Signing a Linear Subspace: Signature Schemes for Network Coding

Dan Boneh[1]*, David Freeman[1]** Jonathan Katz[2]***, and Brent Waters[3]†

[1] Stanford University, {dabo,dfreeman}@cs.stanford.edu
[2] University of Maryland, jkatz@cs.umd.edu
[3] University of Texas at Austin, bwaters@crl.sri.com.

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.

Here, we propose two signature schemes that can be used in conjunction with network coding to prevent malicious modification of data. In particular, our schemes can be viewed as signing linear subspaces in the sense that a signature σ on V authenticates exactly those vectors in V. Our first scheme is *homomorphic* and has better performance, with both public key size and per-packet overhead being *constant*. Our second scheme does not rely on random oracles and uses weaker assumptions.

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

1 Introduction

Network coding [1, 25] 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 (see, e.g., [21] for measurements of the improvement network coding gives even for unicast traffic). It has also been suggested as a means of improving robustness against random network failures since, as with erasure codes [7], 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 [24], and for improving the performance of large-scale data dissemination over the Internet [12].

A major concern in systems that use network coding is to provide protection against malicious modification of packets (i.e., "pollution attacks") by Byzantine nodes in the network; see [14, 23] for

* Supported by DARPA IAMANET, NSF, and the Packard Foundation.
** Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.
*** Supported by NSF CAREER award #0447075, NSF Trusted Computing grant #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.

two recent surveys and Section 2.2 for a more complete 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 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 in the network, 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; as long as this is the case, however, our schemes ensure that the destination can filter out any corrupted packets and hence 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. We defer a more detailed discussion of our schemes, and their advantages relative to prior work, to 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. (The application to network coding will become clear after the following section.) In addition to our two schemes, we also 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 do not assume any background in network coding, and so provide a quick overview of the relevant details 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 (and compare them to prior work) 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 [25] (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* of the file. Before transmission, the source node creates the m *augmented vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_m$ given by:

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

i.e., 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 introduces $\Theta(m^2)$ 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, a node computes the packet (vector) $\mathbf{w} = \sum_{j=1}^{\ell} \alpha_{i,j}\mathbf{w}_j$, where each $\alpha_{i,j} \in \mathbb{F}_p$. 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 communication 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 the latter 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 [9, 15, 17].

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 $\mathbf{w}_i^{\mathrm{L}}$ denote the left-most n positions of the vector, and let $\mathbf{w}_i^{\mathrm{R}}$ denote the right-most m positions. The receiver first computes an $m \times m$ matrix G such that

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

(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

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

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; in fact, for security purposes, we will see that these augmented vectors are necessary.

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., [7].) 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. 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 by the receiver are likely to 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 [23] for further discussion). For the purposes of our work, we separate existing techniques into two categories: *information-theoretic* and *computational*.

Information-theoretic approaches. Information-theoretic methods for enabling recovery from malicious faults are possible by introducing *redundancy* into the original packets transmitted by the sender [16, 18, 19]. 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: rather than limit the adversary's computational power, these constructions place 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 can not 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 an arbitrary fraction of malicious nodes, as long as the destination node receives m correctly formed and linearly independent vectors. They 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 blocks of the file, 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 has a public key known to all other nodes in the network. A public-key scheme makes the most sense when the sender is multicasting files to many receivers in the network (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. [24] (see also [13]) 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 $O(km)$ and the public key has size $O(kn)$, 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. [28] propose a scheme where the sender computes some authentication information $\mathbf{x}$ derived from a vector *orthogonal* to the space $V = \mathrm{span}(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\})$; 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 $O(k(n+m))$. 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 in advance, before the authentication information can be computed.

Charles et al. [8] present a *homomorphic signature scheme* [20] based on the aggregate signatures of Boneh et al. [5]. 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 $O(k(m+n))$, 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 can be used to authenticate exactly those vectors in V. Our security definition requires that no adversary given a signature on a vector subspace V is able to forge a valid signature for any vector not in V. (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 [8, 28] — and so the formal definition 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 to be 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; i.e., the source need not be aware of the entire file before transmitting 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. [24] 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 the scheme can be viewed as a more efficient version of the scheme of Zhao et al. [28].

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.)

In contrast to information-theoretic schemes for achieving resilient network coding [18, 19], our schemes are resilient to an arbitrary number of faults (as long as a minimum number of correct packets reach the receiver) and have substantially lower communication overhead. On the other hand, the computational requirements of our schemes are higher, and security is proven only relative to unproven (but standard) cryptographic assumptions.

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 of $\mathbb{F}_p^N$ so that any $\mathbf{y} \in V$ is accepted as valid and any $\mathbf{y} \notin V$ is rejected as invalid. 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 especially the receiver, to distinguish packets associated with different files.

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

$\mathsf{Setup}(1^k, N)$. Input: a security parameter 1^k and an integer N, the length of a vector to be signed. Output: a prime p, a public key PK, and a secret key SK.

$\mathsf{Sign}(SK, \mathsf{id}, V)$. Input: a secret key SK, a file identifier id that is an element of a randomly samplable set $\mathcal{I}$, and an m-dimensional subspace $V \subset \mathbb{F}_p^N$, with $0 < m < N$, described as a set of basis vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$. Output: a signature σ.

