{pmatrix} —\vec{x}_1— \\ —\vec{x}_2— \\ \vdots \\ —\vec{x}_m— \end{pmatrix},$$

which is computable thanks to to the linear independence of $\vec{u}_1, \ldots, \vec{u}_m$.

2.2 Definitions

We first recall the definition of network coding signatures from [7].

Definition 1. *A network coding signature (NCS) scheme consists of a triple of efficient algorithms* $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following specifications.*

Keygen(λ, n)**:** *is a probabilistic algorithm that takes as input a security parameter* $\lambda \in \mathbb{N}$ *and an integer* $n \in \mathsf{poly}(\lambda)$ *denoting the length of vectors to be signed. It outputs a positive integer* $N \in \mathbb{N}$*, a public key* pk*, the corresponding private key* sk *and the description of an efficiently samplable file identifier space* $\mathcal{I}$*.*

Sign$(\mathsf{sk}, \mathsf{id}, V)$**:** *is a (possibly probabilistic) algorithm that takes as input a private key* sk*, a file identifier* $\mathsf{id} \in \mathcal{I}$ *and a vector subspace* V *(described as a set of linearly independent vectors* $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{Z}_N^n$*) of dimension* $m < n$*. It outputs a signature* σ*.*

Verify$(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma)$**:** *is a deterministic algorithm that takes as input a public key* pk*, a file identifier* $\mathsf{id} \in \mathcal{I}$*, a vector* $\vec{y}$ *and a signature* σ*. It outputs* 1 *or* 0*.*

Correctness requires that, for all $\lambda \in \mathbb{N}$, all integers $n \in \mathsf{poly}(\lambda)$ and all triples $(\mathsf{pk}, \mathsf{sk}, \mathcal{I}) \leftarrow \mathsf{Keygen}(\lambda, n)$, it holds that for all $\mathsf{id} \in \mathcal{I}$ and all vector subspace $V \subset \mathbb{Z}_N^n$, if $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{id}, V)$, then $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma) = 1$ for all $\vec{y} \in V$.

In what follows, we define homomorphic network coding signature schemes. Unlike previous homomorphic schemes [7, 12], the construction in this paper uses a probabilistic signing algorithm. To make it possible to publicly combine signatures on distinct vectors from the same file, the signer has to re-use part of his random coins to sign all vectors of the subspace. As long as these signatures are generated using the appropriate coins, network nodes can always combine them. However, attention must be paid to the fact that anyone can attempt to re-randomize signatures so as to prevent them from being combinable later on and disrupt the system. For this reason, network nodes have to make sure that valid signatures of vectors from the same file were produced using compatible randomness before combining them. We thus slightly modify the specification of homomorphic NCS schemes [7] and add a compatibility-checking algorithm that allows testing whether signatures are indeed combinable.

Definition 2. *A homomorphic network coding signature scheme is a tuple of efficient algorithms* $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{CompatibilityCheck}, \mathsf{Combine}, \mathsf{Verify})$

Keygen(λ, n)**:** *is a probabilistic algorithm that takes as input a security parameter* $\lambda \in \mathbb{N}$ *and an integer* $n \in \mathsf{poly}(\lambda)$ *denoting the length of vectors to be signed. It outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$ *and the description of a file identifier space* $\mathcal{I}$*.*

Sign$(\mathsf{sk}, \mathsf{id}, \vec{v})$**:** *is a possibly randomized algorithm that takes in a private key* sk*, a file identifier* $\mathsf{id} \in \mathcal{I}$ *and a vector* $\vec{v}$*. It outputs a signature* σ*.*

CompatibilityCheck$(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{\ell})$**:** *takes as input a public key* pk*, a file identifier* id *and a set of* ℓ *signatures* $\{\sigma_i\}_{i=1}^{\ell}$*. It outputs* 1 *if these signatures are deemed compatible for combination and* 0 *otherwise.*

Combine$(\mathsf{pk}, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$**:** *is a (possibly randomized) algorithm that takes as input a public key* pk*, a file identifier* id *and* ℓ *tuples* (β_i, σ_i)*, each one of which consists of a weight* β_i *and a signature* σ_i*. Intuitively, the output is a signature* σ *on the vector* $\vec{y} = \sum_{i=1}^{\ell} \beta_i \vec{v}_i$*, where* σ_i *is a signature on* $\vec{v}_i$*.*

$\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma)$: *is a deterministic algorithm that takes as input a public key* pk*, a file identifier* $\mathsf{id} \in \mathcal{I}$*, a signature* σ *and a vector* $\vec{y}$*. It outputs* 0 *or* 1*.*

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

1. For all $\mathsf{id} \in \mathcal{I}$ and all n-vectors $\vec{y}$, if $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{id}, \vec{y})$, then we necessarily have $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma) = 1$.
2. For all $\mathsf{id} \in \mathcal{I}$, any $\ell > 0$ and any set of vectors $\{\vec{v}_i\}_{i=1}^{\ell}$, if $\sigma_i = \mathsf{Sign}(\mathsf{sk}, \mathsf{id}, \vec{v}_i)$ for $i = 1$ to ℓ, then $\mathsf{CompatibilityCheck}(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{\ell}) = 1$.
3. For all $\mathsf{id} \in \mathcal{I}$, any $\ell > 0$ and any set of triples $\{(\beta_i, \sigma_i, \vec{v}_i)\}_{i=1}^{\ell}$, if the following two conditions are satisfied

   a. $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{v}_i, \sigma_i) = 1$ for each $i \in \{1, \ldots, \ell\}$,

   b. $\mathsf{CompatibilityCheck}(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{\ell}) = 1$,

   then it holds that $\mathsf{Verify}\big(\mathsf{pk}, \mathsf{id}, \sum_{i=1}^{\ell} \beta_i \vec{v}_i, \mathsf{Combine}(\mathsf{pk}, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})\big) = 1$.

In the following, we say that signatures $\{\sigma_i\}_{i=1}^{\ell}$ are *compatible* if they correspond to the same $\mathsf{id} \in \mathcal{I}$ and if it holds that $\mathsf{CompatibilityCheck}(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{\ell}) = 1$.

When $\{\sigma_i\}_{i=1}^{\ell}$ is a set of compatible signatures, we say that σ is compatible with $\{\sigma_i\}_{i=1}^{\ell}$ if $\{\sigma\} \cup \{\sigma_i\}_{i=1}^{\ell}$ forms a set of compatible signatures. In particular, when a signature $\tilde{\sigma}$ of a subspace V consists of signatures $(\sigma_1, \ldots, \sigma_m)$ on independent vectors $\vec{v}_1, \ldots, \vec{v}_m \in V$, we say that σ is compatible with $\tilde{\sigma}$ if it is compatible with all $\{\sigma_i\}_{i=1}^{m}$.

Conversion. We recall how a homomorphic network coding signature allows signing vector subspaces, as noted in [7]. Let scheme $\Sigma_2 = (\mathsf{Keygen}_2, \mathsf{Sign}_2, \mathsf{CompatibilityCheck}_2, \mathsf{Combine}_2, \mathsf{Verify}_2)$ be a homomorphic NCS scheme. An ordinary network coding signature $\Sigma_1 = (\mathsf{Keygen}_1, \mathsf{Sign}_1, \mathsf{Verify}_1)$ can be obtained as follows.

