Compact Signatures for Network Coding

Jonathan Katz* Brent Waters†

Abstract

Network coding offers increased throughput and improved robustness to random faults in completely decentralized networks. Since it does not require centralized control, network coding has been suggested for routing packets in ad-hoc networks, for content distribution in P2P file systems, and for improving the efficiency of large-scale data dissemination over the Internet.

In contrast to traditional routing schemes, however, network coding requires intermediate nodes to process and modify data packets *en route*. For this reason, standard signature schemes are inapplicable and it is therefore a challenge to provide resilience to tampering by malicious nodes in the network. Here, we propose a novel *homomorphic* signature scheme that can be used in conjunction with network coding to prevent malicious modification of data. The overhead of our scheme is small and independent of the file or packet size: both public keys and signatures in our scheme consist of only a single group element.

1 Introduction

Network coding [1, 21] refers to a general class of routing mechanisms where, in contrast to more traditional "store-and-forward" routing, intermediate nodes may be required to process and *modify* the data packets in transit. Network coding has been shown to offer a number of advantages with respect to traditional routing, most well-known of which is the possibility of increased throughput in certain network topologies (see, e.g., [18] for measurements of the improvement network coding gives in practice even for unicast traffic). It has also been suggested as a means of improving robustness to random network failures since, as with erasure codes [5], 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 its 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 [20], and for improving the performance of large-scale data dissemination over the Internet [10].

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 [11, 19] for 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

*Dept. of Computer Science, University of Maryland. jkatz@cs.umd.edu. Research 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.

†Dept. of Computer Science, University of Texas at Austin. bwaters@crl.sri.com. 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.

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.)

In this work, we construct a novel homomorphic signature scheme 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/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 scheme ensures 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). The primary improvement of our scheme relative to prior work [7, 24] is that public keys and signatures in our scheme are *constant size* whereas in previous scheme these had size roughly the square root of the file size. Our scheme has other advantages as well; we defer a more detailed discussion to Section 2.3.

Work whose aim is to ensure the *secrecy* of data sent using network coding [6, 9] is orthogonal to our work, which is focused only on data *integrity*.

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 scheme (and compare it to prior work) in Section 2.3. In the remainder of the paper we introduce an appropriate definition of security, propose our new signature scheme, and prove our scheme secure.

2 Background

2.1 Linear Network Coding

In a *linear* network coding scheme [21] (the only type with which we will be concerned in this work), a file to be transmitted is viewed as an ordered sequence of vectors $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m \in \mathbb{F}_p^n$; 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:

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

i.e., each original vector $\bar{\mathbf{v}}_i$ is pre-pended by 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.

Each node in the network processes packets as follows. Upon receiving packets (i.e., vectors) $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathbb{F}_p^{m+n}$ on its k incoming communication edges, a node computes the packet (vector) $\mathbf{w} = \sum_{j=1}^{k} \alpha_{i,j} \cdot \mathbf{w}_j$, 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. In a fault-free execution of the scheme, then, 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 who knows the network topology. More usefully (and more interestingly), however, these values can also be chosen randomly and independently by each node in a completely decentralized fashion, and without any knowledge of the network topology. (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 [8, 12, 13].

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

$$I = G \cdot \begin{pmatrix} \mathbf{w}_1^L \\ \vdots \\ \mathbf{w}_m^L \end{pmatrix},$$

where I denotes the $m \times m$ identity matrix. The original file $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ is then given by

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

We stress that the receiver need not be aware of the weights $\{\alpha_{i,j}\}$ used by any 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., [5].) 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. A first thought is for the source to simply sign each packet that it releases into the network. This approach fails completely, as the packets received by the receiver are, in general, different from those issued by the sender. (The only guarantee is that correctly-formed packets should be a *linear combination* of the vectors packets transmitted by the sender.) Thus, this approach is simply incompatible with network coding.

A second naive idea is for the source to sign the entire file; the resulting signature could either be appended to the file itself and transmitted using network coding, or sent separately. Although this would ensure that the receiver never accepts an incorrect file, the problem with this approach is that there is no computationally-efficient way for the receiver to recover the correct file in the first

place. To see this, suppose the receiver has the sender's signature on the file and receives t packets, out of which $m' \geq m$ are correctly formed. 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 roughly $\binom{t}{m}$ sets of m linearly-independent vectors before it will recover the correct file. We see from this example that an approach whereby the receiver can detect (and discard) corrupt packets immediately is advantageous.

Having rejected the above simplistic approaches, we survey other techniques for combatting data pollution when network coding is used (see [19] for further discussion regarding existing approaches). 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 [14, 15, 16]; in essence, these techniques compose an "inner" error-correcting code with the "outer" network coding scheme, and the destination node reconstructs the file by applying error correction to a set of sufficiently-many received packets. Such techniques do not rely on any computational assumptions, but can inherently offer security only against a relatively limited class of adversaries. Specifically, if we let C denote the maximum achievable flow in the network (in the absence of adversarial behavior) and assume the adversary can eavesdrop on the entire network, then the (optimal) scheme of [16] tolerates at most $C/2$ corrupted links, and correspondingly even fewer corrupted nodes. Under the assumption that the adversary can eavesdrop on at most z_e links, the (optimal) scheme of [16] tolerates at most $\min\{C, \frac{1}{2} \cdot (C - z_e)\}$ corrupted links. More importantly, the communication overhead introduced by these schemes is significant, thus potentially mitigating the advantages of network coding in the first place.