$\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. Input: a public key PK, an identifier $\mathsf{id} \in \mathcal{I}$, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ. Output: 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 subspaces $V \subseteq \mathbb{F}_p^N$ and for all $\mathsf{id} \in \mathcal{I}$, if $\sigma := \mathsf{Sign}(SK, \mathsf{id}, V)$ 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 entire vector space V. However, the case of homomorphic signatures is more precisely modeled by a definition in which the Sign algorithm produces signatures $\sigma_1, \dots, \sigma_m$ on the basis vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$, and the collection of these signatures constitutes a signature on V. This is encapsulated in the following definition and the subsequent lemma.

[1] One can think of this identifier as being equivalent to a filename. In both of our systems we require that identifiers be *unpredictable* (they need not be random); this can be achieved easily by concatenating an arbitrary filename with a random string.

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

$\mathsf{Setup}(1^k, N)$. Input: a security parameter 1^k and an integer N, the length of a vector to be signed. Output: a prime p, a public key PK, and a secret key SK.

$\mathsf{Sign}(SK, \mathsf{id}, m, \mathbf{v})$. Input: a secret key SK, a file identifier id that is an element of a randomly samplable set $\mathcal{I}$, an integer $m < N$ indicating the dimension of the space being signed, and a vector $\mathbf{v} \in \mathbb{F}_p^N$. Output: a signature σ.

$\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. Input: a public key PK, a file identifier id, and ℓ pairs consisting of a weight $\beta_i \in \mathbb{F}_p$ and a signature σ_i. Output: a signature σ.
(The intuition is that if the σ_i are signatures on vectors $\mathbf{v}_i$, then σ is a signature on $\sum_{i=1}^{\ell} \beta_i \mathbf{v}_i$.)

$\mathsf{Verify}(PK, \mathsf{id}, m, \mathbf{y}, \sigma)$. Input: A public key PK, an identifier $\mathsf{id} \in \mathcal{I}$, an integer $m < N$ indicating the dimension of the space being signed, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ. Output: 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 $\mathsf{id} \in \mathcal{I}$ and all $\mathbf{y} \in \mathbb{F}_p^N$, if $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, m, \mathbf{y})$ then $\mathsf{Verify}(PK, \mathsf{id}, m, \mathbf{y}, \sigma) = 1$.
- For all $\mathsf{id} \in \mathcal{I}$, any $\ell > 0$, and all sets of triples $\{(\beta_i, \sigma_i, \mathbf{v}_i)\}_{i=1}^{\ell}$, if $\mathsf{Verify}(PK, \mathsf{id}, m, \mathbf{v}_i, \sigma_i) = 1$ for all i, then it must be the case that

$$\mathsf{Verify}\Big(PK,\ \mathsf{id},\ m,\ \textstyle\sum_i \beta_i \mathbf{v}_i,\ \mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\})\Big) = 1.$$

If we wish to sign a vector space $V \subseteq \mathbb{F}_p^N$ of dimension m, we can use any set of basis vectors to describe V. In particular, by row-reducing and changing the order of variables, we may assume without loss of generality that the basis $\{\mathbf{v}_i\}$ consists of augmented vectors, i.e., the last m coordinates of $\mathbf{v}_i$ form a unit vector with a 1 in the ith position. We will call such a basis *properly augmented.* Let $n = N - m$. It follows directly that if $\mathbf{v}_1, \dots, \mathbf{v}_m$ is a properly augmented basis of V, then for any $\mathbf{y} \in \mathbb{F}_p^N$ with $\mathbf{y} = (y_1, \dots, y_n, y_{n+1}, \dots y_{n+m})$ we have

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

We can thus form a signature on any vector $\mathbf{y} \in V$ from the signatures on a properly augmented basis $\{\mathbf{v}_i\}$ by using the last m coordinates of the vector $\mathbf{y}$ as the weights in the $\mathsf{Combine}$ algorithm. This observation allows us to construct a scheme satisfying Definition 1 directly from a scheme satisfying Definition 2, by defining the signature on a vector space V to be the set of signatures on the basis vectors $\mathbf{v}_i$.

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

$\mathsf{Setup}_1(1^k, N) = \mathsf{Setup}_2(1^k, N)$.

$\mathsf{Sign}_1(SK, \mathsf{id}, V) = (\mathsf{Sign}_2(SK, \mathsf{id}, m, \mathbf{v}_1), \dots, \mathsf{Sign}_2(SK, \mathsf{id}, m, \mathbf{v}_m))$*, where* $\mathbf{v}_1, \dots, \mathbf{v}_m$ *is a properly augmented basis of* $V \subset \mathbb{F}_p^N$*. We denote the signature by* $\sigma = (\sigma_1, \dots, \sigma_m)$*.*

$$\mathsf{Verify}_1(PK, \mathsf{id}, \mathbf{y}, \sigma) = \mathsf{Verify}_2\Big(PK,\ \mathsf{id},\ m,\ \mathbf{y},\ \mathsf{Combine}_2\big(PK, \mathsf{id}, \{(y_{N-m+i}, \sigma_i)\}_{i=1}^{m}\big)\Big).$$

Proof. It suffices to demonstrate the correctness condition of Definition 1; this follows from (1) and correctness of the homomorphic scheme. $\square$

Security. We define security of a network coding signature scheme satisfying Definition 1, and we say that a homomorphic scheme satisfying Definition 2 is secure if the network coding signature scheme constructed as in Lemma 3 is secure.

