Signing a Linear Subspace: Signature Schemes for Network Coding

Dan Boneh[1*] David Freeman[2†] Jonathan Katz[3‡]
Brent Waters[4§]

[1]Stanford University, dabo@cs.stanford.edu
[2]CWI and Universiteit Leiden, freeman@cwi.nl
[3]University of Maryland, jkatz@cs.umd.edu
[4]University of Texas at Austin, bwaters@cs.utexas.edu

Abstract

Network coding offers increased throughput and improved robustness to random faults in completely decentralized networks. In contrast to traditional routing schemes, however, network coding requires intermediate nodes to modify data packets *en route*; for this reason, standard signature schemes are inapplicable and it is a challenge to provide resilience to tampering by malicious nodes.

We propose two signature schemes that can be used in conjunction with network coding to prevent malicious modification of data. Our schemes can be viewed as signing linear subspaces in the sense that a signature σ on a subspace V authenticates exactly those vectors in V. Our first scheme is (suitably) *homomorphic* and has *constant* public-key size and per-packet overhead. Our second scheme does not rely on random oracles and is based on weaker assumptions.

We also prove a lower bound on the length of signatures for linear subspaces showing that our schemes are essentially optimal in this regard.

1 Introduction

Network coding [1, 23] refers to a general class of routing mechanisms where, in contrast to traditional "store-and-forward" routing, intermediate nodes *modify* data packets in transit. Network coding has been shown to offer a number of advantages with respect to traditional routing, the most well-known of which is the possibility of increased throughput in certain network topologies. It has also been suggested as a means of improving robustness against random network failures since, as

*Supported by DARPA IAMANET, NSF, and the Packard Foundation.

†Research was conducted at Stanford University and supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

‡Supported by NSF CNS-0447075, NSF CNS-0627306, the U.S. DoD/ARO MURI program, and the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

§Supported by NSF CNS-0749931, CNS-0524252, CNS-0716199, the U.S. Army Research Office under the CyberTA Grant No. W911NF-06-1-0316, and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. Portions of this research were conducted while the author was at SRI International.

with erasure codes [6], the destination can recover the original data (with high probability) once it has received sufficiently many correct packets, even if a large fraction of packets are lost.

Because of these advantages, network coding has been proposed for applications in wireless and/or ad-hoc networks, where communication is at a premium and centralized control may be unavailable; it has also been suggested as an efficient means for content distribution in peer-to-peer networks [22], and for improving the performance of large-scale data dissemination over the Internet [11].

A major concern in systems that use network coding is protecting against malicious modification of packets (i.e., "pollution attacks") by Byzantine nodes; see [13, 21] for two recent surveys and Section 2.2 for a discussion of previous work. The problem is particularly acute because errors introduced into even a single packet can propagate and pollute multiple packets making their way to the destination. This propagation is a consequence of the processing that honest nodes, downstream of any corrupted packets, apply to all incoming packets.

We propose two signature schemes that can be used to provide cryptographic protection against pollution attacks even when the adversary can corrupt an arbitrary number of nodes, eavesdrop on all network traffic, and insert or modify an arbitrary number of packets. Of course, the destination cannot possibly recover the file unless it receives a minimum number of uncorrupted packets; once this is the case, however, our schemes ensure that the destination can filter out any corrupted packets and recover the correct file. As our signatures are publicly verifiable, intermediate nodes could discard corrupted packets as well (though whether this is actually done will depend on the computational resources of the intermediate nodes). Our first scheme is particularly efficient, with both public-key size and per-packet overhead being constant. A detailed discussion of our schemes, and their advantages relative to prior work, is given in Section 2.3.

Our schemes can be viewed as signing linear subspaces in the sense that a signature σ on the subspace V authenticates exactly those vectors in V. We prove a lower bound on the signature length for any scheme for signing linear subspaces (under some mild restrictions), showing that our constructions are essentially optimal in this regard.

Outline of the paper. We provide a quick overview of network coding in Section 2.1. In Section 2.2 we discuss prior work addressing adversarial behavior in the context of network coding, and we describe the advantages of our schemes in Section 2.3. In Section 3 we introduce appropriate definitions of security for our setting and give relevant mathematical background. Sections 4 and 5 describe our constructions, and in Section 6 we prove our lower bound.

2 Background

2.1 Linear Network Coding

In a *linear* network coding scheme [23] (the only type with which we will be concerned), a file to be transmitted is viewed as an ordered sequence of n-dimensional vectors $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m \in \mathbb{F}_p^n$, where p is prime. We will sometimes refer to individual vectors as *blocks* or *packeets*. Before transmission, the source node creates the m *augmented vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_m$ given by:

$$\mathbf{v}_i = (\text{—}\bar{\mathbf{v}}_i\text{—}, \underbrace{0, \ldots, 0, 1}_{i}, \overbrace{0, \ldots, 0}^{m}) \in \mathbb{F}_p^{n+m};$$

that is, each original vector $\bar{\mathbf{v}}_i$ is appended with the vector of length m containing a single '1' in the ith position. These augmented vectors are then sent by the source as packets in the network. Since this step introduces $\Theta(m^2)$ communication overhead per file, one typically chooses $m \ll n$.

Each node in the network processes packets as follows. Upon receiving packets (i.e., vectors) $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \mathbb{F}_p^{n+m}$ on its ℓ incoming communication edges, node i computes the packet (vector) $\mathbf{w} = \sum_{j=1}^{\ell} \alpha_{i,j} \mathbf{w}_j$, where $\alpha_{i,j} \in \mathbb{F}_p$. The resulting vector $\mathbf{w}$ is then transmitted on the node's outgoing edges. That is, each node transmits a *linear combination* of the packets it receives. Thus, in a fault-free execution of the scheme, all packets transmitted on any link in the network are linear combinations of the original (augmented) file vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$.

The weights $\alpha_{i,j}$ used by the ith node in the network can be established by a central authority. More usefully (and more interestingly), however, these values can also be chosen randomly and independently by each node in a completely decentralized fashion. (In this case the scheme is sometimes referred to as "random network coding".) Although carefully designed codes can potentially have better performance, it has been shown that random network coding does almost as well with high probability [8, 14, 16].

There may be multiple destination nodes (i.e., receivers) who wish to obtain the original file from the source. When any such node receives m linearly independent vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$, it can recover the original file as follows: For a received vector $\mathbf{w}_i$, let $\mathbf{w}_i^L$ denote the left-most n positions of the vector, and let $\mathbf{w}_i^R$ denote the right-most m positions. The receiver first computes an $m \times m$ matrix G such that

$$G = \begin{pmatrix} \text{—}\mathbf{w}_1^R\text{—} \\ \vdots \\ \text{—}\mathbf{w}_m^R\text{—} \end{pmatrix}^{-1}. \tag{1}$$

(The matrix on the right-hand side is invertible as long as all the received vectors are correct.) The original file $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ is then given by

$$\begin{pmatrix} \text{—}\bar{\mathbf{v}}_1\text{—} \\ \vdots \\ \text{—}\bar{\mathbf{v}}_m\text{—} \end{pmatrix} = G \cdot \begin{pmatrix} \text{—}\mathbf{w}_1^L\text{—} \\ \vdots \\ \text{—}\mathbf{w}_m^L\text{—} \end{pmatrix}.$$

We stress that the receiver need not be aware of the weights $\{\alpha_{i,j}\}$ used by any intermediate node in the network in order to recover the file. On the other hand, if the weights used by the intermediate nodes *are* all known to the receiver (and the receiver is aware of the network topology) then the matrix G can be computed in advance and, in fact, the scheme can be run on the original file vectors $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ rather than on the augmented vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$. In our work, however, we will assume that augmented vectors are used.

2.2 Dealing with Adversarial Behavior

Network coding can offer resilience to random packet loss since the receiver can reconstruct the original file from *any* set of m correctly formed, linearly independent vectors. (Notice the similarity with linear erasure codes introduced in other contexts, e.g., [6].) However, the in-network processing done by the nodes makes the basic network coding scheme extremely susceptible to *malicious* errors introduced by even a single intermediate node in the network. For starters, this is because the basic

network coding scheme offers no means of isolating the fault: if one of the vectors $\mathbf{w}_i$ received at the destination is incorrect, then that error will be "spread" across (potentially) every block $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ of the reconstructed file (cf. Equation 1). Furthermore, a single error introduced by one malicious node will be propagated by every node further downstream. Thus, even a faulty transmission on a single edge (say, due to a single corrupted node) will eventually cause almost all vectors being forwarded in the network to be incorrect, and will thus prevent reconstruction of even a portion of the file.