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 is not achieved in our setting by having the source sign each packet, 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 even 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. Finally, 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.

Homomorphic hashing [20, 10] is one suggestion for preventing pollution attacks. In the scheme of Krohn et al. [20], the sender computes a hash $h_i = H(\bar{\mathbf{v}}_i)$ of each block of the file; given $h_1, \ldots, h_m$, anyone can check whether a network-coded packet $\mathbf{w}$ is a correctly-formed linear combination of the augmented vectors $\{\mathbf{v}_i\}$. (We stress that all m hash values are needed in order to verify authenticity of a packet, in general.) Barring a reliable channel from the sender to the destination, the hash values $\{h_i\}$ must be signed by the sender (using a standard signature scheme) and transmitted in

the network. If all m hash values are sent along with each vector — and not modified en route — then this introduces an overhead of $O(m)$ group elements per packet. On the other hand, if the hash values are partitioned among multiple packets then intermediate nodes cannot verify authenticity of the packets they receive and the destination cannot verify authenticity of packets until it has received all m hash values. The sender's public key in their scheme contains $O(n)$ group elements; thus, either the public key or the "authentication information" $\{h_i\}$ have size at least the square root of the file size. We remark that Krohn et al. also suggest a recursive version of their basic scheme, but the recursive scheme does not overcome the above problems.

Most related to our work, two previous papers [7, 24] design signature schemes that are specifically suited to network coding. Charles et al. [7] present a *homomorphic signature scheme* [17] with the property that valid signatures $\sigma_1, \ldots, \sigma_k$ on vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ can be combined, without knowledge of the signer's secret key, to produce a valid signature σ on any linear combination $\sum_i \alpha_i \mathbf{w}_i$. Their scheme can only be used to sign a *single* file, after which the public key must be refreshed; this clearly limits applicability of their solution. Public keys in their scheme have size $O(k \cdot (m+n))$, where k is a cryptographic security parameter, meaning that it will be impractical to re-distribute public keys over the network even if network coding is used for public-key distribution. We remark that Charles et al. do not give formal definitions or proofs of security in their paper.

Zhao et al. [24] also present a scheme for making network coding resilient to the injection of false packets. Their approach, roughly speaking, is to authenticate the vector space $V = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$; this results in authentication information $\mathbf{x}$ that can be used to verify that $\mathbf{w} \in V$ for any received packet $\mathbf{w}$. (The authentication information $\mathbf{x}$ must be signed by the sender using a standard signature scheme.) The most significant drawback of their work is that both the authentication information $\mathbf{x}$ and public keys in their scheme have size $O(k \cdot (m+n))$, which is at least the square root of the file size. Their scheme, too, is not immediately suited for distributing more than one file using the same public key; though they suggest some approaches for handling multiple files, they do not prove security of any of these suggestions. An additional drawback of their signature scheme is that it requires the sender to know the entire file in advance, before the authentication information can be computed. This limits the applicability of their scheme for transmission of *streaming* data, where the sender transmits packets as they are generated rather than buffering them and transmitting them all at once. (This problem remains, though to a lesser extent, even if the sender transmits a file in several *generations*. In that case, the sender must wait until all the data for a given generation is known before it can compute the authentication information for that generation.)

To summarize: all prior approaches for achieving cryptographic security introduce a communication overhead of at least the square root of the file size, if authentication of multiple files using a single public key is desired. Here, we will show a homomorphic signature scheme with *constant-size* signatures and public keys.

2.3 Our Contribution

We propose a new homomorphic signature scheme that can be used to authenticate packets for network coding, and prove its security in the random oracle model based on the computational Diffie-Hellman assumption in bilinear groups. Security is proved relative to a definition, introduced here, that guarantees exactly the security desired in our setting: roughly speaking, given a collection of m signatures corresponding to the m vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ defining a file, an adversary can generate a valid signature for any vector in $V \stackrel{\text{def}}{=} \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ but is unable to forge a valid signature on any vector not in V. (Actually, both our security definition and our construction

directly take into account the distribution of multiple files using a single public key — in contrast to [7, 24] — and so the formal definition is a bit more involved.) Our 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.