An adversary can break a network coding signature scheme in two ways. First, she can create a valid signature on a previously unseen vector space, allowing her to create entire files that will be accepted as valid; this corresponds to the usual notion of signature forgery. Alternatively, given a signature σ on a vector space V, the adversary can produce a vector $\mathbf{y} \notin V$ such that $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$; this allows her to inject malicious packets into the system which the client will then attempt to use to reconstruct the file. We formalize these notions in the following game, played between a challenger and an adversary.

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 sends p and PK to $\mathcal{A}$.

Queries. Proceeding adaptively, $\mathcal{A}$ specifies vector subspaces $V_i \subset \mathbb{F}_p^N$. The challenger chooses an identifier id_i uniformly at random from the set of identifiers $\mathcal{I}$, and sends id_i and $\sigma_i := \mathsf{Sign}(SK, \mathsf{id}_i, V_i)$ to $\mathcal{A}$.

Output. The adversary $\mathcal{A}$ outputs an identifier id^*, a signature σ^*, and a vector $\mathbf{y}^* \in \mathbb{F}_p^N$.

The adversary *wins* the security game if $\mathsf{Verify}(PK, \mathsf{id}^*, \mathbf{y}^*, \sigma^*) = 1$, and either (1) $\mathsf{id}^* \neq \mathsf{id}_i$ for all i and $\mathbf{y}^* \neq \mathbf{0}$ (a *type 1 forgery*), or (2) $\mathsf{id}^* = \mathsf{id}_i$ for some i and $\mathbf{y}^* \notin V_i$ (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.

Note that the "forgery" in which an adversary $\mathcal{A}$ obtains a signature σ on a vector space V and outputs a valid signature σ' on a vector space $V' \subset V$ does not win the security game. 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 $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi)$ with the following properties:

1. $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups (written multiplicatively) of the same order, 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.

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 $\{0, \dots, |\mathbb{G}| - 1\}$, 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 the Tate pairing. Elliptic curves and abelian varieties with the desired properties are called "pairing-friendly." For further details, see [10].

The proof of security of our first signature scheme (Section 4) relies not on the discrete logarithm problem but on a slightly stronger assumption, namely that the *co-computational Diffie Hellman* (or *co-CDH*) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is infeasible. The co-CDH problem is the problem of computing $g^x \in \mathbb{G}_1$ given $g \in \mathbb{G}_1$ and $h, h^x \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ that solves co-CDH in a pair of groups $(\mathbb{G}_1, \mathbb{G}_2)$ takes as input a generator g of $\mathbb{G}_1$, a generator h of $\mathbb{G}_2$, and an element $z = h^x \in \mathbb{G}_2$, 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 $\mathcal{A}$. We note that if in addition to $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ there is also an efficiently computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ then 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 (Definition 2) with constant-size public key and constant-size per-vector signatures.

Signature Scheme NCS$_1$.

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

1. Generate a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi)$ such that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ have prime order $p > 2^k$. Choose generators $g_1, \dots, g_N \xleftarrow{\text{R}} \mathbb{G}_1 \setminus \{1\}$ and $h \xleftarrow{\text{R}} \mathbb{G}_2 \setminus \{1\}$.
2. Choose $\alpha \xleftarrow{\text{R}} \mathbb{F}_p$, and set $u := h^\alpha$.
3. Let $H : \mathbb{Z} \times \mathbb{Z} \to \mathbb{G}_1$ be a hash function, modeled as a random oracle.
4. Output p, the public key $PK := (\mathcal{G}, H, g_1, \dots, g_N, h, u)$ and the private key $SK := \alpha$.

   Since the elements $g_1, \dots, g_N$ can be generated as the output of an independent hash function H' (also modeled as a random oracle), the public key can be compressed to have constant size.

$\mathsf{Sign}(SK, \mathsf{id}, m, \mathbf{v})$. Given a secret key $SK = \alpha$, an identifier $\mathsf{id} \in \{0,1\}^k$, an integer m indicating the dimension of the space being signed, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm sets $n := N - m$ and outputs the signature

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

$\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. Given a public key PK, a file identifier id, and ℓ pairs consisting of a weight $\beta_i \in \mathbb{F}_p$ and a signature σ_i, this algorithm outputs $\sigma := \prod_{i=1}^{\ell} \sigma_i^{\beta_i}$.

$\mathsf{Verify}(PK, \mathsf{id}, m, \mathbf{y}, \sigma)$. Given a public key $PK = (g_1, \dots, g_N, h, u)$, an identifier id, an integer m indicating the dimension of the space being signed, a signature σ, and a vector $\mathbf{y} \in \mathbb{F}_p^N$, set $n := N - m$ and define

$$\gamma_1(PK, \sigma) \stackrel{\text{def}}{=} e(\sigma, h) \quad \text{and} \quad \gamma_2(PK, \mathsf{id}, m, \mathbf{y}) \stackrel{\text{def}}{=} e\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{y_{n+i}} \prod_{j=1}^{n} g_j^{y_j},\ u\right).$$