It is worth mentioning two trivial approaches that do not solve the problem. The source cannot simply sign the packets it releases into the network using a standard signature scheme, since the packets received at the destination will almost surely be different from those issued by the sender. Signing the entire file does not work either: although this would ensure that the receiver never accepts an incorrect file, there is no efficient way for the receiver to recover the correct file. (Since the receiver cannot distinguish correct packets from corrupt packets *a priori*, it is forced to apply the reconstruction procedure from Section 2.1 to all subsets of received vectors of size m.)

We now survey other techniques for combatting data pollution when network coding is used (see [21] for further discussion).

Information-theoretic approaches. Information-theoretic methods for enabling recovery from malicious faults work by introducing *redundancy* into the original packets transmitted by the sender [15, 17, 18]. Such techniques have the advantage of not relying on any computational assumptions, but are limited to offering security only against a relatively limited class of adversaries: these constructions all (inherently) assume limitations on the number of nodes the adversary can corrupt, the number of packets that can be modified, and/or the number of links on which the adversary can eavesdrop. Moreover, the communication overhead introduced by these schemes is significant.

Cryptographic approaches. Existing cryptographic schemes (i.e., those that protect only against a computationally bounded adversary) all work by providing a way for honest nodes to verify authenticity of *individual* packets. (Once again we stress that this cannot be achieved in our setting using standard signatures, since packets are modified in transit.) Cryptographic schemes can potentially offer resilience against an adversary who eavesdrops on the entire network and controls arbitrarily many malicious nodes, as long as the destination node receives m correctly formed and linearly independent vectors. Existing schemes also allow the receiver to recover gracefully when fewer than m legitimate vectors are received; for example, if the destination receives k correctly formed vectors spanning the subspace defined by the first k file blocks, then the receiver can at least recover a portion of the original file. Cryptographic schemes have the additional advantage that intermediate nodes in the network can verify correctness of individual packets, and hence reject ill-formed ones.

Although one could imagine using a symmetric-key approach, all existing work focuses on the public-key setting where the sender's public key is known to all other nodes in the network. A public-key scheme makes the most sense when the sender is multi-casting files to many receivers (as is typically the situation when network coding is used), and furthermore enables all intermediate nodes in the network to potentially verify authenticity of received packets.

Krohn et al. [22] (see also [11, 12]) suggest *homomorphic hashing* for preventing pollution attacks. In their scheme, the sender computes a hash $h_i = H(\bar{\mathbf{v}}_i)$ of each block of the file; given $\mathbf{x} = (h_1, \ldots, h_m)$, anyone can check whether a packet $\mathbf{w}$ is a correctly formed linear combination of the augmented vectors $\{\mathbf{v}_i\}$. Krohn et al. assume a reliable channel for distributing the hash

values for a given file, but it is not hard to show (see Section 5) that signing $\mathbf{x}$ using a standard signature scheme also results in a secure solution. The drawback of this approach is that both the authentication information $\mathbf{x}$ and the public keys are large: $\mathbf{x}$ has size $\Theta(km)$ and the public key has size $\Theta(kn)$, where k is a cryptographic security parameter. (Thus, either the public key or $\mathbf{x}$ has size at least the square root of the file size.) Sending all of $\mathbf{x}$ with each packet introduces a large overhead; on the other hand, if $\mathbf{x}$ is partitioned among multiple packets then intermediate nodes cannot verify authenticity of the packets they receive.

Zhao et al. [25] propose a scheme where the sender computes some authentication information $\mathbf{x}$ derived from a vector *orthogonal* to the space $V = \text{span}(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\})$; this authentication information $\mathbf{x}$ is then signed by the sender (using a standard signature scheme). Unfortunately, this scheme also has relatively poor performance: both $\mathbf{x}$ and the public keys have size $\Theta(k(n+m))$. Furthermore, the scheme can only be used for distributing a single file, after which the public key must be refreshed. (Zhao et al. suggest some approaches for handling multiple files, but do not prove security of any of these suggestions.) An additional drawback of both this scheme and the one of Krohn et al. is that they both require the sender to know the entire file before the authentication information can be computed.

Charles et al. [7] present a *homomorphic signature scheme* [19] based on the aggregate signature scheme of Boneh et al. [4]. This scheme has the property that valid signatures $\sigma_1, \ldots, \sigma_k$ on vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$, respectively, can be combined, without knowledge of the signer's secret key, to produce a valid signature σ on any linear combination $\sum_i \alpha_i \mathbf{w}_i$. The scheme can only be used to sign a *single* file, after which the public key must be refreshed; this restriction clearly limits the scheme's applicability. Public keys in this scheme have size $\Theta(k(m+n))$, meaning that it will be impractical to redistribute public keys over the network even if network coding is used for key distribution. Charles et al. also do not formally prove security of their scheme.

2.3 Our Contributions

We start with clean definitions of the problem at hand and formally define what it means for a signature scheme to be secure in our context. Roughly speaking, we consider signature schemes that can be viewed as authenticating *linear subspaces* in the sense that a signature on the subspace V authenticates all vectors in V. Our security definition requires that no adversary given a signature on a vector subspace V can forge a valid signature for any vector not in V. The application to network coding is clear: to distribute a file, simply sign the subspace $V = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$. (Actually, both our security definition and our constructions directly take into account the distribution of multiple files using a single public key — in contrast to [7, 25] — and so the formal definition and the application to network coding is a bit more involved.)

We show two constructions meeting our definition. Our first scheme, called NCS_1, is a homomorphic signature scheme and has the advantage that signatures can be associated with individual vectors rather than an entire subspace. (The signature on a linear subspace V can then be taken as the collection of signatures on a set of basis vectors for V.) Both the public key and per-vector signatures in this scheme have *constant* size, making the scheme ideally suited for network coding. This scheme also supports the transmission of *streaming data*, in the sense that the sender need not be aware of the entire file before computing the signature on the first packet. Security of this scheme is proved based on the computational Diffie-Hellman assumption in bilinear groups, in the random oracle model.

Our second construction, called NCS_2, provides an instantiation of the scheme of Krohn et

al. [22] that is secure according to our definition. The primary advantage of this scheme is that it can be proven secure based on a potentially weaker assumption (namely, the discrete logarithm assumption) without random oracles. We also show how our scheme can be viewed as a more efficient version of the scheme of Zhao et al. [25].

Finally, we prove a lower bound on the length of secure signatures for linear subspaces, under some mild assumptions on the signature scheme. Specifically, we show (roughly speaking) that the signature on any subspace V must have length proportional to $\dim(V)$. This shows that our two constructions are essentially optimal in this regard. (Note that although our first scheme offers constant size per-vector signatures, the signature on a subspace V consists of $\dim(V)$ per-vector signatures and thus matches the lower bound.)

3 Definitions and Preliminaries

3.1 Signing a Linear Subspace

We abstract our problem, and seek to design a signature scheme that signs a subspace $V \subset \mathbb{F}_p^N$ so that any $\mathbf{y} \in V$ is accepted as valid. We start by defining the abstract interface provided by such a system and then define security.

As discussed previously, we want our scheme to be useful for the distribution of multiple files using the same public key. As such, every file will be associated with an *identifier* id that is chosen by the sender at the time the first packet associated with the file is transmitted.[1] We then require that every packet forwarded in the system is labeled with the appropriate identifier. (Adversarial nodes, of course, can change the identifier any way they like.) The identifier provides a mechanism for honest nodes, and the receiver in particular, to distinguish packets associated with different files.