The primary advantage of our work as compared to prior work is that both public keys and signatures in our scheme have *constant* size; in fact, our signatures consist of only a single group element while public keys also contain only a single group element in addition to some parameters that can be shared among multiple users. This is a significant improvement of prior work in which the public key and/or authentication information had size at least the square root of the file size. From a computational point of view, our scheme is only slightly more expensive than prior work. Specifically, signature verification in our scheme requires $m + n$ exponentiations and one pairing computation; this is the same as the scheme of [7], whereas the scheme of Zhao et al. [24] requires $m+n$ exponentiations plus verification of a standard signature. (The homomorphic hashing approach of [20] requires m exponentiations plus verification of a standard signature, but has various drawbacks as discussed in the previous section.)

In contrast to information-theoretic schemes for achieving resilient network coding [15, 16], our scheme is resilient to an arbitrary number of faults (as long as a minimum number of correct packets reach the receiver) and has substantially lower communication overhead. On the other hand, the computational requirements of our scheme are higher, and security is proven only relative to unproven (but standard) cryptographic assumptions.

3 Definitions and Preliminaries

3.1 Cryptographic Preliminaries

The security of our scheme is based on the (standard) computational Diffie-Hellman assumption but in a group where a bilinear map is defined (and so the *decisional* Diffie-Hellman problem is easy); we review the relevant definitions here. (As in [4], our construction extends naturally to the case of a bilinear map defined over two different groups; we use the current formulation for simplicity of exposition only.) Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p and such that there exists an efficiently-computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Here, bilinearity means that for all $g, h \in \mathbb{G}_1$ and all $\alpha, \beta \in \mathbb{Z}_p$ we have $\hat{e}(g^\alpha, h^\beta) = \hat{e}(g, h)^{\alpha\beta}$.

A *BDH parameter generator* $\mathcal{IG}$ is a randomized, polynomial-time algorithm that takes as input a security parameter 1^k and outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and a map $\hat{e}$ satisfying the above conditions (we assume p, the group order, is implicit in $\mathbb{G}_1, \mathbb{G}_2$). The *computational Diffie-Hellman problem with respect to* $\mathcal{IG}$ is the following: given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ output by $\mathcal{IG}$ along with random $g, g^\alpha, h \in \mathbb{G}_1$, compute h^α. We say that $\mathcal{IG}$ *satisfies the computational Diffie-Hellman assumption* if the following probability is negligible (in k) for all PPT algorithms A:

$$\Pr\left[\begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); \\ g, h \leftarrow \mathbb{G}_1; \alpha \leftarrow \mathbb{Z}_p\end{array} : A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g^\alpha, h) = h^\alpha\right].$$

BDH parameter generators believed to satisfy the above assumption can be constructed from modified Weil or Tate pairings associated with elliptic curves or Abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [2, 4] for details.

3.2 Definitions Specific to Our Setting

Rather than appeal to the general definition of homomorphic[1] signature schemes [17], we introduce definitions of functionality and security specific to our setting. We refer to the primitive under study as a *signature scheme for linear subspaces*. As discussed previously, we want our scheme to be useful for the distribution of multiple files using the same public key. As such, we will modify the network coding scheme as described in the Introduction in the following, simple way: 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.[2] 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.) This just provides a way for honest nodes, and especially the receiver, to distinguish packets associated with different files.

Definition 1 A *signature scheme for linear subspaces* is defined by a tuple of probabilistic, polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Vrfy})$ such that:

- Gen takes as input the security parameter 1^k in unary, and outputs a pair of public and private keys (pk, sk). The public key defines a prime p and integers m and n.
- Sign takes as input the secret key sk, an identifier $\mathsf{id} \in \{0,1\}^*$, and a vector $\mathbf{v} \in \mathbb{F}_p^{m+n}$. It outputs a signature σ. We write this as $\sigma \leftarrow \mathsf{Sign}_{sk}(\mathsf{id}, \mathbf{v})$.
- $\mathsf{Combine}$ takes as input the public key pk, an identifier id, a set of weights $\beta_1, \ldots, \beta_\ell \in \mathbb{F}_p$, and a sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \mathbb{F}_p^{m+n}$ along with their signatures $\sigma_1, \ldots, \sigma_\ell$. It outputs a signature σ.
- Vrfy takes as input the public key pk, an identifier id, a vector $\mathbf{v} \in \mathbb{F}_p^{m+n}$, and a signature σ. It outputs a boolean value indicating acceptance or rejection. We write this as $\mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}, \sigma)$.

Furthermore, the following correctness conditions must hold for all (pk, sk) output by Gen:

- For all $\mathsf{id} \in \{0,1\}^*$, it holds that $\mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}, \mathsf{Sign}_{sk}(\mathsf{id}, \mathbf{v})) = 1$.
- For any $\mathsf{id} \in \{0,1\}^*$, any $\ell > 0$, any weights $\beta_1, \ldots, \beta_\ell \in \mathbb{F}_p$, and any collection of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ and signatures $\sigma_1, \ldots, \sigma_\ell$, if it holds that

$$\forall i: \quad \mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}_i, \sigma_i) = 1\,,$$

  then it must be the case that

$$\mathsf{Vrfy}_{pk}\left(\mathsf{id},\ \textstyle\sum_i \beta_i \mathbf{v}_i,\ \mathsf{Combine}_{pk}\left(\mathsf{id}, \{\beta_i\}, \{\mathbf{v}_i\}, \{\sigma_i\}\right)\right) = 1.$$