$\mathsf{Keygen}_1(\lambda, n) =$ $\mathsf{Keygen}_2(\lambda, n)$

$\mathsf{Sign}_1(\mathsf{sk}, \mathsf{id}, V) =$ $(\sigma_1, \ldots, \sigma_m)$, where $\sigma_i = \mathsf{Sign}_2(\mathsf{sk}, \mathsf{id}, \vec{v}_i)$ for $i = 1$ to m and $\vec{v}_1, \ldots, \vec{v}_m$ is a properly augmented basis of $V \subseteq \mathbb{Z}_N^n$.

$\mathsf{Verify}_1(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma) =$ outputs 1 if and only if

$$\begin{cases} \mathsf{CompatibilityCheck}_2(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{m}) = 1 \\ \mathsf{Verify}_2(\mathsf{pk}, \mathsf{id}, \vec{y}, \mathsf{Combine}_2(\mathsf{pk}, \mathsf{id}, \{(y_{n-m+i}, \sigma_i)\}_{i=1}^{m})) = 1. \end{cases}$$

Security. The security definition hereafter slightly generalizes the one of [7]. It requires that it be infeasible to publicly destroy the "combinability" of valid signatures without rendering them invalid when they are considered individually. Our goal is to guarantee that, if valid signatures of several vectors from the same file have incompatible randomness, the signer is necessarily deviating from the specification of the scheme. When such a bogus or misbehaving signer is detected, honest network nodes may simply stop processing their packets.

Definition 3. *A network coding signature scheme $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ 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 polynomially-many occasions, $\mathcal{A}$ chooses a linear subspace $V_i \subset \mathbb{Z}_N^n$ of dimension $m_i < n$. The challenger replies by choosing a file identifier $\mathsf{id}_i \in \mathcal{I}$ from the identifier space $\mathcal{I}$ and returns id_i and $\sigma_i = \mathsf{Sign}(\mathsf{sk}, \mathsf{id}_i, V_i)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs an identifier $\mathsf{id}^\star$, a signature $\sigma^\star$ and a vector $\vec{y} \in \mathbb{Z}_N^n$. The adversary $\mathcal{A}$ is deemed successful if $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}^\star, \vec{y}^\star, \sigma^\star) = 1$ and either of the following holds:*
   - ◦ *(Class i): $\mathsf{id}^\star \neq \mathsf{id}_i$ for any i and $\vec{y}^\star \neq \vec{0}$.*
   - ◦ *(Class ii): $\mathsf{id}^\star = \mathsf{id}_i$ for some $i \in \{1, \ldots, q\}$ and the signature $\sigma^\star$ is not compatible with σ_i.*
   - ◦ *(Class iii): $\mathsf{id}^\star = \mathsf{id}_i$ for some $i \in \{1, \ldots, q\}$ and $\vec{y}^\star \notin V_i$.*

$\mathcal{A}$'s advantage is defined as his probability of victory taken over all coin tosses.

As in [7], a homomorphic NCS scheme Σ' is said to be secure if the network coding signature constructed via the conversion presented above is secure.

2.3 Complexity Assumptions

We consider groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1 p_2 p_3$ for which a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is computable. In the following, for each $i \in \{1, 2, 3\}$, we denote by $\mathbb{G}_{p_i}$ the subgroup of order p_i. Also, for all distinct $i, j \in \{1, 2, 3\}$, we call $\mathbb{G}_{p_i p_j}$ the subgroup of order $p_i p_j$.

An important property of composite order groups is that pairing two elements of order p_i and p_j, with $i \neq j$, always gives the identity element $1_{\mathbb{G}_T}$.

In this setting, we rely on the following assumptions introduced in [17].

Assumption 1 Given $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}, X_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, and T, it is infeasible to efficiently decide if $T \in_R \mathbb{G}_{p_1 p_2}$ or $T \in_R \mathbb{G}_{p_1}$.

Assumption 2 Let $g, X_1 \overset{R}{\leftarrow} \mathbb{G}_{p_1}, X_2, Y_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}, Y_3, Z_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$. Given a tuple $(g, X_1 X_2, Z_3, Y_2 Y_3)$ and T, it is hard to decide if $T \in_R \mathbb{G}$ or $T \in_R \mathbb{G}_{p_1 p_3}$.

Assumption 3 Let $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}$, $X_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ and $\alpha, s \overset{R}{\leftarrow} \mathbb{Z}_N$. Given $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$, it is infeasible to compute $e(g, g)^{\alpha s}$.

We note that, while Lewko and Waters rely on the decisional variant of Assumption 3 (according to which $e(g, g)^{\alpha s}$ is indistinguishable from a random element of $\mathbb{G}_T$), its computational counterpart suffices here.

3 Homomorphic NCS Scheme in the Standard Model

Intuitively, the construction is based on an observation that network coding signatures can be seen as an implication of the spatial encryption primitive introduced by Boneh and Hamburg [8] in the same way as identity-based encryption implies digital signatures (according to an observation by Naor reported in [6]). In spatial encryption, private keys are associated with affine subspaces while ciphertexts correspond to vectors. Decryption is possible when the ciphertext's vector lies in the subspace of the key. By applying Naor's transformation to the spatial encryption scheme of [8], one readily obtains a sort of selectively secure network coding signature, modulo some twist to bind the file identifier to the subspace which is being signed. By itself, this transformation does not provide the homomorphic property that we are after. To obtain it, we need to start from a specific variant of the NCS scheme derived from the spatial encryption system of [8] and carefully re-use the same randomness to separately sign vectors of the same subspace. Full security (as opposed to selective security [5]) is obtained using the Lewko-Waters techniques to build (hierarchical) identity-based encryption schemes [17].

More precisely, the public key comprises the description of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $N = p_1 p_2 p_3$, a number of $\mathbb{G}_{p_1}$ elements $(g, u, \{h_i\}_{i=0}^{n})$ as well as $e(g,g)^{\alpha}$ for some $\alpha \xleftarrow{R} \mathbb{Z}_N$. The first two components of each signature form a selectively-secure Boneh-Boyen signature [5] $(\sigma_1, \sigma_2) = (g^{\alpha} \cdot (u \cdot h_0^{\mathsf{id}})^r, g^r)$ on the file identifier id. As implicitly showed in [17], this signature can be proved fully secure if g, u and h_0 live in the subgroup of order p_1 and if σ_1, σ_2 are multiplied by a random element of $\mathbb{G}_{p_3}$. This signature (σ_1, σ_2) is then augmented with an element $\sigma_3 = (\prod_{j=1}^{n} h_j^{v_j})^r$, where $(v_1, \ldots, v_n) \in \mathbb{Z}_N^n$ is the vector to be signed. If all the vectors of $\mathrm{span}(\vec{v}_1, \ldots, \vec{v}_m)$ were signed altogether (by introducing one σ_3 per base vector), signatures would have nearly the same shape as private keys in the spatial encryption scheme of [8]: the only difference is the introduction of a file identifier in σ_1. Fortunately, base vectors can be signed separately as long as they are signed using the same exponent r. In this case, anyone can publicly compute a signature on any linear combination of $\vec{v}_1, \ldots, \vec{v}_m$.