Definition 1. A *network coding signature scheme* is a triple of probabilistic, polynomial-time algorithms (Setup, Sign, Verify) with the following functionality:

- $\mathsf{Setup}(1^k, N)$. On input a security parameter 1^k and an integer N, this algorithm outputs a prime p, a public key PK, and a secret key SK.
- $\mathsf{Sign}(SK, \mathsf{id}, V)$. On input a secret key SK, a file identifier $\mathsf{id} \in \{0,1\}^k$, and an m-dimensional subspace $V \subset \mathbb{F}_p^N$ (with $0 < m < N$) described as a set of basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, this algorithm outputs a signature σ.
- $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. On input a public key PK, an identifier $\mathsf{id} \in \{0,1\}^k$, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ, this algorithm outputs either 0 (reject) or 1 (accept).

We require that for each (p, PK, SK) output by $\mathsf{Setup}(1^k, N)$, the following holds: for all m-dimensional subspaces $V \subset \mathbb{F}_p^N$ with $0 < m < N$, and for all $\mathsf{id} \in \{0,1\}^k$, if $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, V)$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$ for all $\mathbf{y} \in V$.

The signature σ output by Sign can be viewed a signature on the vector space V. "Homomorphic signatures" (cf. [19]) are a special case that is more precisely modeled by a definition in which the

[1] One can think of this identifier as being equivalent to a filename, though for our first scheme we require that identifiers be *unpredictable* (they need not be random). Unpredictability is easily achieved by concatenating an arbitrary filename with a random string.

Sign algorithm produces signatures $\sigma_1, \ldots, \sigma_m$ on the basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, and the collection of these signatures constitutes a signature on V. This is encapsulated in the following definition.

Definition 2. A *homomorphic network coding signature scheme* is a tuple of probabilistic, polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ with the following functionality:

- $\mathsf{Setup}(1^k, N)$. On input a security parameter 1^k and an integer N, this algorithm outputs a prime p, a public key PK, and a secret key SK.
- $\mathsf{Sign}(SK, \mathsf{id}, \mathbf{v})$. On input a secret key SK, a file identifier $\mathsf{id} \in \{0,1\}^k$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm outputs a signature σ.
- $\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. On input a public key PK, a file identifier id, and a set of tuples $\{(\beta_i, \sigma_i)\}_{i=1}^{\ell}$ with $\beta_i \in \mathbb{F}_p$, this algorithm outputs a signature σ.
  (The intuition is that if each σ_i is a valid signature on the vector $\mathbf{v}_i$, then σ is a signature on $\sum_{i=1}^{\ell} \beta_i \mathbf{v}_i$.)
- $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. On input a public key PK, an identifier $\mathsf{id} \in \{0,1\}^k$, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ, this algorithm outputs either 0 (reject) or 1 (accept).

We require that for each (p, PK, SK) output by $\mathsf{Setup}(1^k, N)$, the following hold:

- For all id and all $\mathbf{y} \in \mathbb{F}_p^N$, if $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, \mathbf{y})$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$.
- For all $\mathsf{id} \in \{0,1\}^k$ and all sets of triples $\{(\beta_i, \sigma_i, \mathbf{v}_i)\}_{i=1}^{\ell}$, if it holds that $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{v}_i, \sigma_i) = 1$ for all i, then

$$\mathsf{Verify}\left(PK,\ \mathsf{id},\ \textstyle\sum_i \beta_i \mathbf{v}_i,\ \mathsf{Combine}\left(PK,\ \mathsf{id},\ \{(\beta_i, \sigma_i)\}_{i=1}^{\ell}\right)\right) = 1.$$

The following lemma, a proof of which is trivial, shows that homomorphic network coding signatures are indeed a special case of network coding signatures.

Lemma 3. *Let $\mathcal{S}_2 = (\mathsf{Setup}_2, \mathsf{Sign}_2, \mathsf{Combine}_2, \mathsf{Verify}_2)$ be a homomorphic network coding signature scheme. Then $\mathcal{S}_1 = (\mathsf{Setup}_1, \mathsf{Sign}_1, \mathsf{Verify}_1)$ defined as follows is a network coding signature scheme.*

- $\mathsf{Setup}_1(1^k, N)$ *runs* $\mathsf{Setup}_2(1^k, N)$ *and outputs the result.*
- $\mathsf{Sign}_1(SK, \mathsf{id}, V)$ *runs* $\mathsf{Sign}_2(SK, \mathsf{id}, \mathbf{v}_1)$, *...,* $\mathsf{Sign}_2(SK, \mathsf{id}, \mathbf{v}_m)$, *where* $\mathbf{v}_1$, *...,* $\mathbf{v}_m$ *is any basis of* V. *It then outputs* $\sigma = ((\mathbf{v}_1, \sigma_1), \ldots, (\mathbf{v}_m, \sigma_m))$.
- $\mathsf{Verify}_1(PK, \mathsf{id}, \mathbf{y}, \sigma)$ *parses* σ *as* $((\mathbf{v}_1, \sigma_1), \ldots, (\mathbf{v}_m, \sigma_m))$, *and computes coefficients* $\{\beta_i\}$ *such that* $\mathbf{y} = \sum_i \beta_i \mathbf{v}_i$ *(if no solution exists, then it outputs 0). Finally, it outputs*

$$\mathsf{Verify}_2\left(PK,\ \mathsf{id},\ \mathbf{y},\ \mathsf{Combine}_2\big(PK,\ \mathsf{id},\ \{(\beta_i, \sigma_i)\}_{i=1}^{m}\big)\right).$$

We say a basis $\{\mathbf{v}_i\}_{i=1}^m$ of a subspace $V \subseteq \mathbb{F}_p^N$ is *properly augmented* (cf. Section 2.1) if the last m coordinates of each $\mathbf{v}_i$ form a unit vector with a 1 in the ith position. Abusing notation, we say V is properly augmented if it is described using a properly augmented basis. Let

$n = N - m$. If $\mathbf{v}_1, \ldots, \mathbf{v}_m$ is a properly augmented basis of V, then for any $\mathbf{y} \in \mathbb{F}_p^N$ with $\mathbf{y} = (y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+m})$ we have

$$\mathbf{y} \in V \iff \mathbf{y} = \sum_{i=1}^{m} y_{n+i}\mathbf{v}_i. \tag{2}$$

This observation allows us to simplify the construction of Lemma 3 when we only use ($\mathsf{Setup}_1, \mathsf{Sign}_1$, Verify_1) to sign properly augmented vector spaces. (This suffices for our application to network coding.) Namely, $\mathsf{Sign}_1(SK, \mathsf{id}, V)$ simply outputs $\sigma = (\sigma_1, \ldots, \sigma_m)$ (where the σ_i are computed as in Lemma 3), and $\mathsf{Verify}_1(PK, \mathsf{id}, \mathbf{y}, \sigma)$ outputs

$$\mathsf{Verify}_2\left(PK,\ \mathsf{id},\ \mathbf{y},\ \mathsf{Combine}_2\big(PK,\ \mathsf{id},\ \{(y_{N-m+i}, \sigma_i)\}_{i=1}^{m}\big)\right).$$

Security. We define security of a network coding signature scheme, and say that a homomorphic network coding signature scheme $\mathcal{S}_2$ is secure if the network coding signature scheme $\mathcal{S}_1$ constructed from $\mathcal{S}_2$ as in Lemma 3 is secure.

Definition 4. A network coding signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is *secure* if the advantage of any probabilistic, polynomial-time adversary $\mathcal{A}$ in the following security game is negligible in the security parameter k:

Setup: The adversary $\mathcal{A}$ sends a positive integer N to the challenger. The challenger runs $\mathsf{Setup}(1^k, N)$ to obtain (p, PK, SK), and gives p and PK to $\mathcal{A}$.

Queries: Proceeding adaptively, $\mathcal{A}$ specifies a sequence of vector subspaces $V_i \subset \mathbb{F}_p^N$. For each i, the challenger chooses id_i uniformly from $\{0,1\}^k$, and gives id_i and $\sigma_i \leftarrow \mathsf{Sign}(SK, \mathsf{id}_i, V_i)$ to $\mathcal{A}$.

Output: $\mathcal{A}$ outputs $\mathsf{id}^* \in \{0,1\}^k$, a *non-zero* vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a signature σ^*.