In the absence of malicious behavior, a signature scheme of the above type would be used as follows: Given a file, viewed as a sequence of m vectors $\bar{v}_1, \ldots, \bar{\mathbf{v}}_m \in \mathbb{F}_p^n$, the sender would prepare the augmented vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_p^{m+n}$ as described in the Introduction. The sender will then choose an identifier id and compute $\sigma_i \leftarrow \mathsf{Sign}_{sk}(\mathsf{id}, \mathbf{v}_i)$. Finally, it will inject packets

[1] Johnson et al. [17] introduce the general notion of *homomorphic signatures*, but their work does *not* give signature schemes satisfying Definition 1.

[2] One can think of this identifier as being equivalent to a filename. For our proof of security, we require that identifiers be *unpredictable* (they need not be random); this can be achieved easily by concatenating an arbitrary filename with a short random string.

$(\mathsf{id}, \mathbf{v}_1, \sigma_1), \ldots, (\mathsf{id}, \mathbf{v}_m, \sigma_m)$ into the network. Observe that the entire file need not be known before the signer generates a signature on the first vector in the file; the signer only needs to keep track of the file identifier until the entire file is processed.

An intermediate node in the network receiving packets $(\mathsf{id}_1, \mathbf{v}_1, \sigma_1), \ldots, (\mathsf{id}_\ell, \mathbf{v}_\ell, \sigma_\ell)$ would, ideally, first verify the integrity of each of these packets by computing $\mathsf{Vrfy}_{pk}(\mathsf{id}_i, \mathbf{v}_i, \sigma_i)$ and discarding any packets which failed this step. The node would group the remaining packets by their identifers, and compute a linear combination of vectors associated with the same identifier as described in the Introduction. Along with this, it would also apply the $\mathsf{Combine}$ algorithm in order to produce valid signatures for each resulting vector. It then forwards the resulting vectors, along with their appropriate identifiers and signatures, to the next hop.

The destination node recovers a file in the obvious way: it first collects all packets with a given identifier, and discards any packets whose signatures do not verify correctly. It then reconstructs the file exactly as described in the Introduction. We remark that the receiver can begin decoding the file as soon as it begins to receive packets. (For example, using the same notation as in the Introduction, as soon as the receiver obtains correctly-generated packets $\mathbf{w}_1, \ldots, \mathbf{w}_k$ for which $(1, 0, \ldots, 0)$ is in the span of $\mathbf{w}_1^L, \ldots, \mathbf{w}_k^L$, the receiver can recover the first block of the file.)

Note: The definition above assumes that all files contain the same number of blocks m. This is, of course, without loss of generality since a shorter file can always be padded. Moreover, our construction easily generalizes to support shorter files, without introducing any overhead, by incorporating the file length into the identifier.

We now present our definition of security. For notational convenience, we identify a file with a vector (sub)space $V \subset \mathbb{F}_p^{m+n}$. (In practice, V will just be the vector space spanned by the augmented vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ constituting the file. In our definition of security, however, we do not force the adversary to use m-dimensional subspaces but instead allow the adversary to specify a subspace of arbitrary dimension.) The adversary may request signatures for a series of files; in response to the ith such request, the signer chooses a random[3] identifier $\mathsf{id}_i \in \{0,1\}^k$, generates signatures on the blocks of the given file with respect to that identifier, and gives the resulting signatures to the adversary. This defines a sequence $(\mathsf{id}_1, V_1), \ldots, (\mathsf{id}_\ell, V_\ell)$ of identifiers and their corresponding subspaces/files. The adversary succeeds if it is able to output an identifier id^*, a vector $\mathbf{v}^*$, and a signature σ^* such that $\mathsf{Vrfy}_{pk}(\mathsf{id}^*, \mathbf{v}^*, \sigma^*) = 1$ and $\mathbf{v}^*$ is not a vector lying in the appropriate subspace (see the definition for details). A scheme is secure if every polynomial-time adversary succeeds with only negligible probability.

Definition 2 A signature scheme for linear subspaces is *existentially unforgeable under adaptive chosen-message attacks* (or simply *secure*) if for every probabilistic, polynomial-time adversary $\mathcal{F}$, the probability that $\mathcal{F}$ succeeds in the following experiment is negligible in the security parameter k:

1. $\mathsf{Gen}(1^k)$ is run to obtain (pk, sk), and the public key pk is given to $\mathcal{F}$.

2. $\mathcal{F}$ may then adaptively request signatures on files of its choice. Recalling that we identify files with vector subspaces of $\mathbb{F}_p^{m+n}$, each such query is handled as follows:

   - $\mathcal{F}$ specifies a vector space V by giving a basis $\mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{F}_p^{m+n}$, where $1 \leq t \leq m+n$. (For application to network coding, the signer would never sign a "vector space" of dimension greater than m but we allow it in our definition.)