To save the signer from maintaining a state and remember which random exponents were used to sign the vectors of all subspaces, $r \in \mathbb{Z}_N$ can be derived by applying a pseudorandom function to the file identifier id so as to be re-computable later on. We emphasize that the use of a PRF is not meant to de-randomize the scheme in an attempt to obtain unique signatures. The goal is simply to render the signer stateless.

To achieve security in the sense of definition 3, we need to keep signatures from being publicly re-randomizable in their $\mathbb{G}_{p_1}$ components. A simple solution is to compute (σ_1, σ_2) as a signature on a hash value of both id and $e(g,g)^r$, which prevents from altering the underlying r without invalidating the signature. Although this simple trick would not work with Waters signatures [23] (because their security proof would cease to go through), it is compatible with the dual encryption technique [24, 17] which is used to prove security. In addition, anyone can detect if vectors of the same file are signed using different values of r and only the signer can be blamed in such a situation.

3.1 Construction

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 order $N = p_1 p_2 p_3$, where $p_i > 2^\lambda$ for each $i \in \{1, 2, 3\}$. Choose $\alpha \overset{R}{\leftarrow} \mathbb{Z}_N$, $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $X_{p_3} \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, $b, a_i \overset{R}{\leftarrow} \mathbb{Z}_N$ for $i = 0$ to n. Then, select a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_N$, an identifier space $\mathcal{I}$ and pick a random seed $\kappa \overset{R}{\leftarrow} \{0,1\}^\tau$ for a pseudorandom function $\Psi : \{0,1\}^\tau \times \mathcal{I} \to \mathbb{Z}_N$, where $\tau \in \mathsf{poly}(\lambda)$. The private key is $\mathsf{sk} := \big(g^\alpha, \kappa\big)$ while the public key is

$$\mathsf{pk} := \Big((\mathbb{G}, \mathbb{G}_T),\ g,\ e(g,g)^\alpha,\ u = g^b,\ \{h_i = g^{a_i}\}_{i=0,\ldots,n},\ X_{p_3},\ H\Big).$$

Sign$(\mathsf{sk}, \mathsf{id}, \vec{v})$**:** on input of a vector $\vec{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_N^n$, a file identifier $\mathsf{id} \in \mathcal{I}$ and the private key $\mathsf{sk} = \big(g^\alpha, \kappa\big)$, conduct the following steps. First, compute a pseudorandom scalar $r = \Psi(\kappa, \mathsf{id}) \in \mathbb{Z}_N$. Then, compute $\mathsf{id}' = H(\mathsf{id}, e(g,g)^r) \in \mathbb{Z}_N$, choose $R_3, R_3', R_3'' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ and compute a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ as

$$\sigma_1 = g^\alpha \cdot (u \cdot h_0^{\mathsf{id}'})^r \cdot R_3, \qquad \sigma_2 = g^r \cdot R_3', \qquad \sigma_3 = \big(h_1^{v_1} \cdots h_n^{v_n}\big)^r \cdot R_3''$$

CompatibilityCheck$(\mathsf{pk}, \mathsf{id}, \{\sigma_i\}_{i=1}^{\ell})$**:** parses σ_i as $(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}) \in \mathbb{G}^3$ for $i = 1$ to ℓ. The algorithm will return 1 if and only if all $\sigma_{i,2}$ have the same $\mathbb{G}_{p_1}$ component: for $i = 2$ to ℓ, it checks if $e(\sigma_{1,2}/\sigma_{i,2}, g) = 1_{\mathbb{G}_T}$ and returns 0 otherwise. If all checks succeed, it returns 1.

Combine$(\mathsf{pk}, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$**:** given pk, a file identifier id and ℓ tuples (β_i, σ_i), parse σ_i as $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})$ for $i = 1$ to ℓ. Set $\sigma_1 = \sigma_{1,1} \cdot R_3$, $\sigma_2 = \sigma_{1,2} \cdot R_3'$ for randomly chosen $R_3, R_3' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$. Then, compute $\sigma_3 = \prod_{i=1}^{\ell} \sigma_{i,3}^{\beta_i} \cdot R_3''$, with $R_3'' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, and output $(\sigma_1, \sigma_2, \sigma_3)$.

Verify$(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma)$**:** given a public key $\mathsf{pk} = \big(g, e(g,g)^\alpha, u, \{h_i\}_{i=0}^{n}, X_{p_3}\big)$, a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ and a vector $\vec{y} = (y_1, \ldots, y_n) \in (\mathbb{Z}_N)^n$, compute $\mathsf{id}' = H(\mathsf{id}, e(\sigma_2, g))$ and return 1 if and only if

$$e(\sigma_1, g) = e(g,g)^\alpha \cdot e(u \cdot h_0^{\mathsf{id}'}, \sigma_2) \quad \text{and} \quad e(\sigma_3, g) = e(\sigma_2, h_1^{y_1} \cdots h_n^{y_n}). \quad (2)$$

Verifying the correctness of the scheme is straightforward since pairing an element of $\mathbb{G}_{p_1}$ with an element of $\mathbb{G}_{p_3}$ always gives the identity element in $\mathbb{G}_T$.

Efficiency. Signatures only consist of 3 group elements. Without optimizations, verifying individual signatures entails to compute four pairings. However, when multiple signatures must be checked before being combined, a constant number of pairing evaluations suffices when randomized batch verification techniques are used.

Indeed, when network nodes process ℓ signatures $\{(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})\}_{i=1}^{\ell}$ from the same file identified by $\mathsf{id} \in \mathcal{I}$, they can first check that all $\{\sigma_{i,2}\}_{i=1}^{\ell}$ have the same $\mathbb{G}_{p_1}$ component by testing if $e\big(\prod_{i=2}^{\ell}(\sigma_{1,2}/\sigma_{i,2})^{\omega_i}, g\big) = 1_{\mathbb{G}_T}$ for randomly chosen $\omega_2, \ldots, \omega_\ell \overset{R}{\leftarrow} \mathbb{Z}_N$. If this test is satisfied, $\sigma_{1,2}, \ldots, \sigma_{\ell,2}$ all correspond to the same r, with overwhelming probability, and the same $\sigma_{1,2}$ can be used to verify equations (2) for $i = 1$ to ℓ. Namely, if $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})$ pertains to $\vec{v}_i = (v_{i,1}, \ldots, v_{i,n})$, signatures are all valid if $e(\sigma_{i,1}, g) = e(g,g)^{\alpha} \cdot e(u \cdot h_0^{\mathsf{id}'}, \sigma_{1,2})$ and $e(\sigma_{i,3}, g) = e(\sigma_{1,2}, \prod_{k=1}^{n} h_k^{v_{i,k}})$ for $i = 1$ to ℓ. Then, if the network node picks randomizers $\delta_i, \delta'_i \xleftarrow{R} \mathbb{Z}_N$, for $i = 1$ to ℓ, all signatures can be batch-verified by testing if

$$e\Big(g, \prod_{i=1}^{\ell} \sigma_{i,1}^{\delta_i} \cdot \prod_{j=1}^{\ell} \sigma_{j,3}^{\delta'_j}\Big) = e(g,g)^{\alpha \cdot \sum_{i=1}^{\ell} \delta_i} \cdot e\Big(\sigma_{1,2}, (u \cdot h_0^{\mathsf{id}'})^{\sum_{i=1}^{\ell} \delta_i} \cdot \prod_{k=1}^{n} h_k^{\sum_{j=1}^{\ell} \delta'_j v_{jk}}\Big).$$

When verification fails, recent techniques [21] can be adapted to determine which signatures are bad and which packets should be filtered.