The adversary *wins* if $\mathsf{Verify}(PK, \mathsf{id}^*, \mathbf{y}^*, \sigma^*) = 1$, and either (1) $\mathsf{id}^* \neq \mathsf{id}_i$ for all i (a *type 1 forgery*), or (2) $\mathsf{id}^* = \mathsf{id}_i$ for some i but $\mathbf{y}^* \notin V_i$ (a *type 2 forgery*). The *advantage* NC-Adv$[\mathcal{A}, \mathcal{S}]$ of $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins the security game.

We require the adversary to output a *non-zero* vector $\mathbf{y}^*$ since the zero vector lies in every linear subspace. (Furthermore, by adding a dimension it is possible to rule out type-1 forgeries on the zero vector if desired.) Note also that it is not counted as a forgery if $\mathcal{A}$ obtains a signature σ on a vector space V and outputs a valid signature σ' on a vector space $V' \subset V$. Indeed, in the context of network coding this would not be problematic.

3.2 Bilinear Groups and Complexity Assumptions

We briefly review the framework of groups with bilinear maps.

Definition 5. A *bilinear group tuple* is a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ with the following properties:

1. $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, in which random sampling and group operations are efficiently computable.

2. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable map satisfying the following:

(a) Bilinearity: for any $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$, $e(g^a, h^b) = e(g, h)^{ab}$.

(b) Non-degeneracy: if g generates $\mathbb{G}_1$ and h generates $\mathbb{G}_2$, then $e(g, h)$ generates $\mathbb{G}_T$.

3. $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism.[2]

For cryptographic applications, we require that the *discrete logarithm problem* — i.e., computing x given g and g^x — be infeasible in the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Given an algorithm $\mathcal{A}$ that takes as input two elements g, h in a group $\mathbb{G}$ and outputs an integer x in $\{0, \ldots, |\mathbb{G}| - 1\}$, we define DL-Adv$[\mathcal{A}, \mathbb{G}]$ to be the probability that $h = g^x$, taken over inputs (g, h) and the random coins of $\mathcal{A}$.

Currently the only known bilinear group tuples in which the discrete logarithm problems are believed to be infeasible are those for which $\mathbb{G}_1, \mathbb{G}_2$ are (subgroups of) groups of rational points on elliptic curves or abelian varieties over finite fields; $\mathbb{G}_T$ is (a subgroup of) a multiplicative group of a finite field; and e is (a variant of) the Weil pairing or Tate pairing [9]. Elliptic curves and abelian varieties with the desired properties are called "pairing-friendly." Bilinear group tuples as described in Definition 5 exist on all pairing-friendly elliptic curves and abelian varieties with embedding degree $k > 1$.

The proof of security of our first signature scheme (Section 4) relies on the assumption that the *co-computational Diffie Hellman* (*co-CDH*) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ — i.e., computing $g^x \in \mathbb{G}_1$ given $g \in \mathbb{G}_1 \setminus \{1\}$ and $h, h^x \in \mathbb{G}_2 \setminus \{1\}$ — is infeasible. Given $\mathcal{A}$ that takes as input $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $z = h^x \in \mathbb{G}_2$, and outputs an element $\omega \in \mathbb{G}_1$, we define co-CDH-Adv$[\mathcal{A}, (\mathbb{G}_1, \mathbb{G}_2)]$ to be the probability that $\omega = g^x$, taken over inputs (g, h, z) and the random coins of $\mathcal{A}$. Note that if in addition to $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ there is an efficiently computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ then the co-CDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is equivalent to the standard computational Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$.

4 A Homomorphic Network Coding Signature Scheme

In this section we construct a homomorphic network coding signature scheme with constant-size public key and constant-size per-vector signatures. Our signatures are modeled on the pairing-based aggregate signatures of Boneh et al. [4].

Signature Scheme NCS$_0$.

Setup$(1^k, N)$. *Given a security parameter 1^k and a positive integer N, do:*

1. *Generate a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ such that $p > 2^k$. Choose a generator $h \xleftarrow{R} \mathbb{G}_2 \setminus \{1\}$.*
2. *Choose $\alpha \xleftarrow{R} \mathbb{F}_p^*$, and set $u := h^\alpha$.*
3. *Let $H : \{0,1\}^* \times \{0,1\}^* \to \mathbb{G}_1$ be a hash function, viewed as a random oracle.*
4. *Output p, the public key $PK := (\mathcal{G}, H, h, u)$ and the private key $SK := \alpha$.*

[2] Existence of φ is not needed if we use a different cryptographic assumption to prove security of our first scheme. We omit further discussion.

$\mathsf{Sign}(SK, \mathsf{id}, \mathbf{v})$. *Given a secret key* $SK = \alpha$, *an identifier* $\mathsf{id} \in \{0,1\}^k$, *and a vector* $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}_p^N$, *this algorithm outputs the signature*

$$\sigma := \left(\prod_{i=1}^{N} H(\mathsf{id}, i)^{v_i} \right)^{\alpha}.$$

$\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. *Given a public key* PK, *an identifier* id, *and* $\{(\beta_i, \sigma_i)\}_{i=1}^{\ell}$ *with* $\beta_i \in \mathbb{F}_p$, *this algorithm outputs* $\sigma := \prod_{i=1}^{\ell} \sigma_i^{\beta_i}$.

$\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. *Given a public key* $PK = (\mathcal{G}, H, h, u)$, *an identifier* id, *a signature* σ, *and a vector* $\mathbf{y} \in \mathbb{F}_p^N$, *define*

$$\gamma_1(PK, \sigma) \stackrel{\text{def}}{=} e(\sigma, h) \quad \text{and} \quad \gamma_2(PK, \mathsf{id}, \mathbf{y}) \stackrel{\text{def}}{=} e\left(\prod_{i=1}^{N} H(\mathsf{id}, i)^{y_i},\ u \right).$$

If $\gamma_1(PK, \sigma) = \gamma_2(PK, \mathsf{id}, \mathbf{y})$ *this algorithm outputs* 1*; otherwise it outputs* 0.

Due to lack of space, we omit the (straightforward) proof of correctness.

A signature is just a single element of $\mathbb{G}_1$. Groups $\mathbb{G}_1$ whose elements can be represented using $\log_2 p$ bits can be obtained by using pairing-friendly elliptic curves of prime or near-prime order; see [10] for further details.

A variant of the above scheme is more efficient when only properly augmented vectors will be signed (as is the case for applications to network coding), and the dimension m of the resulting vector space is known at the time any vector is signed or verified. In this setting, the signer can choose random $g_1, \ldots, g_N \in \mathbb{G}_1$ at the time of key generation and publish these as part of the public key. To sign a vector $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}_p^N$ using the identifier id, the signer sets $n := N - m$ and computes

$$\sigma := \left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{n+i}} \prod_{j=1}^{n} g_j^{v_j} \right)^{\alpha}.$$

(Verification is changed in the obvious way.) In this variant, signing the augmented vector $\mathbf{v}$ is dominated by computing a single hash into $\mathbb{G}_1$ (taking time similar to a full exponentiation) plus n additional exponentiations in $\mathbb{G}_1$. The public key in this variant can be compressed to size linear in the security parameter k by generating $g_1, \ldots, g_N$ as the output of an independent hash function H' (also modeled as a random oracle).

We now prove the security of signature scheme NCS_0. More precisely, we consider the variant scheme (call it NCS_1) suggested above, and prove security of the network coding signature scheme NCS_1' constructed from NCS_1 as described in the optimization following Lemma 3. Our security proof assumes that only properly augmented vector spaces are signed. (We stress that NCS_0 itself is also secure, even without this assumption.)