[3]Recall that we only need identifiers to be *unpredictable*. Thus, even if the adversary has some control over the filename, we can achieve this condition by having the signer append a random k-bit string to the filename.

- A random $\mathsf{id} \in \{0,1\}^k$ is chosen, and then $\mathcal{F}$ is given the t signatures $\mathsf{Sign}_{sk}(\mathsf{id}^*, \mathbf{v}_1)$, ..., $\mathsf{Sign}_{sk}(\mathsf{id}^*, \mathbf{v}_t)$.

If $\mathcal{F}$ makes a total of ℓ queries, this defines a sequence $(\mathsf{id}_1, V_1), \ldots, (\mathsf{id}_\ell, V_\ell)$ of ℓ identifiers and their associated subspaces.

3. Finally, $\mathcal{F}$ outputs $(\mathsf{id}^*, \mathbf{v}^*, \sigma^*)$. It succeeds if $\mathsf{Vrfy}_{pk}(\mathsf{id}^*, \mathbf{v}^*, \sigma^*) = 1$ and either (1) $\mathsf{id}^* \notin \{\mathsf{id}_1, \ldots, \mathsf{id}_\ell\}$ or (2) for some i it holds that $\mathsf{id}^* = \mathsf{id}_i$, but $\mathbf{v}^* \notin V_i$.

4 Our Construction

Our construction is somewhat similar to both the BLS (standard) signature scheme [4] as well as the BGLS aggregate signature scheme [3]. The scheme of Charles et al. [7] is also adapted, in a different way, from the BGLS scheme; as noted earlier, however, the scheme of Charles et al. only allows for the distribution of one file per public key, and the public key in their scheme has size $O(m+n)$ (and does not seem to be compressible even in the random oracle model). Our scheme is also similar, but not identical, to a signature scheme used recently by Shacham and Waters [22] in a different context.

Our scheme is defined as follows:

- $\mathsf{Gen}(1^k)$ runs $\mathcal{IG}(1^k)$ to generate parameters $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ have prime order p. It then chooses $g, h \leftarrow \mathbb{G}_1$ and $\alpha \leftarrow \mathbb{Z}_p$. The public key is (g, g^α) and the secret key is α. We also assume a hash function $H : \{0,1\}^* \to \mathbb{G}_1$ that will be modeled as a random oracle in our security proof. (Note that $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, H)$ could be shared among multiple signers.)
- $\mathsf{Sign}_{sk}(\mathsf{id}, \mathbf{v})$, where $\mathbf{v} = (v_1, \ldots, v_{m+n}) \in \mathbb{F}_p^{m+n}$, outputs the signature
$$\sigma = \left(\prod_{i=1}^{m+n} H(\mathsf{id} \| i)^{v_i} \right)^\alpha ,$$
where "$\|$" denotes concatenation.
- $\mathsf{Combine}$, given as inputs an identifier id, a set of weights $\beta_1, \ldots, \beta_\ell \in \mathbb{F}_p$, and a sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \mathbb{F}_p^{m+n}$ along with their signatures $\sigma_1, \ldots, \sigma_\ell$, outputs the signature $\sigma = \prod_{i=1}^{\ell} \sigma_i^{\beta_i}$ on the vector $\mathbf{v} = \sum_{i=1}^{\ell} \beta_i \cdot \mathbf{v}_i$.
- $\mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}, \sigma)$, where $pk = (g, y)$ and $\mathbf{v} = (v_1, \ldots, v_{m+n}) \in \mathbb{F}_p^{m+n}$, outputs 1 iff
$$\hat{e}(g, \sigma) \stackrel{?}{=} \hat{e}\left(y, \prod_{i=1}^{m+n} H(\mathsf{id} \| i)^{v_i} \right).$$

Correctness. Consider a signature σ output by $\mathsf{Sign}_{sk}(\mathsf{id}, \mathbf{v})$. We have

$$\begin{aligned} \hat{e}(g, \sigma) &= \hat{e}\left(g, \left(\prod_{i=1}^{m+n} H(\mathsf{id} \| i)^{v_i} \right)^\alpha \right) \\ &= \hat{e}\left(g^\alpha, \prod_{i=1}^{m+n} H(\mathsf{id} \| i)^{v_i} \right) \\ &= \hat{e}\left(y, \prod_{i=1}^{m+n} H(\mathsf{id} \| i)^{v_i} \right), \end{aligned}$$

and so $\mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}, \sigma) = 1$. As for the $\mathsf{Combine}$ algorithm, say $\sigma \leftarrow \mathsf{Combine}(\mathsf{id}, \{\beta_i\}, \{\mathbf{v}_i\}, \{\sigma_i\})$ and $\mathsf{Vrfy}_{pk}(\mathsf{id}, \mathbf{v}_i, \sigma_i) = 1$ for all i. Then