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

We now demonstrate the correctness conditions of Definition 2. The fact that signatures output by the Sign algorithm verify correctly follows immediately from the bilinearity of the pairing e and the fact that $u = h^{\alpha}$. As for correctness of the $\mathsf{Combine}$ algorithm, let $\{(\mathbf{v}_k, \sigma_k)\}_{k=1}^{\ell}$ be a set of valid message-signature pairs; i.e., $\gamma_1(PK, \sigma_k) = \gamma_2(PK, \mathsf{id}, m, \mathbf{v}_k)$ for all k. Let $\beta_1, \dots, \beta_\ell \in \mathbb{F}_p$, and let $\mathbf{y} = \sum_{k=1}^{\ell} \beta_k \mathbf{v}_k$. Define

$$\sigma := \mathsf{Combine}(PK, \mathsf{id}, \{(\beta_k, \sigma_k)\}) = \prod_{k=1}^{\ell} \sigma_k^{\beta_k}.$$

We need to show that $\gamma_1(PK, \sigma) = \gamma_2(PK, \mathsf{id}, m, \mathbf{y})$.

By the bilinear property of e, we have

$$\gamma_1(PK, \sigma) = e\left(\prod_{k=1}^{\ell} \sigma_k^{\beta_k},\ h\right) = \prod_{k=1}^{\ell} e(\sigma_k, h)^{\beta_k} = \prod_{k=1}^{\ell} \gamma_1(PK, \sigma_k)^{\beta_k},$$

while

$$\begin{aligned}
\gamma_2(PK, \mathsf{id}, m, \mathbf{y}) &= e\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{y_{n+i}} \prod_{j=1}^{n} g_j^{y_j},\ u\right) \\
&= e\left(\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{\sum_{k=1}^{\ell} \beta_k v_{k,n+i}}\right)\left(\prod_{j=1}^{n} g_j^{\sum_{k=1}^{\ell} \beta_k v_{kj}}\right),\ u\right) \\
&= e\left(\left(\prod_{k=1}^{\ell}\prod_{i=1}^{m} H(\mathsf{id}, i)^{\beta_k v_{k,n+i}}\right)\left(\prod_{k=1}^{\ell}\prod_{j=1}^{n} g_j^{\beta_k v_{kj}}\right),\ u\right) \\
&= \prod_{k=1}^{\ell} e\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{k,n+i}} \prod_{j=1}^{n} g_j^{v_{kj}},\ u\right)^{\beta_k} = \prod_{k=1}^{\ell} \gamma_2(PK, \mathsf{id}, \mathbf{v}_k)^{\beta_k}.
\end{aligned}$$

The fact that $\gamma_1(PK, \sigma) = \gamma_2(PK, \mathsf{id}, m, \mathbf{y})$ now follows from the fact that $\gamma_1(PK, \sigma_k) = \gamma_2(PK, \mathsf{id}, m, \mathbf{v}_k)$ for all k.

Note that if $\mathbf{v}$ is a properly augmented basis vector, so that $(v_{n+1}, \dots, v_{n+m})$ is a unit vector, then the time to sign $\mathbf{v}$ is dominated by the time to compute a single hash into $\mathbb{G}_1$ (taking time similar to a full exponentiation) plus n additional exponentiations in $\mathbb{G}_1$. If elements of $\mathbb{G}_1$ are represented using $\log_2 p$ bits, then a signature on a vector is just $\log_2 p$ bits. Such groups $\mathbb{G}_1$ can be obtained by using pairing-friendly elliptic curves of prime or near-prime order, such as supersingular curves or the families of ordinary curves given by Miyaji, Nakabayashi, and Takano [26], Barreto-Naehrig [2], or Freeman [11].

We now prove the security of signature scheme NCS_1; more precisely, we prove the security of the network coding signature scheme NCS_1' constructed from NCS_1 as in Lemma 3. We show that either type of forgeries can be used to solve the co-CDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. An essential ingredient in the proof is the efficiently computable isomorphism $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$, which exists naturally in all of the commonly used bilinear groups. We assume that all vector spaces V queried to the signing oracle are described by a set of properly augmented basis vectors.

Theorem 6. *Let NCS_1' be the network coding signature scheme NCS_1 constructed from Signature Scheme NCS_1 via the method of Lemma 3. The scheme NCS_1' is secure in the random oracle model assuming that 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 $(\mathbb{G}_1, \mathbb{G}_2)$, such that

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

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

Proof. Let $\mathcal{A}$ be an adversary as in Definition 4. We construct an algorithm $\mathcal{B}$ that takes as input $g \in \mathbb{G}_1$ 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. (Recall that NCS_1' is the network coding signature scheme constructed from NCS_1 as in Lemma 3.)

Setup. When $\mathcal{A}$ requests $\mathsf{Setup}(1^k, N)$, $\mathcal{B}$ does the following:
1. Generate a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi)$ such that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ have prime order $p > 2^k$.
2. Choose random $s_1, t_1, \dots, s_N, t_N \in \mathbb{F}_p$, and set $g_j := g^{s_j} \varphi(h)^{t_j}$ for $j = 1, \dots, N$.
3. Output the public key $PK := (\mathcal{G}, H, g_1, \dots, g_N, h, z)$, where H queries $\mathcal{B}$'s hash simulator.