As in earlier standard model constructions [7, 3], the public key size is linear in the dimension n of vectors. We leave it as an interesting open problem to avoid this dependency without resorting to random oracles.

Converted Scheme. From the homomorphic network coding signature, one can obtain an ordinary network coding signature via the generic conversion given by Boneh *et al.* [7] (and recalled in section 2.2). Applying this conversion to our scheme results in the signature of the form $\{(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})\}_{i=1}^{m}$. This scheme is redundant and we can reuse the first two elements for all i. Indeed to sign a subspace V where $\vec{v}_1, \ldots, \vec{v}_m$ is the properly augmented basis, the signing algorithm outputs $\sigma = (\sigma_1, \sigma_2, \{\sigma_{3,i}\}_{i=1}^{m})$ where

$$\sigma_1 = g^{\alpha} \cdot (g^{b + a_0 \mathsf{id}'})^r \cdot R_3, \qquad \sigma_2 = g^r \cdot R'_3, \qquad \sigma_{3,i} = (g^{\langle \vec{a}, \vec{v}_i \rangle})^r \cdot R''_{3,i},$$

where $R_3, R'_3, R''_{3,i} \in_R \mathbb{G}_{p_3}$ and we denote $\vec{a} = (a_1, \ldots, a_n)$. In the next section, we will prove the security of this scheme instead of the scheme converted with the generic conversion.

3.2 Security Proof

We first give a simple lemma describing the general form of signatures that are accepted by the verification of the proposed NCS scheme (with redundancy cut as mentioned above).

Lemma 1. *For any identifier-vector-signature tuple* $(\mathsf{id}, \vec{y}, \sigma = (\sigma_1, \sigma_2, \{\sigma_{3,i}\}_{i=1}^{m}))$, *if it holds that* $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma) = 1$, *then we have*

$$\sigma_1 = g^{\alpha} \cdot (g^{b + a_0 \mathsf{id}'})^r \cdot Z_1, \qquad \sigma_2 = g^r \cdot Z_2, \qquad \sigma_{3,i} = (g^{\langle \vec{a}, \vec{v}_i \rangle})^r \cdot Z_{3,i}, \tag{3}$$

where $\mathsf{id}' = H(\mathsf{id}, e(\sigma_2, g))$, *for some* $r \in \mathbb{Z}_N$, $Z_1, Z_2, Z_{3,i} \in \mathbb{G}_{p_2 p_3}$ *and some vectors* $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{Z}_N^n$ *such that*

$$\vec{a}(U(\vec{y}^R)^{\top} - \vec{y}^{\top}) = 0, \ \textit{where } U = \begin{pmatrix} | & & | \\ \vec{v}_1^{\top} & \cdots & \vec{v}_m^{\top} \\ | & & | \end{pmatrix} \tag{4}$$

where we write $\vec{y} = \vec{y}^L || \vec{y}^R$ *with* $\vec{y}^L \in \mathbb{Z}_N^{n-m}, \vec{y}^R \in \mathbb{Z}_N^m$.

Proof. Let an id-vector-signature tuple $(\mathsf{id}, \vec{y}, \sigma = (\sigma_1, \sigma_2, \{\sigma_{3,i}\}_{i=1}^m))$ be a valid tuple, that is $\mathsf{Verify}(\mathsf{pk}, \mathsf{id}, \vec{y}, \sigma) = 1$. We will prove that σ will have the form of equation (3). First, since the tuple is accepted, we have

$$e(\sigma_1, g) = e(g, g)^\alpha \cdot e(g^b \cdot (g^{a_0})^{\mathsf{id}'}, \sigma_2) \tag{5}$$

$$e(\prod_{i=1}^{m} \sigma_{3,i}^{y_{n-m+i}}, g) = e(\sigma_2, g^{\langle \vec{a}, \vec{y} \rangle}), \tag{6}$$

where $\mathsf{id}' = H(\mathsf{id}, e(\sigma_2, g))$. Since $\sigma_2 \in \mathbb{G}$, we can write $\sigma_2 = g^r Z_2$ for some $r \in \mathbb{Z}_N$ and $Z_2 \in \mathbb{G}_{p_2p_3}$. Equation (5) then implies $\sigma_1 = g^\alpha \cdot (g^{b+a_0\mathsf{id}'})^r \cdot Z_1$ for some $Z_1 \in \mathbb{G}_{p_2p_3}$, as claimed. Similarly, we have $\sigma_{3,i} = (g^{\beta_i})^r \cdot Z_{3,i}$ for some $\beta_i \in \mathbb{Z}_N$. It remains to prove the property of β_i. Equation (6) implies that $\sum_{i=1} \beta_i y_{n-m+i} = \langle \vec{a}, \vec{y} \rangle$. If we write $\beta_i = \langle \vec{a}, \vec{v}_i \rangle$ for some $\vec{v}_i \in \mathbb{Z}_N^n$, then the equation (4) is obtained. This concludes the proof. □

Theorem 1. *The scheme is a secure homomorphic network coding signature if Ψ is a secure pseudorandom function, if H is a collision-resistant hash function and if Assumption 1, Assumption 2 and Assumption 3 all hold.*

Proof. The proof follows the dual system methodology used in [24, 17]. From Lemma 1, any valid identifier-vector-signature triple $(\mathsf{id}, \vec{y}, \sigma)$ will have the following generic form:

$$\sigma_1 = g^\alpha \cdot (g^{b+a_0\mathsf{id}'})^r \cdot g_2^{w_1} \cdot R_1, \qquad\qquad \sigma_2 = g^r \cdot g_2^{w_2} \cdot R_2, \tag{7}$$

$$\sigma_{3,i} = (g^{\langle \vec{a}, \vec{v}_i \rangle})^r \cdot g_2^{w_{3,i}} \cdot R_{3,i}, \tag{8}$$

where $\mathsf{id}' = H(\mathsf{id}, e(\sigma_2, g))$, for some $r \in \mathbb{Z}_N$, $w_1, w_2, w_{3,i} \in \mathbb{Z}_N$, some group elements $R_1, R_2, R_{3,i} \in \mathbb{G}_{p_3}$ and vectors $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{Z}_N^n$ which satisfied Eq. (4).

We will distinguish two types of signatures as follows.

- Type A: $(w_1, w_2, w_{3,1}, \ldots, w_{3,n}) = (0, 0, 0, \ldots, 0) \bmod p_2$.
- Type B: $(w_1, w_2, w_{3,1}, \ldots, w_{3,n}) \neq (0, 0, 0, \ldots, 0) \bmod p_2$.

We will call Type A forgery (resp. Type B forgery) a fake signature of Type A (resp. Type B) which is produced by the forger in the game of definition 3.

The proof considers a sequence of $q + 3$ games. It starts with the real attack game Game$_{real}$ followed by Game$_1$, Game$_2$, Game$_3$, Game$_{4.1}, \ldots,$ Game$_{4.q}$. In the following we let $V^{(j)}$ be the j-th query where $j \in \{1, \ldots, q\}$ and let $(\sigma_1^{(j)}, \sigma_2^{(j)}, \{\sigma_{3,i}^{(j)}\}_{i=1}^m)$ be the answer to the query.