$$\begin{aligned}\hat{e}(g,\sigma) &= \hat{e}\left(g, \prod_{i=1}^{\ell} \sigma_i^{\beta_i}\right) = \prod_{i=1}^{\ell} \hat{e}(g,\sigma_i)^{\beta_i} \\ &= \prod_{i=1}^{\ell} \hat{e}\left(y, \prod_{j=1}^{m+n} H(\mathsf{id}\|j)^{v_{ij}}\right)^{\beta_i},\end{aligned}$$

where v_{ij} denotes the jth component of vector $\mathbf{v}_i$. Continuing, we have

$$\begin{aligned}\hat{e}(g,\sigma) &= \prod_{i=1}^{\ell} \hat{e}\left(y, \prod_{j=1}^{m+n} H(\mathsf{id}\|j)^{v_{ij}}\right)^{\beta_i} \\ &= \prod_{i=1}^{\ell} \hat{e}\left(y, \prod_{j=1}^{m+n} H(\mathsf{id}\|j)^{\beta_i \cdot v_{ij}}\right) \\ &= \prod_{j=1}^{m+n} \hat{e}\left(y, \prod_{i=1}^{\ell} H(\mathsf{id}\|j)^{\beta_i \cdot v_{ij}}\right) \\ &= \prod_{j=1}^{m+n} \hat{e}\left(y, H(\mathsf{id}\|j)^{\sum_{i=1}^{\ell} \beta_i \cdot v_{ij}}\right) \\ &= \prod_{j=1}^{m+n} \hat{e}\left(y, H(\mathsf{id}\|j)^{v_j^*}\right),\end{aligned}$$

where $(v_1^*, \ldots, v_{m+n}^*) = \mathbf{v}^* \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} \beta_i \mathbf{v}_i$. This proves correctness of the scheme.

4.1 Proof of Security

We now prove security of the scheme introduced in the previous section.

Theorem 1 *If $\mathcal{IG}$ satisfies the computational Diffie-Hellman assumption, then the scheme of the previous section is a secure signature scheme for linear subspaces.*

Proof Let $\mathcal{F}$ be an adversary attacking the scheme in the sense of Definition 2, and let $\varepsilon_{\mathcal{F}}(k)$ denote the success probability of this adversary. We define an algorithm A that will solve the computational Diffie-Hellman problem with probability related to the success probability of $\mathcal{F}$. Algorithm A is given $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, y, h$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ have prime order p and $y = g^{\alpha}$ for some $\alpha \in \mathbb{F}_p$ unknown to A. Algorithm A then simulates an instance of the signature scheme for $\mathcal{F}$. It sets the public key $pk = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, y)$ and then runs $\mathcal{F}(pk)$. It answers the oracle queries of $\mathcal{F}$ as follows:

Random oracle queries. If the value of $H(\mathsf{id}\|i)$ has already been defined at some point earlier in the experiment, then A simply returns that value. Otherwise, A chooses random $z, w \leftarrow \mathbb{F}_p$ and returns $g^z \cdot h^w \in \mathbb{G}_1$.

Signing queries. If $\mathcal{F}$ requests a signature on the vector subspace V, given by a basis $\mathbf{v}_1, \ldots, \mathbf{v}_t$, then A proceeds as follows:

1. Choose random $\mathsf{id} \leftarrow \{0,1\}^k$. If the same value id was used to answer a previous signing query, then abort.

2. Choose $\mathbf{w} \leftarrow V^\perp$; that is, let $\mathbf{w}$ be a random vector that is orthogonal to V. (If $V = \mathbb{F}_p^{m+n}$ then $\mathbf{w}$ is the 0-vector.) Let $\mathbf{w} = (w_1, \ldots, w_{m+n})$.

3. Also choose $\mathbf{z} = (z_1, \ldots, z_{m+n}) \leftarrow \mathbb{F}_p^{m+n}$.

4. For $i = 1, \ldots, m+n$, set $H(\mathsf{id}\|i) = g^{z_i} \cdot h^{w_i}$. If the value of H on any of these points has already been defined at some point earlier in the experiment, then abort.

5. For $i = 1, \ldots, t$, set $\sigma_i = y^{\langle \mathbf{v}_i, \mathbf{z} \rangle}$, where $\langle \cdot, \cdot \rangle$ denotes the standard inner product over $\mathbb{F}_p$ (i.e., $\langle \mathbf{v}_i, \mathbf{z} \rangle = \sum_{j=1}^{m+n} v_{ij} \cdot z_j \bmod p$). Return $\sigma_1, \ldots, \sigma_t$ to $\mathcal{F}$.

As in Definition 2, let $(\mathsf{id}_1, V_1), \ldots, (\mathsf{id}_\ell, V_\ell)$ denote the sequence of identifiers and subspaces defined by the signing queries of $\mathcal{F}$.