Hash query. When $\mathcal{A}$ requests value of $H(\mathsf{id}, i)$, algorithm $\mathcal{B}$ does the following:
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 $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{F}_p^N$, 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 $1, \dots, m$ then abort. (The simulation has failed.)
2. Set $n := N - m$ and compute $\varsigma_i := -\sum_{j=1}^{n} s_j v_{ij}$ for $i = 1, \dots, m$. Set $\mathbf{s} := (s_1, \dots, s_n, \varsigma_1, \dots, \varsigma_m)$.
3. Choose $\tau_i \xleftarrow{\text{R}} \mathbb{F}_p$ for $i = 1, \dots, m$. Set $\mathbf{t} := (t_1, \dots, t_n, \tau_1, \dots, \tau_m)$.
4. Set $H(\mathsf{id}, i) := g^{\varsigma_i} \varphi(h)^{\tau_i}$ for $i = 1, \dots, m$.
5. Compute $\sigma_i := \varphi(z)^{\mathbf{v}_i \cdot \mathbf{t}}$.

6. Output id and $\sigma := (\sigma_1, \ldots, \sigma_m)$.

Output. If $\mathcal{B}$ does not abort, then eventually $\mathcal{A}$ outputs a signature σ of length m, an identifier id, and a nonzero vector $\mathbf{y}$.

1. If id is not one of the identifiers chosen on a signature query, run hash queries on $H(\mathsf{id}, i)$ as above for $i = 1, \ldots, m$. Set $\mathbf{s} := (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{t} := (t_1, \ldots, t_n, \tau_1, \ldots, \tau_m)$.
2. If id is one of the identifiers chosen on a signature query, let $\mathbf{s}$ and $\mathbf{t}$ be the vectors produced during that query.
3. Set $n := N - m$ and output $\quad \omega := \left(\dfrac{\prod_{i=1}^{m} \sigma_i^{y_{n+i}}}{\varphi(z)^{\mathbf{t}\cdot\mathbf{y}}} \right)^{1/(\mathbf{s}\cdot\mathbf{y})}$.

We first observe that the responses to all hash queries are 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 hash queries produced by $\mathcal{B}$. Since the secret key corresponding to PK is x, it suffices to show that for each $\mathbf{v}_i$,

$$\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{i,n+i}} \prod_{j=1}^{n} g_j^{v_{i,j}} \right)^{x} = \varphi(z)^{\mathbf{v}_i \cdot \mathbf{t}}, \tag{2}$$

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

$$\left(\prod_{i=1}^{m} \left(g^{\varsigma_i} \varphi(h)^{\tau_i}\right)^{v_{i,n+i}} \prod_{j=1}^{n} \left(g^{s_j} \varphi(h)^{t_j}\right)^{v_{i,j}} \right)^{x} = \left(g^{\mathbf{s}\cdot\mathbf{v}_i} \varphi(h)^{\mathbf{t}\cdot\mathbf{v}_i}\right)^{x}. \tag{3}$$

Now observe that we constructed $\mathbf{s}$ so that $\mathbf{s} \in V^{\perp}$ (i.e., $\mathbf{s} \cdot \mathbf{v}_i = 0$ for all i), so this expression is equal to $\varphi(h)^{x(\mathbf{t}\cdot\mathbf{v}_i)}$. Equation (2) 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 nonzero vector $\mathbf{y}$. Let $\sigma = (\sigma_1, \ldots, \sigma_m)$. If $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$ then

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

By the same reasoning as in (3) 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.

We now consider the two types of forgeries, and show that $\mathbf{s}\cdot\mathbf{y} = 0$ with probability $1/p$ in both cases. If $\mathcal{A}$ outputs a type 1 forgery, then $\mathcal{A}$ did not obtain id as the result of a signing query. In this case the only functions of ς_i for this id that $\mathcal{B}$ sends to $\mathcal{A}$ are the values of $H(\mathsf{id}, i)$ for $i = 1, \ldots, m$, and it is clear from the construction that the values of ς_i are uniform in $\mathbb{F}_p$ and independent of the adversary's view. In addition, by Lemma 7 below the variables $s_1, \ldots, s_N$ are uniform in $\mathbb{F}_p$ and independent of the adversary's view. Since $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{y}$ is nonzero, 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 is the identifier returned on some query V, and $\mathbf{y} \notin V$. By Lemma 7 below the variables $s_1, \ldots, s_N$ are uniform in $\mathbb{F}_p$ and independent of the adversary's view. Since $\{\mathbf{v}_i\}$ is a properly augmented basis, this implies that for any given adversary view, the vector $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ is uniformly random in $V^\perp$.

To complete the proof, we show that a uniform distribution on $\mathbf{s} \in V^\perp$ produces a uniform distribution on $\mathbf{s}\cdot\mathbf{y} \in \mathbb{F}_p$. To see this, first recall that $(V^\perp)^\perp = V$. It follows that if $\mathbf{y} \notin V$ then there is some $\mathbf{w}_1 \in V^\perp$ such that $\mathbf{w}_1 \cdot \mathbf{y} \neq 0$. Extend $\mathbf{w}_1$ to a basis $\mathbf{w}_1, \ldots, \mathbf{w}_n$ of $V^\perp$ and write $\mathbf{s} = \sum_{i=1}^n a_i \mathbf{w}_i$. Since $\mathbf{s}$ is uniformly distributed in $V^\perp$ the coefficients a_i are independent and uniformly distributed in $\mathbb{F}_p$. Since $\mathbf{w}_1 \cdot \mathbf{y} \neq 0$, it follows that $\mathbf{s}\cdot\mathbf{y} = \sum_{i=1}^n a_i(\mathbf{w}_i \cdot \mathbf{y})$ is uniformly distributed in $\mathbb{F}_p$. We conclude that the probability that $\mathbf{s}\cdot\mathbf{y} = 0$ is $1/p$, and hence $\omega = g^x$ as required. $\square$