Theorem 6. *Let* NCS_1' *be the network coding signature scheme constructed from* NCS_1 *via the (optimized) method of Lemma 3. Then* NCS_1' *is secure in the random oracle model assuming that the co-CDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is infeasible.*

In particular, let $\mathcal{A}$ be a polynomial-time adversary as in Definition 4. Then there exists a polynomial-time algorithm $\mathcal{B}$ that computes co-CDH in $(\mathbb{G}_1, \mathbb{G}_2)$, and such that

$$\text{co-CDH-Adv}[\mathcal{B}, (\mathbb{G}_1, \mathbb{G}_2)] \geq \text{NC-Adv}[\mathcal{A}, \text{NCS}_1'] - \frac{1}{p} - \frac{q_s(q_s + q_h)}{2^k},$$

where q_s and q_h are the number of signature and hash queries made by $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an adversary as in Definition 4, though recall we make the assumption that $\mathcal{A}$ only requests signatures on properly augmented vector spaces. We construct $\mathcal{B}$ that takes as input parameters $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ and inputs $g \in \mathbb{G}_1$ and $h, z \in \mathbb{G}_2$, with $z = h^x$, and outputs an element $\omega \in \mathbb{G}_1$. Algorithm $\mathcal{B}$ simulates the hash function H and the Setup and Sign algorithms of NCS_1', and works as follows.

Setup. $\mathcal{A}$ chooses an integer N, and $\mathcal{B}$ does the following:

1. Choose random $s_1, t_1, \ldots, s_N, t_N \in \mathbb{F}_p$, and set $g_i := g^{s_i}\varphi(h)^{t_i}$ for all i.
2. Output the public key $PK := (\mathcal{G}, H, g_1, \ldots, g_N, h, z)$.

Hash query. When $\mathcal{A}$ requests the value of $H(\mathsf{id}, i)$, algorithm $\mathcal{B}$ does:

1. If (id, i) has already been queried, return $H(\mathsf{id}, i)$.
2. If (id, i) has not been queried, choose $\varsigma_i, \tau_i \xleftarrow{\text{R}} \mathbb{F}_p$ and set $H(\mathsf{id}, i) := g^{\varsigma_i}\varphi(h)^{\tau_i}$.

Sign. When $\mathcal{A}$ requests a signature on a vector space $V \subset \mathbb{F}_p^N$, described by properly augmented basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_p^N$ (where $\mathbf{v}_i = (v_{i1}, \ldots, v_{iN})$), algorithm $\mathcal{B}$ does the following:

1. Choose a random $\mathsf{id} \xleftarrow{\text{R}} \{0,1\}^k$. If $H(\mathsf{id}, i)$ has already been queried for some $i \in \{1, \ldots, N\}$ then abort. (The simulation has failed.)
2. Set $n := N - m$ and compute $\varsigma_i := -\sum_{j=1}^{n} s_j v_{ij}$ for $i = 1, \ldots, m$.
3. Choose $\tau_i \xleftarrow{\text{R}} \mathbb{F}_p$ and set $H(\mathsf{id}, i) := g^{\varsigma_i}\varphi(h)^{\tau_i}$ for $i = 1, \ldots, m$. Set $\mathbf{t} := (t_1, \ldots, t_n, \tau_1, \ldots, \tau_m)$.
4. Compute $\sigma_i := \varphi(z)^{\mathbf{v}_i \cdot \mathbf{t}}$ for $i = 1, \ldots, m$.
5. Output id and $\sigma := (\sigma_1, \ldots, \sigma_m)$.

Output. If $\mathcal{B}$ does not abort, then eventually $\mathcal{A}$ outputs a signature $\sigma = (\sigma_1, \ldots, \sigma_m)$, an identifier id, and a non-zero vector $\mathbf{y}$.

1. If id is not one of the identifiers used to answer a previous signature query, then compute $H(\mathsf{id}, i)$ as above for $i = 1, \ldots, m$. Thus, in any case, $H(\mathsf{id}, i) = g^{\varsigma_i}\varphi(h)^{\tau_i}$ for ς_i, τ_i known to $\mathcal{B}$.
2. Set $\mathbf{s} := (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{t} := (t_1, \ldots, t_n, \tau_1, \ldots, \tau_m)$. If $\mathbf{s} \cdot \mathbf{y} = 0$ then abort.
3. Set $n := N - m$ and output $\omega := \left(\dfrac{\prod_{i=1}^{m} \sigma_i^{y_{n+i}}}{\varphi(z)^{\mathbf{t} \cdot \mathbf{y}}} \right)^{1/(\mathbf{s} \cdot \mathbf{y})}$.

We first observe that the responses to all hash queries are independent and uniformly random in $\mathbb{G}_1$. We also observe that the $g_1, \ldots, g_N$ are random group elements, and thus the public key PK output by $\mathcal{B}$ is distributed identically to the public key produced by the real Setup algorithm.

Next we show that the signatures σ output by $\mathcal{B}$ are identical to the signatures that would be produced by the real Sign algorithm given the public key PK and the hash answers computed by $\mathcal{B}$. Since the secret key corresponding to PK is x, it suffices to show that for each vector $\mathbf{v}$ "signed" by $\mathcal{B}$, we have

$$\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{n+i}} \prod_{j=1}^{n} g_j^{v_j} \right)^x = \varphi(z)^{\mathbf{v} \cdot \mathbf{t}}, \tag{3}$$

where the left-hand side is the "real" signature and the right-hand side is the signature computed by $\mathcal{B}$. The left-hand side is equal to

$$\left(\prod_{i=1}^{m} (g^{\varsigma_i} \varphi(h)^{\tau_i})^{v_{n+i}} \prod_{j=1}^{n} \left(g^{s_j} \varphi(h)^{t_j} \right)^{v_j} \right)^x = \left(g^{\mathbf{s} \cdot \mathbf{v}} \varphi(h)^{\mathbf{t} \cdot \mathbf{v}} \right)^x. \tag{4}$$

Now observe that we constructed $\mathbf{s}$ so that $\mathbf{s} \in V^\perp$ (i.e., $\mathbf{s} \cdot \mathbf{v} = 0$ for all $\mathbf{v} \in V$), so the expression (4) is equal to $\varphi(h)^{x(\mathbf{t} \cdot \mathbf{v})}$. Equation (3) now follows from the fact that $\varphi(z) = \varphi(h)^x$.

We next analyze the probability that $\mathcal{B}$ aborts while interacting with $\mathcal{A}$. There are two scenarios in which this can happen: if $\mathcal{B}$ responds to two different signature queries by choosing the same identifier id, or if $\mathcal{B}$ responds to a signature query by choosing an identifier id such that $\mathcal{A}$ has already requested the value of $H(\mathsf{id}, i)$ for some i. The probability of the first event is at most $q_s^2/2^k$, while the probability of the second event is at most $q_s q_h / 2^k$.

Suppose $\mathcal{B}$ does not abort and $\mathcal{A}$ outputs a signature σ, an identifier id, and a non-zero vector $\mathbf{y}$. Let $\sigma = (\sigma_1, \ldots, \sigma_m)$. If $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$ then

$$e\left(\prod_{i=1}^{m} \sigma_i^{y_{n+i}},\, h \right) = e\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{y_{n+i}} \prod_{j=1}^{n} g_j^{y_j},\, z \right).$$

By the same reasoning as above the right-hand side is equal to

$$e\left(g^{\mathbf{s} \cdot \mathbf{y}} \varphi(h)^{\mathbf{t} \cdot \mathbf{y}}, z \right) = e\left(g^{x(\mathbf{s} \cdot \mathbf{y})} \varphi(z)^{\mathbf{t} \cdot \mathbf{y}}, h \right),$$

where $\mathbf{s}$ and $\mathbf{t}$ are determined from id as in steps (1) and (2) of $\mathcal{B}$'s output procedure. The non-degeneracy of e then implies that

$$\prod_{i=1}^{m} \sigma_i^{y_{n+i}} = g^{x(\mathbf{s} \cdot \mathbf{y})} \varphi(z)^{\mathbf{t} \cdot \mathbf{y}}.$$

It follows that if $\mathbf{s} \cdot \mathbf{y} \neq 0$ then the element ω output by $\mathcal{B}$ is equal to g^x.

To complete the proof, we show that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. In preparation, we first prove the following lemma:

Lemma 7. *Assume $\mathcal{B}$ does not abort. Then the variables $s_1, \ldots, s_N$ are each independently uniform in $\mathbb{F}_p$ even conditioned on the view of $\mathcal{A}$.*

Proof. In proving the lemma we can ignore any queries $H(\mathsf{id}, i)$ where id is not an identifier used to respond to a signing query, since (a) the variables $s_1, \ldots, s_N$ are not involved in these queries, and (b) the variables that are involved in these queries are not involved in any other interaction between $\mathcal{A}$ and $\mathcal{B}$.