At some point, $\mathcal{F}$ outputs $(\mathsf{id}^*, \mathbf{v}^*, \sigma^*)$. Assume without loss of generality that, before outputting such a tuple, $\mathcal{F}$ queries $H(\mathsf{id}^*\|i)$ for $i = 1, \ldots, m+n$. Because of the way hash queries are simulated (whether in response to a hash query or in the course of answering a signing query), for all i it holds that $H(\mathsf{id}^*\|i) = g^{z_i^*} h^{w_i^*}$ for some values z_i^*, w_i^* known to A. Let $\mathbf{z}^* \stackrel{\text{def}}{=} (z_1^*, \ldots, z_{m+n}^*)$ and $\mathbf{w}^* \stackrel{\text{def}}{=} (w_1^*, \ldots, w_{m+n}^*)$, and set

$$Z \stackrel{\text{def}}{=} \langle \mathbf{z}^*, \mathbf{v}^* \rangle, \quad W \stackrel{\text{def}}{=} \langle \mathbf{w}^*, \mathbf{v}^* \rangle. \tag{1}$$

If $W = 0$ then A outputs fail. Otherwise, A outputs $\left(\sigma^*/y^Z\right)^{1/W}$.

We claim that A outputs the correct solution to its given input instance with probability that is negligibly close to $\varepsilon_{\mathcal{F}}(k)$. To see this, first observe that A aborts (when answering signing queries) with only negligible probability since id is chosen uniformly at random from $\{0,1\}^k$ when answering each such query. Assuming A does not abort, the responses to all random oracle queries are uniformly and independently distributed as required. The signatures returned by A also have the correct form since, if $H(\mathsf{id}\|i) = g^{z_i} \cdot h^{w_i}$ for $i = 1, \ldots, m+n$, then, for any $\mathbf{v}_j = (v_{j1}, \ldots)$, a valid signature on $\mathbf{v}_j$ would be computed as

$$\begin{aligned} \left(\prod_{i=1}^{m+n} H(\mathsf{id}\|i)^{v_{ji}} \right)^\alpha &= \left(\prod_{i=1}^{m+n} (g^{z_i} \cdot h^{w_i})^{v_{ji}} \right)^\alpha \\ &= \left(g^{\sum_{i=1}^{m+n} v_{ji} \cdot z_i} \cdot h^{\sum_{i=1}^{m+n} v_{ji} \cdot w_i} \right)^\alpha . \end{aligned}$$

If $\mathbf{v}_j$ and $\mathbf{w}$ are orthogonal, as they are by construction when A answers signing queries, then it holds that $\sum_{i=1}^{m+n} v_{ji} \cdot w_i = 0$ and so

$$\begin{aligned} \left(g^{\sum_{i=1}^{m+n} v_{ji} \cdot z_i} \cdot h^{\sum_{i=1}^{m+n} v_{ji} \cdot w_i} \right)^\alpha &= \left(g^{\langle \mathbf{v}_j, \mathbf{z} \rangle} \right)^\alpha \\ &= y^{\langle \mathbf{v}_j, \mathbf{z} \rangle} , \end{aligned}$$

exactly as returned by A. We conclude that $\mathcal{F}$, when run by A as above, succeeds with probability negligible close to $\varepsilon(k)$.

We next claim that, conditioned on the fact that $\mathcal{F}$ succeeds, A outputs a correct answer with all but negligible probability. First observe that when $\mathcal{F}$ succeeds and $W \neq 0$ (where W is as in

Equation (1)), then A outputs the correct answer. To see this, let $H(\mathsf{id}^* \| i) = g^{z_i^*} h^{w_i^*}$ as above, and note that when $\mathcal{F}$ succeeds then

$$\hat{e}(g, \sigma^*) = \hat{e}\left(y, \prod_{i=1}^{m+n} \left(g^{z_i^*} h^{w_i^*}\right)^{v_i^*}\right)$$

and so

$$\begin{aligned} \sigma^* &= \left(\prod_{i=1}^{m+n} \left(g^{z_i^*} h^{w_i^*}\right)^{v_i^*}\right)^{\alpha} \\ &= \left(g^{\langle \mathbf{z}^*, \mathbf{v}^* \rangle} \cdot h^{\langle \mathbf{w}^*, \mathbf{v}^* \rangle}\right)^{\alpha} = y^Z \cdot (h^{\alpha})^W. \end{aligned}$$

Thus, $h^{\alpha} = (\sigma^*/y^Z)^{1/W}$ as computed by A.

It remains to show that $W = 0$ with only negligible probability. We separately consider the two cases in which $\mathcal{F}$ succeeds.

Case 1: $\mathsf{id}^* \notin \{\mathsf{id}_1, \ldots, \mathsf{id}_\ell\}$. Observe that, for each i, we have $H(\mathsf{id}^* \| i) = g^{z_i^*} \cdot h^{w_i^*}$ for values z_i^*, w_i^* known to A. From the viewpoint of $\mathcal{F}$, the vector $\mathbf{w}^* = (w_1^*, \ldots, w_{m+n}^*)$ is uniformly distributed in $\mathbb{F}_p^{m+n}$. So the probability that $W \stackrel{\text{def}}{=} \langle \mathbf{w}^*, \mathbf{v}^* \rangle = 0$ is $1/p$, which is negligible.