Lemma 7. *Suppose that adversary $\mathcal{A}$ interacts with the simulator $\mathcal{B}$ as above. Then the variables $s_1, \ldots, s_N$ are uniform in $\mathbb{F}_p$ and independent of the adversary's view.*

Proof. In proving the lemma we can ignore any queries for $H(\mathsf{id}, i)$ 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 adversary view 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 $i = 1, \ldots, m_k$.

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:

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

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 system of equations seen by the adversary is

$$\mathbf{s}^{\text{L}} + \alpha \mathbf{t}^{\text{L}} = \mathbf{c}_1 \qquad \text{(public key)} \tag{4}$$

$$\mathbf{s}^{\text{R}}_k + \alpha \mathbf{t}^{\text{R}}_k = \mathbf{c}_{2,k} \qquad \text{(values of } H(\mathsf{id}, i)\text{)} \tag{5}$$

$$\bar{V}_k \mathbf{t}^{\text{L}} + \mathbf{t}^{\text{R}}_k = \mathbf{c}_{3,k} \qquad \text{(signatures)} \tag{6}$$

$$\bar{V}_k \mathbf{s}^{\text{L}} + \mathbf{s}^{\text{R}}_k = \mathbf{0} \qquad (\mathbf{s} \in V^{\perp}) \tag{7}$$

for some vectors $\mathbf{c}_1 \in \mathbb{F}_p^N$, $\mathbf{c}_{2,k} \in \mathbb{F}_p^{m_k}$, $\mathbf{c}_{3,k} \in \mathbb{F}_p^{m_k}$ known to the adversary (who we assume can compute discrete logs in $\mathbb{G}_1$). We wish to show that the system has a unique solution for any value of $\mathbf{s}^{\text{L}}$.

We now observe that the last equation (7) is redundant: specifically, (7) $= \bar{V}_k(4) + (5) - \alpha(6)$. Since the system has at least one solution by construction, any choice of variables satisfying (4)–(6) must also satisfy (7).

Now suppose $\mathbf{s}^{\text{L}}$ is fixed; then equation (4) determines a unique value for $\mathbf{t}^{\text{L}}$. For each k, equation (6) and this value of $\mathbf{t}^{\text{L}}$ determine a unique value for $\mathbf{t}^{\text{R}}_k$, from which equation (5) determines a unique value of $\mathbf{s}^{\text{R}}_k$. Thus for any value of $\mathbf{s}^{\text{L}}$ there is a unique solution to the system (4)–(7). We conclude that $\mathbf{s}^{\text{L}} = (s_1, \ldots, s_N)$ is uniform in $\mathbb{F}_p^N$ and independent of the adversary's view. $\square$

5 A Secure Network Coding Signature Scheme without Random Oracles

Krohn et al. [24] propose authenticating network coding data using a homomorphic hash function. As in Definition 2, their system produces a signature σ_i on each basis vector of the subspace to be authenticated. However, their system is not secure according to our definition, as there is no mechanism to ensure that basis vectors from different files are not combined at reconstruction time. Our solution is to authenticate all of the hash values, along with the file identifier, using a standard signature scheme, which we denote by $\mathcal{S}_0$. This modification produces a secure network coding signature scheme (Definition 1) but eliminates the homomorphic property.

We now describe the scheme using the framework of Definition 1. Security is based on the discrete logarithm assumption, without random oracles. Let $\mathcal{S}_0 = (\mathsf{Setup}_0, \mathsf{Sign}_0, \mathsf{Verify}_0)$ be a signature system for signing messages in $\{0,1\}^*$.

Signature Scheme NCS$_2$.

$\mathsf{Setup}(1^k, N)$. Input: positive integers 1^k and N.

1. Choose a group $\mathbb{G}$ of prime order $p > 2^k$.
2. Run $\mathsf{Setup}_0(1^k)$ and let the public/private keys be PK_0, SK_0.

3. Choose generators $g_i \xleftarrow{\text{R}} \mathbb{G} \setminus \{1\}$ for $i = 1, \ldots, N$.
4. Output the prime p, the public key $PK := (g_1, \ldots, g_N, PK_0)$, the private key $SK := SK_0$, and descriptions of $\mathbb{G}$ and $\mathcal{S}_0$.

$\mathsf{Sign}(SK, \mathsf{id}, V)$. Input: A secret key SK, a file identifier id, and an m-dimensional subspace $V \subset \mathbb{F}_p^{n+m}$, with $0 < m, n \le N$, described as a set of properly augmented basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_p^N$.

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

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

1. Run $\mathsf{Verify}_0(PK_0, (\mathsf{id}, \sigma_1, \ldots, \sigma_m), \tau)$. If the answer is 0, output 0.
2. Set $n := N - m$. If $\left(\prod_{j=1}^n g_j^{y_j}\right)\left(\prod_{i=1}^m \sigma_i^{y_{n+i}}\right) = 1$ output 1; otherwise output 0.