Game$_1$: Replacing r with random. This game is identical to as Game$_{real}$ with the difference that the challenger generates all signatures using truly random exponents $r \overset{R}{\leftarrow} \mathbb{Z}_N$ (and care is taken to use the same r to sign all vectors of the same subspace) instead of pseudorandom values. Clearly, the security of the PRF implies that Game$_1$ is computationally indistinguishable from Game$_{real}$.

Game$_2$: ELIMINATING COLLISIONS. It is as Game$_1$ but the game will abort if there exists an index $j \in \{1, \ldots, q\}$ such that $\mathsf{id}'^{\star} = H(\mathsf{id}^{\star}, e(\sigma_2^{\star}, g)) = H(\mathsf{id}_j, e(\sigma_2^{(j)}, g)) = \mathsf{id}'_j$ but it holds that either $\mathsf{id}^{\star} \neq \mathsf{id}_j$ or $\mathsf{id}^{\star} = \mathsf{id}_j$ but $e(\sigma_2^{\star}, g) \neq e(\sigma_2^{(j)}, g)$. It is straightforward that, assuming the collision-resistance of H, the difference between Game$_1$ and Game$_2$ is negligible.

Game$_3$: RESTRICTION MODULO p_2. It is as Game$_2$ but the game will further abort if either of the following event occurs.

- Adversary $\mathcal{A}$ outputs a class-(i) forgery (*i.e.,* $\mathsf{id}^{\star} \neq \mathsf{id}_j$ for any j and $\vec{y}^{\star} \neq \vec{0}$) or a class-(ii) forgery (*i.e.,* for which $\mathsf{id}^{\star} = \mathsf{id}_j$ for some j and $e(\sigma_2^{\star}, g) \neq e(\sigma_{2,j}, g)$) but $\mathsf{id}'^{\star} = \mathsf{id}'_j \bmod p_2$ (even if $\mathsf{id}'^{\star} \neq \mathsf{id}'_j$) for some index $j \in \{1, \ldots, q\}$.
- Adversary $\mathcal{A}$ outputs a class-(iii) forgery (*i.e.,* $\mathsf{id}^{\star} = \mathsf{id}_j$ for some j and $\vec{y}^{\star} \notin V_j$) but for which $\vec{y}^{\star} \bmod p_2 \in V_j \bmod p_2$. Here, we denote by $V \bmod p_2$ the subspace V reduced in $\mathbb{Z}_{p_2}^n$. More precisely, for any subspace $V = \mathrm{span}(\vec{v}_1, \ldots, \vec{v}_m) \subset \mathbb{Z}_N^n$, the notation $V \bmod p_2$ denotes $\mathrm{span}(\vec{v}_1 \bmod p_2, \ldots, \vec{v}_m \bmod p_2) \subset \mathbb{Z}_{p_2}^n$.

Lemma 2 shows that, under Assumption 1 and Assumption 2, the difference between Game$_2$ and Game$_3$ is negligible. Then, Lemma 3 shows that, if $\mathcal{A}$ can output a Type B forgery in Game$_3$, Assumption 1 is false.

Game$_{4.0}$: SIMPLIFICATION. This is a reformulation of Game$_3$ for ease of reading. The game will accept only the following forgery. (Otherwise, it will abort).

- Adversary $\mathcal{A}$ outputs a forgery with $\mathsf{id}'^{\star} \neq \mathsf{id}'_j \bmod p_2$ for any j and $\vec{y}^{\star} \neq \vec{0}$.
- Adversary $\mathcal{A}$ outputs a forgery for which $\mathsf{id}^{\star} = \mathsf{id}_j$, for some $j \in \{1, \ldots, q\}$, and $\vec{y}^{\star} \bmod p_2 \notin V_j \bmod p_2$.

We note that in this game, as in Game$_{real}$, $\mathcal{A}$ is only given Type A signatures.

Game$_{4.k}$ $(1 \leq k \leq q)$: HYBRID TYPES. It is as Game$_0$ but the adversary obtains Type B signatures at the first k signing queries whereas the challenger answers the remaining $q-k$ signing queries by returning Type A signatures. Lemma 4 shows that, if the adversary has noticeably higher probability to output a Type A forgery in Game$_{4.(k+1)}$ than in Game$_{4.k}$, there must be a breach in Assumption 2.

Game$_{4.q}$: ALL TYPE B. The forger $\mathcal{A}$ only obtains Type B signatures and it becomes easy to prove that any Type A forgery allows breaking Assumption 3, as shown by Lemma 5.

Denote negl as a negligible function in λ. Let $\mathsf{W}_i, \mathsf{W}_i^{\mathsf{A}}, \mathsf{W}_i^{\mathsf{B}}$ be the probability that the adversary successfully outputs a forgery in game i of either type, type A, and type B respectively. We then have that $|\mathsf{W}_{real} - \mathsf{W}_3| \leq \mathsf{negl}$ guaranteed by the security of PRF, collision-resistance hash, and Lemma 2. Also $\mathsf{W}_3^{\mathsf{B}} \leq \mathsf{negl}$, $|\mathsf{W}_{4,0}^{\mathsf{A}} - \mathsf{W}_{4,q}^{\mathsf{A}}| \leq \mathsf{negl}$, and $\mathsf{W}_{4,q}^{\mathsf{A}} \leq \mathsf{negl}$ are implied by Lemma 3, 4, and 5, respectively. Combining the above, we obtain

$$\mathsf{W}_{real} \leq |\mathsf{W}_{real} - \mathsf{W}_3| + \mathsf{W}_3^{\mathsf{B}} + |\mathsf{W}_{4,0}^{\mathsf{A}} - \mathsf{W}_{4,q}^{\mathsf{A}}| + \mathsf{W}_{4,q}^{\mathsf{A}} \leq \mathsf{negl},$$

where we recall that $\mathsf{W}_3 = \mathsf{W}_{4,0}$ and see that $\mathsf{W}_3 = \mathsf{W}_3^{\mathsf{A}} + \mathsf{W}_3^{\mathsf{B}}$. This concludes the proof. □

Lemma 2. *Any significant difference between the adversary's behaviors in Game$_2$ and Game$_3$ contradicts either Assumption 1 or Assumption 2.*