Case 2: $\mathsf{id}^* = \mathsf{id}_i$ but $\mathbf{v}^* \notin V_i$. (Note that, by construction of A, the $\{\mathsf{id}_i\}$ are distinct or else A aborts. Furthermore, if $\mathbb{F}$ succeeds then it must be the case that $V_i \neq \mathbb{F}_p^{m+n}$.) As above, for each value of i let $H(\mathsf{id}^* \| i) = g^{z_i^*} \cdot h^{w_i^*}$ for some values z_i^*, w_i^* known to A. From the viewpoint of $\mathbb{F}$, the vector $\mathbf{w}^* = (w_1^*, \ldots, w_{m+n}^*)$ is uniformly distributed in $V_i^{\perp}$. Since $\mathbf{v}^* \notin V_i$, the probability that $W \stackrel{\text{def}}{=} \langle \mathbf{w}^*, \mathbf{v}^* \rangle = 0$ is again $1/p$, which is negligible. This concludes the proof. ■

4.2 Efficiency Improvements

Using standard techniques, batching can be used to speed up verification when the fraction of corrupted packets is expected to be low. We omit the details.

5 Conclusion

We show in this paper a homomorphic signature scheme in which signatures on a set of vectors $\{\mathbf{v}_i\}$ can be used to generate a valid signature on any vector $\mathbf{v} \in \text{span}(\{\mathbf{v}_i\})$. This signature scheme can be naturally used in conjunction with network coding to ensure protection against malicious modification of data in transit, in the presence of an arbitrary number of corrupted nodes. In contrast to previous suggestions offering such protection [7, 24], our construction adds only a (small) constant overhead to the communication complexity and is therefore suitable for environments where network coding is used.

References

[1] R. Ahlswede, N. Cai, S.-Y.R. Li, W. Yeung. Network Information Flow. *IEEE Trans. Information Theory* 46(4): 1204–1216, 2000.

[2] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Computing* 32(3): 586–615, 2003.

[3] D. Boneh, E.-J. Goh, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *Advances in Cryptology — Eurocrypt 2003.*

[4] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *J. Cryptology* 17(4): 297–320.

[5] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. *ACM SIGCOMM*, 1998.

[6] N. Cai and W. Yeung. Secure Network Coding. *IEEE Intl. Symposium on Information Theory (ISIT)*, 2002.

[7] D. Charles, K. Jain, and K. Lauter. Signatures for Network Coding. *40th Annual Conference on Information Sciences and Systems*, 2006. Available at `http://eprint.iacr.org`.

[8] P. A. Chou, Y. Wu, and K. Jain. Practical Network Coding. *41st Allerton Conference on Communication, Control, and Computing*, 2003.

[9] J. Feldman, T. Malkin, R. Servedio, and C. Stein. On the Capacity of Secure Network Coding. *42nd Annual Allerton Conference on Communication, Control, and Computing,* 2004.

[10] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. *IEEE Infocom*, 2005.

[11] K. Han, T. Ho, R. Koetter, M. Médard, and F. Zhao. On Network Coding for Security. *IEEE MILCOM*, 2007.

[12] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. *IEEE International Symposium on Information Theory (ISIT)*, 2003.

[13] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effors, J. Shi, and B. Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Information Theory* 52(10): 4413–4430, 2006.

[14] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. Karger. Byzantine Modification Detection in Multicast Networks using Randomized Network Coding. *IEEE International Symposium on Information Theory (ISIT)*, 2004.

[15] S. Jaggi. Design and Analysis of Network Codes. PhD Thesis, California Institute of Technology, 2006.

[16] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard. Resilient Network Coding in the Presence of Byzantine Adversaries. *IEEE Infocom*, 2007.

[17] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic Signature Schemes. *RSA Conference — Cryptographers' Track*, 2002.

[18] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcraft. XORs in the Air: Practical Wireless Network Coding. *ACM SIGCOMM*, 2006.

[19] M. Kim, M. Médard, and J. Barros. Counteracting Byzantine Adversaries with Network Coding: An Overhead Analysis. Available at `http://arxiv.org/abs/0806.4451`

[20] M. Krohn, M.J. Freedman, and D. Maziéres. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. *IEEE Symposium on Security and Privacy*, 2004.

[21] S.-Y.R. Li, W. Yeung, and N. Cai. Linear Network Coding. *IEEE Trans. Information Theory* 49(2): 371–381, 2003.

[22] H. Shacham and B. Waters. Compact Proofs of Retrievability. Available at `http://eprint.iacr.org/2008/073`.

[23] R.W. Yeung and N. Cai. Network Error Correction, Part I: Basic Concepts and Upper Bounds. *Communications in Information and Systems* 6(1): 19–36, 2006.

[24] F. Zhao, T. Kalker, M. Médard, and K.J. Han. Signatures for Content Distribution with Network Coding. *IEEE Intl. Symposium on Information Theory (ISIT)*, 2007.