We show that for any given view of $\mathcal{A}$ and any choice of values for $s_1, \ldots, s_N$, there is a unique choice of values for all of the other variables in the system that is consistent with the adversary's view. The adversary's view consists of the public key PK and the signatures on subspaces V_k for $k = 1, \ldots, q_s$. Let $m_k = \dim V_k$. Hence, the adversary's view is derived from $2N + \sum 2m_k$ random variables:

- The public key is derived from s_j, t_j for $j = 1, \ldots, N$.
- The kth signature is derived from the s_j, t_j and ς_i, τ_i for $i = 1, \ldots, m_k$. (Here we use the fact that $\mathcal{B}$ did not abort, and so no two signing queries use the same value of id.)

Moreover, the adversary has $N + \sum 3m_k$ linear relations on these variables:

- N relations derived from the public key,
- m_k relations derived from the values of $H(\mathsf{id}, i)$ for the kth query,
- m_k relations derived from the signature $(\sigma_1, \ldots, \sigma_{m_k})$ for the kth query,
- m_k relations derived from the fact that $\mathbf{s} \in V^\perp$ for the kth query.

We set the following notation:

$$\begin{aligned}
\mathbf{s}^L &= (s_1, \ldots, s_N) \\
\mathbf{s}_k^R &= (\varsigma_1, \ldots, \varsigma_{m_k}) \text{ for the } k\text{th signature query} \\
\mathbf{t}^L &= (t_1, \ldots, t_N) \\
\mathbf{t}_k^R &= (\tau_1, \ldots, \tau_{m_k}) \text{ for the } k\text{th signature query.}
\end{aligned}$$

Let $\bar{V}_k$ be the $m_k \times N$ matrix whose ith row consists of the first $N - m_k$ entries (i.e., the unaugmented part) of the basis vector $\mathbf{v}_i$ for the kth query, followed by m_k zeroes. Let $\varphi(h) = g^\alpha$. The view of $\mathcal{A}$ imposes the following constraints:

$$\begin{aligned}
\mathbf{s}^L + \alpha \mathbf{t}^L &= \mathbf{c}_1 && \text{(public key)} && (5) \\
\mathbf{s}_k^R + \alpha \mathbf{t}_k^R &= \mathbf{c}_{2,k} && \text{(values of } H(\mathsf{id}, i)) && (6) \\
\bar{V}_k \mathbf{t}^L + \mathbf{t}_k^R &= \mathbf{c}_{3,k} && \text{(signatures)} && (7) \\
\bar{V}_k \mathbf{s}^L + \mathbf{s}_k^R &= \mathbf{0} && (\mathbf{s} \in V^\perp) && (8)
\end{aligned}$$

for some vectors $\mathbf{c}_1 \in \mathbb{F}_p^N$, $\mathbf{c}_{2,k} \in \mathbb{F}_p^{m_k}$, $\mathbf{c}_{3,k} \in \mathbb{F}_p^{m_k}$ that are determined (in an information-theoretic sense) from the view of $\mathcal{A}$. We wish to show that the system has a unique solution for any value of $\mathbf{s}^L$.

Observe that equation (8) is linearly dependent on equations (5), (6), and (7). Specifically, for each k we have $(8) = \bar{V}_k(5) + (6) - \alpha(7)$. Since the system has at least one solution by construction, any choice of variables satisfying equations (5)–(7) must also satisfy (8).

Suppose $\mathbf{s}^L$ is fixed; then equation (5) determines a unique value for $\mathbf{t}^L$. For each k, equation (7) and this value of $\mathbf{t}^L$ determine a unique value for $\mathbf{t}_k^R$, from which equation (6) determines a unique value of $\mathbf{s}_k^R$. Thus for any value of $\mathbf{s}^L$ there is a unique solution to equations (5)–(8). We conclude that $\mathbf{s}^L = (s_1, \ldots, s_N)$ is uniform in $\mathbb{F}_p^N$ even conditioned on the view of $\mathcal{A}$. □

To complete the proof of Theorem 6, we now show that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. If $\mathcal{A}$ outputs a type 1 forgery, then id was not used in response to any previous signing query. In this case the $\{\varsigma_i\}$ are independently uniform in $\mathbb{F}_p$ even conditioned on the view of $\mathcal{A}$. By Lemma 7 the $\{s_i\}$ are also independently uniform in $\mathbb{F}_p$ conditioned on $\mathcal{A}$'s view. Since $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{y}$ is non-zero, it follows that $\mathbf{s} \cdot \mathbf{y}$ is uniformly distributed in $\mathbb{F}_p$, and thus the probability that $\mathbf{s} \cdot \mathbf{y} = 0$ is $1/p$.

Now suppose that $\mathcal{A}$ outputs a type 2 forgery, so id was used in response to the signing query for some vector subspace V, and $\mathbf{y} \notin V$. (Note that id was used to answer only one signing query, otherwise $\mathcal{B}$ aborts.) By Lemma 7 the variables $\{s_i\}$ are independently uniform in $\mathbb{F}_p$ even conditioned on the adversary's view. This implies that, conditioned on the adversary's view, the vector $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ is uniformly random in $V^\perp$. So for any $\mathbf{y} \notin V$ we see that $\mathbf{s} \cdot \mathbf{y}$ is uniform in $\mathbb{F}_p$, and we conclude that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. This completes the proof. □

5 Network Coding Signatures without Random Oracles

Krohn et al. [22] propose authenticating network coding data using a homomorphic hash function (see below). As in Definition 2, their scheme produces a signature σ_i on each basis vector of the subspace to be authenticated. Their system is not secure according to our definition, however, as there is no mechanism to ensure that basis vectors from different files cannot be combined. Our solution is to authenticate all the hash values (along with the file identifier) using a standard signature scheme, which we denote by $\mathcal{S}_0 = (\mathsf{Gen}_0, \mathsf{Sign}_0, \mathsf{Verify}_0)$. This modification produces a secure network coding signature scheme (as in Definition 1) but eliminates the homomorphic property. The scheme can thus be used to sign entire subspaces, but not individual vectors.

Signature Scheme $\mathbf{NCS}_2$.

$\mathsf{Setup}(1^k, N)$. *Given a security parameter 1^k and a positive integer N do:*

1. *Choose a group $\mathbb{G}$ of prime order $p > 2^k$.*
2. *Run $\mathsf{Gen}_0(1^k)$ and let the public/private keys be PK_0, SK_0.*
3. *Choose generators $g_1, \ldots, g_N \xleftarrow{\mathrm{R}} \mathbb{G} \setminus \{1\}$.*
4. *Output the prime p, the public key $PK := (\mathbb{G}, g_1, \ldots, g_N, PK_0)$, and the private key $SK := SK_0$.*

$\mathsf{Sign}(SK, \mathsf{id}, V)$. *Given a secret key SK, a file identifier id, and an m-dimensional subspace $V \subset \mathbb{F}_p^N$ described by a properly augmented basis $\mathbf{v}_1, \ldots, \mathbf{v}_m$, do:*

1. *Set $n := N - m$. Compute $\sigma_i := \prod_{j=1}^{n} g_j^{-v_{ij}}$ for $i = 1, \ldots, m$.*
2. *Compute $\tau \leftarrow \mathsf{Sign}_0(SK, (\mathsf{id}, \sigma_1, \ldots, \sigma_m))$.*
3. *Output $\sigma := (\sigma_1, \ldots, \sigma_m, \tau)$.*

$\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. *Given a public key $PK = (\mathbb{G}, g_1, \ldots, g_N, PK_0)$, an identifier id, a signature $\sigma = (\sigma_1, \ldots, \sigma_m, \tau)$, and a vector $\mathbf{y} \in \mathbb{F}_p^N$, do:*

1. *Run $\mathsf{Verify}_0(PK_0, (\mathsf{id}, \sigma_1, \ldots, \sigma_m), \tau)$. If the answer is 0, output 0.*

2. If $\left(\prod_{j=1}^{n} g_j^{y_j}\right)\left(\prod_{i=1}^{m} \sigma_i^{y_{n+i}}\right) = 1$ *then output 1; otherwise output 0.*

We omit the (straightforward) proof of correctness.