We now demonstrate the correctness condition of Definition 1. Clearly the τ produced by the algorithm satisfies

$$\mathsf{Verify}_0(PK_0, (\mathsf{id}, \sigma_1, \ldots, \sigma_m), \tau) = 1.$$

If the σ_i are computed correctly, then Step (2) of the Verify algorithm computes

$$\left(\prod_{j=1}^n g_j^{y_j}\right)\left(\prod_{i=1}^m \prod_{j=1}^n g_j^{-v_{ij}}\right)^{y_{n+i}} = \prod_{j=1}^n g_j^{y_j - \sum_{i=1}^m y_{n+i} v_{ij}}.$$

If $\mathbf{y} \in V$ then since $\{\mathbf{v}_i\}$ is a properly augmented basis we have $\mathbf{y} = \sum_{i=1}^m y_{n+i}\mathbf{v}_i$, so the exponent of each factor on the right-hand side is zero, and thus the entire product is the identity in $\mathbb{G}$.

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. We remark that if one is willing to use the random oracle model, we may consider signature scheme NCS_2 to have a constant-size public key, as the values $g_1, \ldots, g_N$ may be computed as the output of a hash function H (viewed as a random oracle).

Theorem 8. *The network coding signature scheme* NCS_2 *is secure assuming hardness of the discrete logarithm problem in* $\mathbb{G}$*, and assuming* $\mathcal{S}_0$ *is a secure signature scheme.*

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*

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

where $\text{Sig-Adv}[\mathcal{B}_1, \mathcal{S}_0]$ *is the probability that* $\mathcal{B}_1$ *wins the security game for the standard signature scheme* $\mathcal{S}_0$ *(see [22, §12.2]).*

Proof. Suppose algorithm $\mathcal{A}$ as in Definition 4 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 an $\mathcal{S}_0$ signature τ for the message $(\mathsf{id}, \sigma_1, \ldots, \sigma_m)$; the construction of the adversary $\mathcal{B}_1$ is standard.

Now suppose id is an identifier returned in response to a signature query on the vector space V, and that $\mathbf{y} \notin V$ (type 2 forgery). Let H be the "hash function" $H(\mathbf{v}) = \prod_{j=1}^{n} g_j^{v_j}$. 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 [6, 3], 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 $\varepsilon/2$. For further details, see [24, §VI]. $\square$

Relation with [28]. Signature Scheme NCS_2 can also be viewed as a secure instantiation of the signature scheme proposed by Zhao et al. [28]. Given a vector space V described as a set of basis vectors $\mathbf{v}_i$, the Zhao et al. scheme computes a secret vector $\mathbf{u} \in V^{\perp}$ and publishes $(h_1, \ldots, h_N) = (g^{u_1}, \ldots, g^{u_N})$, where g is a generator of a group $\mathbb{G}$ in which the discrete logarithm problem is infeasible, along with a signature on this tuple. Verification of $\mathbf{y}$ involves checking whether $\prod_{j=1}^{N} h_j^{y_j} = g^{\mathbf{u}\cdot\mathbf{y}}$ is the identity in $\mathbb{G}$. 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 [28, Theorem 1] or [4, Lemma 3.2]).

To view the scheme NCS_2 from this perspective, we pick a generator g of $\mathbb{G}$ and write the first $n = N - m$ elements of the public key of Scheme NCS_2 as $g^{u_1}, \ldots, g^{u_n}$. If we let $u_{n+i} = \sum_{j=1}^{n} -u_j v_{ij}$, then the signature element σ_i is equal to $g^{u_{n+i}}$, and if $\{\mathbf{v}_i\}_{i=1}^{m}$ is a properly augmented basis 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 Zhao et al. scheme.

The signatures produced by scheme NCS_2 have length $O(m)$ and are thus much shorter than those signatures of Zhao et al., which have length $O(N)$. In addition, the incorporation of the file identifier into the $\mathcal{S}_0$ signature τ allows us to sign multiple messages with a single public key, which the original Zhao et al. scheme was unable to do.

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 satisfy the following two properties:

– **Additive**: 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 of our constructions above are additive.

– **Fixed size**: For a given $m > 0$ and a given SK, the size in bits of the signature $\mathsf{Sign}(SK, \mathsf{id}, V)$ 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 satisfy these two properties, we show that the signature size must be about $m \log_2 p$ bits or more. In particular, we construct an attack algorithm that forges signatures for any

SK that generates signatures shorter than our 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 N, m be integers with $0 < m \leq N$. Let $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ be a network coding signature scheme satisfying the two properties above. There is an adversary $\mathcal{A}$ with running time polynomial in 1^k with the following property: let $(p, SK, PK) \xleftarrow{\mathrm{R}} \mathsf{Setup}(1^k, N)$ and suppose the quantity $\ell_{SK,N,m}$ satisfies*

$$\ell_{SK,N,m} \leq m \log_2 p - (4m/p) - 1. \tag{8}$$

Then $\mathcal{A}$ makes a single signature query and wins the security game of Definition 4 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 [27, 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

$$p^{mn}\left(\frac{\beta}{\alpha}\right) \leq \#\mathcal{V} < p^{m(n-1)}\left(1+\frac{2}{p}\right)^m. \tag{9}$$