Proof. The two games are identical unless the adversary $\mathcal{A}$ outputs a forgery involving a pair $(\mathsf{id}'^{\star}, \vec{y}^{\star})$ such that either: (1) $\mathsf{id}'^{\star} = \mathsf{id}'_j \bmod p_2$ whereas we have $\mathsf{id}'^{\star} \neq \mathsf{id}'{}_j \bmod N$ for some $j \in \{1, \ldots, q\}$; (2) there exists $j \in \{1, \ldots, q\}$ such that $\mathsf{id}'^{\star} = \mathsf{id}'_j \bmod N$ but $\det(M) = 0 \bmod p_2$ and $\det(M) \neq 0 \bmod N$, where $M \in \mathbb{Z}_N^{n \times n}$ is the matrix

$$M = \left(\begin{array}{cccc} & | & | & | \\ R_{n\times(n-m_j-1)} & \vec{v}_{j,1}^{\top} \cdots & \vec{v}_{j,m_j}^{\top} & \vec{y}^{\star\top} \\ & | & | & | \end{array} \right),$$

with $m_j = \dim(V_j) < n$, such that $R_{n\times(n-m_j-1)}$ is a $n \times (n - m_j - 1)$ matrix whose columns are orthogonal to $\mathrm{span}(\vec{v}_{j,1}, \ldots, \vec{v}_{j,m_j}, \vec{y}^{\star})$ (such a matrix can be obtained via the Gram-Schmidt process). The matrix has the desired properties since $\vec{y}^{\star} \bmod p_2 \in V \bmod p_2$ although $\vec{y}^{\star} \notin V$. The simulator $\mathcal{B}$ can extract a non-trivial factor of N by computing $\gcd(\mathsf{id}'^{\star} - \mathsf{id}'_j, N)$ in case (1) or $\gcd(\det(M), N)$ in case (2). As shown in [17][Lemma 5], this allows breaking either Assumption 1 or Assumption 2 depending on which factor is extracted. □

Lemma 3. *Under Assumption*[3] *1, no PPT adversary can output a Type B forgery in Game$_3$.*

Proof. We show that, if the adversary outputs a Type B forgery in Game$_3$, there is an algorithm $\mathcal{B}$ that, given (g, X_3, T), decides if $T \in_R \mathbb{G}_{p_1}$ or $T \in_R \mathbb{G}_{p_1 p_2}$.

The distinguisher $\mathcal{B}$ sets up the public key pk as $e(g,g)^{\alpha}$, $X_{p_3} = X_3$, $u = g^b$, $h_i = g^{a_i}$ for $i = 0$ to n. Denote $\vec{a} = (a_1, \ldots, a_n)$. It answers all private key queries according to the specification of the signing algorithm since it knows the private key.

At the end, $\mathcal{A}$ outputs a file identifier $\mathsf{id}^{\star}$, a Type-B signature $(\sigma_1^{\star}, \sigma_2^{\star}, \{\sigma_{3,i}^{\star}\}_{i=1}^{m})$ and a vector $\vec{y}^{\star}$. The algorithm $\mathcal{B}$ then computes

$$\eta_1 = \frac{\sigma_1^{\star}}{g^{\alpha} \cdot {\sigma_2^{\star}}^{b + a_0 \mathsf{id}'^{\star}}}, \qquad \eta_2 = \frac{\prod_{i=1}^{m} {\sigma_{3,i}^{\star}}^{y_{n-m+i}}}{{\sigma_2^{\star}}^{\langle \vec{a}, \vec{y}^{\star} \rangle}}.$$

The $\mathbb{G}_{p_1}$ components of these terms are necessarily canceled out due to equations (3)-(4). Recall that a Type-B signature is in the generic form (7) with $(w_1, w_2, w_{3,1}, \ldots, w_{3,n}) \neq (0, 0, 0, \ldots, 0) \bmod p_2$. For this reason, the $\mathbb{G}_{p_2}$ components in η_1, η_2 will be $g_2^{w_1 - w_2(b + a_0 \mathsf{id}'^{\star})}$ and $g_2^{\sum_{i=1}^{m} w_{3,i} y_{n-m+i} - w_2 \langle \vec{a}, \vec{y}^{\star} \rangle}$, respectively. Hence, as long as $b, a_0, \vec{a} \bmod p_2$ are information theoretically hidden to

[3] We note that the lemma holds under a weaker assumption which is the hardness of finding an element of order p_2 or $p_2 p_3$ given (g, X_3).

the adversary, there must be an element of $\mathbb{G}_{p_2p_3}$ with non-trivial $\mathbb{G}_{p_2}$ component among η_1, η_2. But this is true since $b, a_0, \vec{a} \bmod p_2$ is uncorrelated to $b, a_0, \vec{a} \bmod p_1$, which is the only information available from the public key. Therefore, our distinguisher $\mathcal{B}$ can conclude that $T \in \mathbb{G}_{p_1p_2}$ if and only if either $e(T, \eta_1) \neq 1_{\mathbb{G}_T}$ or $e(T, \eta_2) \neq 1_{\mathbb{G}_T}$. □

Lemma 4. *The adversary outputs a Type A forgery with negligibly different probabilities in Game$_{4.k}$ and Game$_{4.(k+1)}$ if Assumption 2 holds.*

Proof. Let us assume that a forger $\mathcal{A}$ has significantly better probability of outputting a Type A forgery in Game$_{4.(k+1)}$ than in Game$_{4.k}$. We outline a distinguisher $\mathcal{B}$ that breaks Assumption 2 with non-negligible advantage.

Algorithm $\mathcal{B}$ takes as input $(g, X_1X_2, Z_3, Y_2Y_3, T)$ and uses its interaction with $\mathcal{A}$ to decide if $T \in \mathbb{G}$ or $T \in \mathbb{G}_{p_1p_3}$. Recall that $\mathcal{A}$ must obtain Type B signatures at her first k signing queries and Type A signatures at the last $q - k - 1$ queries. We will simulate the interaction so that the k^{th}-query will be a Type A signature (hence Game$_{4.k}$) if $T \in \mathbb{G}_{p_1p_3}$ and a Type B signature (hence Game$_{4.(k+1)}$) if $T \in \mathbb{G}$. We then show that the distinguisher $\mathcal{B}$ can indeed distinguish whether $\mathcal{A}$'s forgery will be of Type A or Type B with overwhelming probability.

To this end, $\mathcal{B}$ prepares the public key pk by choosing $\alpha, b \overset{R}{\leftarrow} \mathbb{Z}_N$, $a_i \overset{R}{\leftarrow} \mathbb{Z}_N$, for $i = 0$ to n, and setting $u = g^b$, $h_i = g^{a_i}$ for $i = 0$ to n. The public key $\mathsf{pk} = \{g, e(g,g)^\alpha, u, h_0, h_1, \ldots, h_n, Z_3\}$ is given to $\mathcal{A}$. Then, $\mathcal{B}$ answers $\mathcal{A}$'s queries depending on the index $j \in \{1, \ldots, q\}$ of the query.

[**Case** $j < k$] To sign the j^{th} vector space $V^{(j)} = \mathrm{span}(\vec{v}_1^{(j)}, \ldots, \vec{v}_m^{(j)})$ chosen by $\mathcal{A}$, $\mathcal{B}$ first chooses a random file identifier $\mathsf{id}_j \overset{R}{\leftarrow} \mathcal{I}$ and a random exponent $r \overset{R}{\leftarrow} \mathbb{Z}_N$. It then chooses $w_1, w_2, \{w_{3,i}\}_{i=1}^m \overset{R}{\leftarrow} \mathbb{Z}_N$, $Z_3, Z_3', \{Z_{3,i}''\}_{i=1}^m \overset{R}{\leftarrow} \mathbb{G}_{p_3}$. Let $\mathsf{id}_j' = H(\mathsf{id}_j, e(g,g)^r)$. It finally computes a Type-B signature $(\sigma_1, \sigma_2, \{\sigma_{3,i}\}_{i=1}^m)$ on $V^{(j)}$ as

$$\sigma_1 = g^\alpha \cdot (u \cdot h_0^{\mathsf{id}_j'})^r \cdot (Y_2Y_3)^{w_1} \cdot Z_3, \qquad \sigma_2 = g^r \cdot (Y_2Y_3)^{w_2} \cdot Z_3',$$
$$\sigma_{3,i} = (g^{\langle \vec{a}, \vec{v}_i^{(j)} \rangle})^r \cdot (Y_2Y_3)^{w_{3,i}} \cdot Z_{3,i}''.$$

[**Case** $j > k$] In this case, $\mathcal{A}$ simply computes a Type A signature using the private key g^α as specified by the signing algorithm (except that, as in Game$_1$, r is chosen at random in $\mathbb{Z}_N$ rather than as a pseudorandom value).