If elements of $\mathbb{G}$ are represented using $\log_2 p$ bits and the signature scheme $\mathcal{S}_0$ produces signatures of size $\log_2 p$, then the size of the signature σ is $(m+1)\log_2 p$ bits. If one is willing to use the random oracle model, we can achieve a constant-size public key by letting the values $g_1, \ldots, g_N$ be computed as the output of a hash function H (viewed as a random oracle).

Theorem 8. *Assume $\mathcal{S}_0$ is a secure signature scheme. Then* NCS_2 *is secure assuming hardness of the discrete logarithm problem in $\mathbb{G}$.*

In particular, let $\mathcal{A}$ be a polynomial-time adversary as in Definition 4. Then there exists a polynomial-time adversary $\mathcal{B}_1$ that forges signatures for $\mathcal{S}_0$ and a polynomial-time algorithm $\mathcal{B}_2$ that computes discrete logarithms, such that

$$\text{Sig-Adv}[\mathcal{B}_1, \mathcal{S}_0] + 2 \cdot \text{DL-Adv}[\mathcal{B}_2, \mathbb{G}] \geq \text{NC-Adv}[\mathcal{A}, \mathrm{NCS}_2] - \frac{q_s^2}{2^k},$$

where q_s denotes the number of signature queries made by $\mathcal{A}$, and $\text{Sig-Adv}[\mathcal{B}_1, \mathcal{S}_0]$ *is the probability that $\mathcal{B}_1$ wins the security game for the standard signature scheme $\mathcal{S}_0$ (see [20, §12.2]).*

Proof. Suppose algorithm $\mathcal{A}$ produces a signature $\sigma = (\sigma_1, \ldots, \sigma_m, \tau)$, an identifier id, and a vector $\mathbf{y}$ such that $\mathsf{Verify}(PK, \mathsf{id}, \sigma, \mathbf{y}) = 1$. If id is not one of the identifiers returned on a signature query (type 1 forgery), then $\mathcal{A}$ has forged a signature relative to $\mathcal{S}_0$. In case of a type 2 forgery, say id was used in response to a unique signature query on the vector space V, and that $\mathbf{y} \notin V$. Define H via $H(\mathbf{v}) = \prod_{j=1}^{n} g_j^{v_j}$. Since $\mathbf{y} \notin V$ we have $\mathbf{y}' := \sum_{i=1}^{m} y_{n+i}\mathbf{v}_i \neq \mathbf{y}$. The fact that the signature verifies implies that $H(\mathbf{y}) = H(\mathbf{y}')$, and thus we have produced a collision for $H(\cdot)$. By standard arguments [5, 2], an algorithm $\mathcal{A}$ that produces such a collision with probability ε can be used to compute discrete logarithms in $\mathbb{G}$ with probability at least $\varepsilon/2$. □

Relation with [25]. Signature Scheme NCS_2 can also be viewed as a secure instantiation of the signature scheme proposed by Zhao et al. [25]. Their scheme computes a signature on V by choosing a random vector $\mathbf{u} \in V^{\perp}$ and outputting $(h_1, \ldots, h_N) = (g^{u_1}, \ldots, g^{u_N})$ along with a signature on this tuple. Verification of $\mathbf{y}$ involves checking whether $\prod_{j=1}^{N} h_j^{y_j} = g^{\mathbf{u}\cdot\mathbf{y}} = 1$. Correctness follows from the fact that $\mathbf{u} \in V^{\perp}$, and security follows from the fact that computing $\mathbf{y} \notin V$ such that $\mathbf{u} \cdot \mathbf{y} = 0$ permits computation of discrete logarithms in $\mathbb{G}$ (see [25, Theorem 1] or [3, Lemma 3.2]).

To view the scheme NCS_2 from this perspective, fix a generator g of $\mathbb{G}$ and write the first $n = N - m$ elements of the public key as $g^{u_1}, \ldots, g^{u_n}$. Letting $u_{n+i} = \sum_{j=1}^{n} -u_j v_{ij}$, we have $\sigma_i = g^{u_{n+i}}$. Furthermore, if $\{\mathbf{v}_i\}_{i=1}^{m}$ is a properly augmented basis of V then the vector $\mathbf{u} = (u_1, \ldots, u_N)$ is in $V^{\perp}$. The verification step then computes $g^{\mathbf{u}\cdot\mathbf{y}}$ just as in the scheme of Zhao et al.

The signatures produced by scheme NCS_2 have length $\Theta(m)$ and are thus much shorter than the signatures produced by the scheme of Zhao et al., which have length $\Theta(N)$.

6 A Lower Bound on Signature Size

We now prove a lower bound on the length of signatures for linear subspaces. Our lower bound applies to network coding signature schemes (Definition 1) that have the following two properties:

Additivity: For any PK, id, σ, and vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_p^N$, if $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{u}, \sigma) = \mathsf{Verify}(PK, \mathsf{id}, \mathbf{v}, \sigma) = 1$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{u} + \mathbf{v}, \sigma) = 1$. Both our constructions are additive.

Fixed size: For a given $m > 0$ and a given SK, the size of the signature output by $\mathsf{Sign}(SK, \mathsf{id}, V)$ is the same for all identifiers id and m-dimensional spaces $V \subset \mathbb{F}_p^N$. Again, this holds for both of the systems in this paper. We make this assumption primarily to simplify the presentation; a version of our bound holds even if this property is not satisfied.

For a secret key SK and integers N, m let $\ell_{SK,N,m}$ be the length in bits of signatures $\mathsf{Sign}(SK, \mathsf{id}, V)$ where V is an m-dimensional subspace of $\mathbb{F}_p^N$.

For signatures that have these two properties, we show that the signature size must be at least (roughly) $m \log_2 p$ bits. In particular, we construct an attack algorithm that forges signatures whenever $\ell_{SK,N,m}$ is shorter than this bound. Hence, if the scheme is to be secure then for almost all secret keys the signature size must be greater than our bound.

The intuition behind our lower bound is that if signatures are short, then by the pigeonhole principle there is a large set $\mathcal{V}$ of linear subspaces that all have the same signature σ. If signatures are sufficiently short, then the direct sum of the spaces in $\mathcal{V}$ spans all of $\mathbb{F}_p^N$. Since the signature scheme is additive this implies that $\mathsf{Verify}(PK, \mathsf{id}, \sigma, \mathbf{y}) = 1$ for *any* $\mathbf{y} \in \mathbb{F}_p^N$; we will call a signature σ with this property (for a fixed identifier id) *trivial*. We conclude that there are many subspaces V with trivial signatures; the system can then be easily attacked by choosing a random subspace V, obtaining a signature on V, and producing a vector $\mathbf{y} \notin V$.

Theorem 9. *Let m, N be integers with $0 < m < N$, and let $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ be a network coding signature scheme satisfying the two properties above. Then there is a polynomial-time adversary $\mathcal{A}$ making a single signature query and such that the following holds: when the secret key SK used in the security game of Definition 4 satisfies*

$$\ell_{SK,N,m} \leq m \log_2 p - 4m/p - 1, \tag{9}$$

then $\mathcal{A}$ wins with probability at least $1/2$.

Proof. Fix a public/private key pair PK, SK. When the adversary queries a vector space V to the challenger, the challenger produces an identifier id uniformly at random from the space $\mathcal{I}$ of identifiers; in particular, id is independent of V. We may thus fix the randomness of the challenger in advance and let id_1 be the identifier produced on the first query. Although the Sign algorithm may be probabilistic, once we have fixed the randomness each m-dimensional subspace $V \subset \mathbb{F}_{p^N}$ is mapped to a *unique* signature $\sigma := \mathsf{Sign}(SK, \mathsf{id}_1, V)$.