Now suppose that for the key pair PK, SK the quantity $\ell_{SK,N,m}$ satisfies (8). Then the number α of distinct nontrivial signatures satisfies

$$\alpha \leq p^m \cdot 2^{-4m/p} \left(\frac{1}{2}\right) < p^m \left(1 + \frac{2}{p}\right)^{-m} \left(\frac{1}{2}\right). \tag{10}$$

where the second inequality follows from $2^{2x} > 1 + x$ for $x > 0$. Combining inequalities (9) and (10), we see that the fraction β of subspaces with nontrivial signatures satisfies

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

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 (11) 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. $\square$

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 for integrity in networks using network coding. We defined the problem and described two secure systems. The first system is based on the CDH assumption in the random oracle model, and is more functionally useful for network coding applications. The second system is based on the weaker discrete-log assumption without random oracles. In both of our systems, a single public key can be used to sign many linear spaces. We also proved a lower bound on the length of such signatures and observed that both of our systems meet the lower bound.

In real-life network coding applications, one may wish to vary the dimension of the ambient space $\mathbb{F}_p^N$ as well as the dimension of the subspaces V_i being signed. Our definitions above assume that the dimension of the ambient space is fixed while the dimension of the subspaces V_i may vary. 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 $\mathcal{S}_0$ signature (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. P. Barreto and M. Naehrig. "Pairing-friendly elliptic curves of prime order." In *Selected Areas in Cryptography — SAC 2005*, Springer LNCS **3897** (2006), 319–331.
3. M. Bellare, O. Goldreich, and S. Goldwasser. "Incremental cryptography: The case of hashing and signing." In *Advances in Cryptology — CRYPTO 1994*, Springer LNCS **839** (1994), 216–233.
4. D. Boneh and M. Franklin. "An efficient public key traitor tracing scheme." In *Advances in Cryptology — CRYPTO 1999*, Springer LNCS **1666** (1999), 338–353.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. "Aggregate and verifiably encrypted signatures from bilinear maps." In *Advances in Cryptology — EUROCRYPT 2003*, Springer LNCS **2656** (2003), 416–432.

6. S. Brands. "An efficient off-line electronic cash system based on the representation problem." (1993). CWI Technical Report CS-R9323.
7. J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. "A digital fountain approach to reliable distribution of bulk data." *SIGCOMM Comput. Commun. Rev.* **28** (1998), 56–67.
8. D. Charles, K. Jain, and K. Lauter. "Signatures for network coding." In *40th Annual Conference on Information Sciences and Systems (CISS '06)* (2006). Available at `http://eprint.iacr.org/2006/025`. To appear in *International Journal of Information and Coding Theory.*
9. P. A. Chou, Y. Wu, and K. Jain. "Practical network coding." In *41st Allerton Conference on Communication, Control, and Computing* (2003).
10. S. Duquesne and G. Frey. "Background on pairings." In *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Chapman & Hall/CRC, Boca Raton, FL (2006), 115–124.
11. D. Freeman. "Constructing pairing-friendly elliptic curves with embedding degree 10." In *Algorithmic Number Theory — ANTS-VII*, Springer LNCS **4076** (2006), 452–465.
12. C. Gkantsidis and P. Rodriguez. "Network coding for large scale content distribution." In *Proc. of IEEE INFOCOM 2005* (2005), 2235–2245.
13. C. Gkantsidis and P. Rodriguez. "Network coding for large scale content distribution." In *Proc. of IEEE INFOCOM 2005* (2005).
14. K. Han, T. Ho, R. Koetter, M. Médard, and F. Zhao. "On network coding for security." In *IEEE MILCOM* (2007).
15. T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros. "The benefits of coding over routing in a randomized setting." In *Proc. of International Symposium on Information Theory (ISIT)* (2003).
16. 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. of International Symposium on Information Theory (ISIT)* (2004), 144–152.
17. 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.
18. S. Jaggi. *Design and Analysis of Network Codes.* Ph.D. dissertation, California Institute of Technology (2006).
19. 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.
20. R. Johnson, D. Molnar, D. Song, and D. Wagner. "Homomorphic signature schemes." In *Topics in cryptology — CT-RSA 2002*, Springer LNCS **2271** (2002), 244–262.
21. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. "XORs in the air: practical wireless network coding." *IEEE/ACM Trans. Netw.* **16** (2008), 497–510.
22. J. Katz and Y. Lindell. *Introduction to modern cryptography.* Chapman & Hall/CRC, Boca Raton, FL (2008).
23. 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`.
24. 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.
25. S.-Y. R. Li, R. W. Yeung, and N. Cai. "Linear network coding." *IEEE Trans. Inform. Theory* **49** (2003), 371–381.
26. A. Miyaji, M. Nakabayashi, and S. Takano. "Characterization of elliptic curve traces under FR-reduction." In *Information Security and Cryptology — ICISC 2000*, Springer LNCS **2015** (2001), 90–108.
27. R. P. Stanley. *Enumerative combinatorics. Vol. 1*, Cambridge Studies in Advanced Mathematics **49**. Cambridge University Press, Cambridge (1997).
28. F. Zhao, T. Kalker, M. Médard, and K. Han. "Signatures for content distribution with network coding." In *Proc. of International Symposium on Information Theory (ISIT)* (2007).