[**Case** $j = k$] To answer the k^{th} private key query $V^{(j)} = \mathrm{span}(\vec{v}_1^{(j)}, \ldots, \vec{v}_m^{(j)})$, $\mathcal{B}$ first picks a random file identifier $\mathsf{id}_j \overset{R}{\leftarrow} \mathcal{I}$. It then chooses $w_1, w_2, \{w_{3,i}\}_{i=1}^m \overset{R}{\leftarrow} \mathbb{Z}_N$, $Z_3, Z_3', \{Z_{3,i}''\}_{i=1}^m \overset{R}{\leftarrow} \mathbb{G}_{p_3}$. It uses its input T to compute a hash value $\mathsf{id}_j' = H(\mathsf{id}_j, e(T, g))$. It finally computes the signature $(\sigma_1, \sigma_2, \{\sigma_{3,i}\}_{i=1}^m)$ on the subspace $V^{(j)}$ as

$$\sigma_1 = g^\alpha \cdot T^{b + a_0 \mathsf{id}_j'} \cdot Z_3, \qquad \sigma_2 = T \cdot Z_3', \qquad \sigma_{3,i} = T^{\langle \vec{a}, \vec{v}_i^{(j)} \rangle} \cdot Z_{3,i}''.$$

It is easy to observe that, in the situation where $T \in_R \mathbb{G}$, if we let g_2^x be the $\mathbb{G}_{p_2}$ component of T for some $x \in \mathbb{Z}_{p_2}^*$, we obtain a Type B signature where

$w_1 = x(b + a_0\mathsf{id}'_j) \bmod p_2$, $w_2 = x \bmod p_2$, and $w_{3,i} = x(\langle \vec{a}, \vec{v}^{(j)} \rangle) \bmod p_2$ for $i = 1$ to m. In contrast, if $T \in_R \mathbb{G}_{p_1 p_3}$, the above forms a Type A signature.

At the end of the game, $\mathcal{A}$ outputs a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \{\sigma_{3,i}^\star\}_{i=1}^m)$ and a vector $\vec{y}^\star$ and a file identifier $\mathsf{id}^\star$ such that the property stated in the Game_0 holds. That is either

- a forgery with $\mathsf{id}'^\star \neq \mathsf{id}'_j \bmod p_2$ for any j and $\vec{y}^\star \neq \vec{0}$.
- a forgery with $\mathsf{id}^\star = \mathsf{id}_j$ for some j and $\vec{y}^\star \bmod p_2 \notin V^{(j)} \bmod p_2$.

At this stage, $\mathcal{B}$ halts and checks whether the forgery is of Type A or B. If the forgery is of Type A, it returns 0 (meaning that $T \in_R \mathbb{G}_{p_1 p_3}$). If the forgery is believed to be of Type B, $\mathcal{B}$ rather bets on $T \in_R \mathbb{G}_{p_1 p_2 p_3}$ and outputs 1.

In order to decide which kind of forgery $\mathcal{A}$ comes up with, $\mathcal{B}$ uses the input value $X_1 X_2$ as follows. The algorithm $\mathcal{B}$ computes $\mathsf{id}'^\star = H(\mathsf{id}^\star, e(\sigma_2^\star, g))$ and

$$\eta_1 = \frac{\sigma_1^\star}{g^\alpha \cdot {\sigma_2^\star}^{b + a_0 \mathsf{id}'^\star}}, \qquad \eta_2 = \frac{\prod_{i=1}^m {\sigma_{3,i}^\star}^{y_{n-m+i}}}{{\sigma_2^\star}^{\langle \vec{a}, \vec{y}^\star \rangle}}.$$

The $\mathbb{G}_{p_1}$ component of each term is canceled out due to equations (3)-(4). If $e(X_1X_2, \eta_1) = 1$ and $e(X_1X_2, \eta_2) = 1$, then the algorithm $\mathcal{B}$ deduces that $\sigma^\star$ is of Type A. Otherwise, it is seen as a Type B signature. To see why this test works with overwhelming probability, we note that, since $\sigma^\star$ properly verifies, it must be of the form of equations (7)-(8) with $(w_1^\star, w_2^\star, w_{3,1}^\star, \ldots, w_{3,m}^\star)$ so that we have

$$e(X_1X_2, \eta_1) = e(X_2, g_2)^{w_1^\star - w_2^\star(b + a_0 \mathsf{id}'^\star)},$$
$$e(X_1X_2, \eta_2) = e(X_2, g_2)^{\sum_{i=1}^m w_{3,i}^\star y_{n-m+i} - w_2^\star(\langle \vec{a}, \vec{y}^\star \rangle)}.$$

If $\sigma^\star$ is of Type B, it can only be interpreted as a Type A signature if and only if

$$w_1^\star - w_2^\star(b + a_0 \mathsf{id}'^\star) = 0 \bmod p_2, \text{ and} \tag{9}$$
$$\sum_{i=1}^m w_{3,i}^\star y_{n-m+i} - w_2^\star(\langle \vec{a}, \vec{y}^\star \rangle) = 0 \bmod p_2. \tag{10}$$

We show that this occurs with negligible probability as follows.

- If the forgery is of the first class, that is $\mathsf{id}^\star \neq \mathsf{id}_j$ for any $j \in \{1, \ldots, q\}$, then $b + a_0 \mathsf{id}'^\star \bmod p_2$ is independent of $\mathcal{A}$'s view which consists only of $b + a_0 \mathsf{id}'_k \bmod p_2$. Therefore equation (9) occurs with negligible probability.
- If the forgery is of the second class, that is $\vec{y}^\star \bmod p_2 \notin V^{(j)} \bmod p_2$ for any j, then $\langle \vec{a}, \vec{y}^\star \rangle \bmod p_2$ is independent of $\mathcal{A}$'s view and (10) can only be verified with negligible chance. Indeed, let us consider what $\mathcal{A}$ knows in the information theoretic sense about the values $(a_1, \ldots, a_n)$ taken modulo p_2.