We now proceed with a combinatorial argument. Let $n = N - m$. The number of m-dimensional subspaces $V \subset \mathbb{F}_p^{n+m}$ is the p-binomial coefficient [24, Proposition 1.3.18]

$$\binom{n+m}{m}_p = \frac{(p^{n+m}-1)(p^{n+m-1}-1)\cdots(p^{n+1}-1)}{(p^m-1)(p^{m-1}-1)\cdots(p-1)} > p^{mn}.$$

Let $\mathcal{U}$ be the set of vector spaces V such that the signature on V is nontrivial, and let β be the fraction of vector spaces V with nontrivial signatures; then the cardinality of $\mathcal{U}$ is at least $p^{mn}\beta$. Let α be the number of distinct nontrivial signatures produced by signing all vector spaces $V \in \mathcal{U}$ with identifier id_1. Then by the pigeonhole principle, there is a set of vector spaces $\mathcal{V} \subseteq \mathcal{U}$ of

cardinality at least $p^{mn}\beta/\alpha$ such that the signatures $\mathsf{Sign}(SK, \mathsf{id}_1, V)$ are identical for all $V \in \mathcal{V}$. Call this signature σ.

Let $W \subseteq \mathbb{F}_p^{n+m}$ be the direct sum of all the spaces in $\mathcal{V}$. Since the signature system is additive, we know that $\mathsf{Verify}(PK, \mathsf{id}_1, \mathbf{w}, \sigma) = 1$ for all $\mathbf{w} \in W$. If $W = \mathbb{F}_p^{n+m}$ then σ is trivial, contradicting the assumption that $\mathcal{V} \subseteq \mathcal{U}$. Thus W is a subspace of $\mathbb{F}_p^{n+m}$ of dimension at most $n+m-1$. Then the number of m-dimensional subspaces V contained in W is at most $\binom{n+m-1}{m}_p < p^{m(n-1)}(1+2/p)^m$, and we have

$$p^{mn}\left(\frac{\beta}{\alpha}\right) \leq \#\mathcal{V} < p^{m(n-1)}\left(1+\frac{2}{p}\right)^m. \tag{10}$$

Now suppose that for the key pair PK, SK the quantity $\ell_{SK,N,m}$ satisfies (9). Then the number α of distinct nontrivial signatures satisfies

$$\alpha \leq p^m \cdot 2^{-4m/p}\left(\frac{1}{2}\right) < p^m\left(1+\frac{2}{p}\right)^{-m}\left(\frac{1}{2}\right). \tag{11}$$

where the second inequality follows from $2^{2x} > 1+x$ for $x > 0$. Combining inequalities (10) and (11), we see that the fraction β of subspaces with nontrivial signatures satisfies

$$\beta < \frac{\alpha}{p^m} \cdot \left(1+\frac{2}{p}\right)^m < \frac{1}{2}. \tag{12}$$

Now adversary $\mathcal{A}$ works as follows: it chooses at random a vector space $V \subset \mathbb{F}_p^{n+m}$ and obtains id_1 and $\sigma := \mathsf{Sign}(SK, \mathsf{id}_1, V)$ from the signing oracle. The adversary then computes a vector $\mathbf{y} \notin V$ and outputs $(\mathsf{id}_1, \sigma, \mathbf{y})$ as the forgery. By (12) the probability that σ is trivial is at least $1/2$, and if this is the case then $\mathsf{Verify}(PK, \mathsf{id}_1, \mathbf{y}, \sigma) = 1$. Hence, $\mathcal{A}$ has advantage (as in Definition 4) at least $1/2$ while making a single signature query. □

7 Conclusion and Extensions

We studied the problem of signing a subspace $V \subset \mathbb{F}_p^N$ in a manner that authenticates all vectors in V. The question is motivated by the need to provide integrity when using network coding. We defined the problem and described two secure schemes. Our first scheme is homomorphic and has constant-size public keys and per-vector signatures; its security is based on the co-CDH assumption in the random oracle model. Our second scheme offers security based on the weaker discrete logarithm assumption without random oracles. In both schemes, a single public key can be used to sign many linear spaces (i.e., files). We also proved a lower bound on the length of signatures for linear subspaces, and observe that both our systems meet the lower bound.

In real-life network coding applications, one may wish to vary the dimension of the ambient space $\mathbb{F}_p^N$. Our definitions assume that the dimension of the ambient space is fixed. However, our systems can easily be adapted to sign subspaces V_i contained in varying ambient spaces $\mathbb{F}_p^{N_i}$ with a single public key by incorporating the dimension N_i into the hash function (for scheme NCS_1) or the outer signature with respect to $\mathcal{S}_0$ (for scheme NCS_2).

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung. "Network information flow." *IEEE Transactions on Information Theory* **46** (2000), 1204–1216.

[2] M. Bellare, O. Goldreich, and S. Goldwasser. "Incremental cryptography: The case of hashing and signing." In *Advances in Cryptology — Crypto '94*, LNCS **839**. Springer (1994), 216–233.

[3] D. Boneh and M. Franklin. "An efficient public key traitor tracing scheme." In *Adv. in Cryptology — Crypto '99*, LNCS **1666**. Springer (1999), 338–353.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. "Aggregate and verifiably encrypted signatures from bilinear maps." In *Advances in Cryptology — Eurocrypt 2003*, LNCS **2656**. Springer (2003), 416–432.

[5] S. Brands. "An efficient off-line electronic cash system based on the representation problem." (1993). CWI Technical Report CS-R9323.

[6] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. "A digital fountain approach to reliable distribution of bulk data." In *ACM SIGCOMM* (1998).

[7] D. Charles, K. Jain, and K. Lauter. "Signatures for network coding." In *40th Annual Conference on Information Sciences and Systems (CISS '06)* (2006).

[8] P. A. Chou, Y. Wu, and K. Jain. "Practical network coding." In *41st Allerton Conference on Communication, Control, and Computing* (2003).

[9] S. Duquesne and G. Frey. "Background on pairings." In *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC Press (2006).

[10] D. Freeman, M. Scott, and E. Teske. "A taxonomy of pairing-friendly elliptic curves." Cryptology ePrint Archive, Report 2006/372 (2006). Available at `http://eprint.iacr.org/`.

[11] C. Gkantsidis and P. Rodriguez. "Network coding for large scale content distribution." In *IEEE INFOCOM* (2005).

[12] C. Gkantsidis and P. Rodriguez. "Cooperative security for network coding file distribution." In *IEEE INFOCOM* (2006).

[13] K. Han, T. Ho, R. Koetter, M. Médard, and F. Zhao. "On network coding for security." In *IEEE MILCOM* (2007).

[14] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros. "The benefits of coding over routing in a randomized setting." In *Proc. International Symposium on Information Theory (ISIT)* (2003).

[15] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. Karger. "Byzantine modification detection in multicast networks using randomized network coding." In *Proc. Intl. Symposium on Information Theory (ISIT)* (2004), 144–152.

[16] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. "A random linear network coding approach to multicast." *IEEE Trans. Inform. Theory* **52** (2006), 4413–4430.

[17] S. Jaggi. *Design and Analysis of Network Codes.* Ph.D. dissertation, California Institute of Technology (2006).

[18] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros. "Resilient network coding in the presence of Byzantine adversaries." *IEEE Trans. on Information Theory* **54** (2008), 2596–2603.

[19] R. Johnson, D. Molnar, D. Song, and D. Wagner. "Homomorphic signature schemes." In *Topics in Cryptology — CT-RSA*, LNCS **2271** (2002), 244–262.

[20] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC Press (2007).

[21] M. Kim, M. Médard, and J. Barros. "Counteracting Byzantine adversaries with network coding: An overhead analysis." (2008). Available at `http://arxiv.org/abs/0806.4451`.

[22] M. Krohn, M. Freedman, and D. Mazieres. "On the-fly verification of rateless erasure codes for efficient content distribution." In *Proc. of IEEE Symposium on Security and Privacy* (2004), 226–240.

[23] S.-Y. R. Li, R. W. Yeung, and N. Cai. "Linear network coding." *IEEE Trans. Info. Theory* **49** (2003), 371–381.

[24] R. Stanley. *Enumerative Combinatorics, vol. 1.* Cambridge University Press (1997).

[25] F. Zhao, T. Kalker, M. Médard, and K. Han. "Signatures for content distribution with network coding." In *Proc. Intl. Symp. Info. Theory (ISIT)* (2007).