It amounts to the right-hand side of the following system of linear equations:

$$\begin{pmatrix} —\vec{v}_1^{(k)}— \\ \vdots \\ —\vec{v}_m^{(k)}— \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \bmod p_2,$$

where we let $g_2^{t_i}$ be the $\mathbb{G}_{p_2}$ component of $\sigma_{3,i}^{(k)}$. Since $\vec{y}^\star \bmod p_2 \notin V^{(k)} \bmod p_2$, $\vec{y}^\star$ is not in the row space of the above matrix. Since $m < n$, $\langle \vec{a}, \vec{y}^\star \rangle \bmod p_2$ is independent of $\mathcal{A}$'s view. □

Lemma 5. *Any PPT algorithm $\mathcal{A}$ outputting a Type A forgery in Game$_{4.q}$ allows breaking Assumption 3.*

Proof. We outline an algorithm $\mathcal{B}$ that takes as input $\big(g, g^\alpha X_2, X_3, g^s Y_2, Z_2\big)$ and aims at computing $T = e(g,g)^{\alpha s}$ using its interaction with $\mathcal{A}$. To this end, $\mathcal{B}$ generates the public key $\mathsf{pk} = \big(g, e(g,g)^\alpha, u, \{h_i\}_{i=0,\ldots,n}, X_{p_3}\big)$ by choosing $b, a_0, \ldots, a_n \xleftarrow{R} \mathbb{Z}_N$ and setting $X_{p_3} = X_3$, $e(g,g)^\alpha = e(g^\alpha X_2, g)$ as well as $u = g^b$ and $h_i = g^{a_i}$ for $i = 0$ to n.

When the forger $\mathcal{A}$ makes a private key query $V^{(j)} = \mathrm{span}(\vec{v}_1^{(j)}, \ldots, \vec{v}_n^{(j)})$, $\mathcal{B}$ chooses $\mathsf{id} \xleftarrow{R} \mathcal{I}$, $r \xleftarrow{R} \mathbb{Z}_N$, $w_1, w_2 \xleftarrow{R} \mathbb{Z}_N$, $R_3, R_3', R_3'' \xleftarrow{R} \mathbb{G}_{p_3}$, $w_{3,i} \xleftarrow{R} \mathbb{Z}_N$, $R_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, for $i = 1$ to n. It defines $\mathsf{id}' = H(\mathsf{id}, e(g,g)^r) \in \mathbb{Z}_N$ and computes

$$\sigma_1 = (g^\alpha X_2) \cdot (u \cdot h_0^{\mathsf{id}'})^r \cdot Z_2^{w_1} \cdot R_3, \qquad \sigma_2 = g^r \cdot Z_2^{w_2} \cdot R_3',$$
$$\sigma_{3,i} = (g^{\langle a, \vec{v}_i^{(j)} \rangle})^r \cdot Z_2^{w_{3,i}} \cdot R_{3,i}$$

which has the distribution of a Type B signature.

At the end of the game, $\mathcal{A}$ outputs a valid tuple of a file identifier $\mathsf{id}^\star$, a signature $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \{\sigma_{3,i}^\star\}_{i=1}^m)$ of Type-A and a vector $\vec{y}^\star$. Algorithm $\mathcal{B}$ then computes

$$T = e\big(g^s Y_2, \frac{\sigma_1^\star}{\sigma_2^{\star\, b + a_0 \mathsf{id}'^\star}}\big) = e\big(g^s Y_2, \frac{g^\alpha \cdot (g^{b + a_0 \mathsf{id}'^\star})^r \cdot Z_1}{(g^r Z_2)^{b + a_0 \mathsf{id}'^\star}}\big) = e(g,g)^{\alpha s}. \qquad (11)$$

where the second equality is due to lemma 1 ($Z_1, Z_2 \in \mathbb{G}_{p_2 p_3}$). Since $\sigma^\star$ is a Type A signature, $\sigma_1^\star, \sigma_2^\star$ have no $\mathbb{G}_{p_2}$ component. Hence, the component Y_2 is canceled out in the pairing computation, which explains why $T = e(g,g)^{s\alpha}$ is obtained in the rightmost equality of (11). To conclude, in Game$_{4.q}$, $\mathcal{A}$'s advantage is thus negligible if Assumption 3 holds. □

References

1. R. Ahlswede, N. Cai, S. Li, R. Yeung. Network Information Flow. In *IEEE Trans. on Information Theory* 46, pp. 1204–1216, 2000.
2. S. Agrawal, D. Boneh. Homomorphic MACs: MAC-Based Integrity for Network Coding. In *ACNS'09*, *LNCS* 5536, pp. 292–305, 2009.

3. S. Agrawal, D. Boneh, X. Boyen, D. Freeman. Preventing Pollution Attacks in Multi-source Network Coding. In *PKC'10*, *LNCS* 6056, pp. 161–176, 2010.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pages 62–73, 1993.
5. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Eurocrypt'04*, *LNCS* 3027, pp. 223–238, 2004.
6. D. Boneh, M. Franklin. Identity-Based Encryption from the Weil Pairing. In *SIAM* Journal of Computing 32(3), pp. 586–615, 2003, earlier version in *Crypto'01*, *LNCS* 2139, pp. 213–229, 2001.
7. 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.
8. D. Boneh, M. Hamburg. Generalized Identity Based and Broadcast Encryption Schemes. In *Asiacrypt'08*, *LNCS* 5350, pp. 455–470, 2008.
9. D. Charles, K. Jain, K. Lauter. Signatures for Network Coding. In *40th Annual Conference on Information Sciences and Systems (CISS'06)*.
10. C. Fragouli, E. Soljanin. Network Coding Fundamentals. *Now Publishers Inc.*, Hanover, MA, USA, 2007.
11. D. Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Eurocrypt'10*, *LNCS* 6110, pp. 44–61, 2010.
12. R. Gennaro, J. Katz, H. Krawczyk, T. Rabin. Secure Network Coding over the Integers. In *PKC'10*, *LNCS* 6056, pp. 142–160, 2010.
13. C. Gkantsidis, P. Rodriguez. Network Coding for Large Scale Content Distribution. In *IEEE INFOCOM*, 2005.
14. T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, D. Karger. Byzantine Modification Detection in Multicast Networks using Randomized Network Coding. In *International Symposium on Information Theory (ISIT)* pp. 144–152, 2004.
15. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, M. Effros. Resilient Network Coding in the Presence of Byzantine Adversaries. In *IEEE Trans. on Information Theory* vol. 54, pp. 2596–2603, 2008.
16. M. Krohn, M. Freedman, D. Mazieres. On-the-fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In *IEEE Symposium on Security and Privacy*, pp. 226–240, 2004.
17. A. Lewko, B. Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010*, LNCS 5978, pp. 455-479, Springer, 2010.
18. S.-Y.-R. Li, R.-W. Yeung, N. Cai. Linear Network Coding. In *IEEE Trans. on Information Theory* vol. 49, pp. 371–381, 2003.
19. Y. Li, H. Yao, M. Chen, S. Jaggi, A. Rosen. RIPPLE Authentication for Network Coding. In *IEEE INFOCOM 2010*, 2010.
20. R. Johnson, D. Molnar, D. Song, D. Wagner. Homomorphic Signature Schemes. In *CT-RSA'02*, LNCS 2271, pp. 244–262, 2002.
21. B. Matt. Identification of Multiple Invalid Signatures in Pairing-Based Batched Signatures. In *PKC'09*, *LNCS* 5443, pp. 337–356, 2009.
22. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Crypto'84*, *LNCS* 196, pp. 47–53, 1984.
23. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, pp. 114–127, 2005.
24. B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Crypto'09*, *LNCS* series, 2009.
25. F. Zhao, T. Kalker, M. Médard, K. Han. Signatures for Content Distribution with Network Coding. In *International Symposium on Information Theory (ISIT)*, 2007.