Efficient Network Coding Signatures in the Standard Model

Dario Catalano[1], Dario Fiore[2,⋆], and Bogdan Warinschi[3]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
catalano@dmi.unict.it
[2] Department of Computer Science, New York University, USA
fiore@cs.nyu.edu
[3] Dept. Computer Science, University of Bristol, UK
bogdan@cs.bris.ac.uk

Abstract. Network Coding is a routing technique where each node may actively modify the received packets before transmitting them. While this departure from passive networks improves throughput and resilience to packet loss it renders transmission susceptible to *pollution attacks* where nodes can misbehave and change in a malicious way the messages transmitted. Nodes cannot use standard signature schemes to authenticate the modified packets: this would require knowledge of the original sender's signing key. Network coding signature schemes offer a cryptographic solution to this problem. Very roughly, such signatures allow signing vector spaces (or rather bases of such spaces), and these signatures are homomorphic: given signatures on a set of vectors it is possible to create signatures for any linear combination of these vectors. Designing such schemes is a difficult task, and the few existent constructions either rely on random oracles or are rather inefficient. In this paper we introduce two new network coding signature schemes. Both of our schemes are provably secure in the standard model, rely on standard assumptions, *and* are in the same efficiency class as previous solutions based on random oracles.

1 Introduction

Network Coding [1,23] is an elegant and novel routing approach that is alternative to traditional routing where each node simply stores and forwards the incoming packets. The main difference is that in Network Coding intermediate nodes can modify data packets in transit, still allowing the final recipients to obtain the original information.

More specifically, we consider a network setting where a *source* node wants to transmit a piece of information (a file) to a set of *target* nodes. The source node splits the file into m network packets and sends them to its neighboring nodes. An intermediate node who receives a set of packets from its incoming links, modifies them and sends the resulting packets into the network through its outgoing edges. In *Linear Network Coding* packets are seen as vectors in a

⋆ Work done while at École Normale Supérieure.

M. Fischlin, J. Buchmann, and M. Manulis (Eds.): PKC 2012, LNCS 7293, pp. 680–696, 2012.

linear space over some field and the modifications by the intermediate nodes are linear combinations of these vectors. Such linear combinations can be performed by using ad-hoc coefficients (e.g., fixed by the application or defined by a central authority), or random coefficients chosen by the intermediate nodes in a suitable domain. The latter case is referred to as *Random (Linear) Network Coding*. In addition to offering a more decentralized approach, random network coding has been shown to perform almost as well as network coding with ad-hoc coefficients [12,16,18]. One important aspect of linear network coding is that it enables target nodes to recover the original information with high probability if they receive sufficiently many correct packets. Interestingly, the target nodes can do so without knowledge of the coefficients chosen by the intermediate nodes. We give a more detailed description of these techniques in Section 2.2.

The original motivation for network coding was to increase throughput in decentralized networks and indeed, the technique performs well in wireless/ad-hoc network topologies where a centralized control may not be available. For example, it has been suggested as a good means to improve file sharing in peer-to-peer networks [22], and digital content distribution over the Internet [15].

The main issue of (random) linear network coding is its susceptibility to *pollution attacks* in which malicious nodes (or simple network error transmission) may inject into the network invalid packets to prevent the target nodes from reconstructing the original information. In the specific setting of linear network coding, an invalid packet is simply a vector outside the space spanned by the initial m vectors sent by the source node. In turn, intermediary nodes can later use the invalid incoming vectors thus generating even more invalid packets. This means that errors may dramatically propagate through the network, and adversaries might easily mount a Denial of Service attack to prevent the file from being reconstructed by only injecting *a few* invalid packets.

Two main approaches have been proposed to deal with this problem. One is information-theoretic and uses error-correction techniques [17,18,20]. Unfortunately, this introduces redundant information that badly affects the communication efficiency. The other approach (the one considered in our work) relies on computational assumptions and uses cryptographic techniques. Here, the main idea is to provide a way to authenticate valid vectors. However, standard authentication techniques, such as MACs or digital signatures, do not solve the problem as we want to grant the intermediate nodes some malleability on the received vectors.

The main tool that has been proposed to achieve this goal employs *network coding signature* schemes [7]. In a few words, a network coding signature allows to sign a linear subspace $\mathcal{W} \subset \mathbb{F}^N$ in such a way that a signature σ on $\mathcal{W}$ is verified only by those vectors $w \in \mathcal{W}$.

These schemes can be constructed either from *homomorphic hash functions*, or from *homomorphic signatures*. Very briefly, a homomorphic hash function H satisfies the property that for any vectors a, b and scalar coefficients α and β, it holds that $H(\alpha a + \beta b) = H(a)^{\alpha} H(b)^{\beta}$. Constructions based on homomorphic hashing [22,16,7,14] are less recent and their security can be based on well-established

assumptions in the standard model, such as solving discrete log or factoring. The main drawback of this approach is that the public key and the authentication information that has to be sent along with the packets are linear in the size m of the vector space and thus defeats the purpose of increasing the throughput. Furthermore, the sender has to know the entire file before sending the first packet (which is undesirable for example in the ubiquitous streaming applications).

In contrast, solutions based on homomorphic signatures [7,14,3,11] are more communication-efficient, even though they are computationally somewhat more expensive than those built from homomorphic hashing. In a nutshell, a homomorphic signature is a special type of signature scheme that enjoys a linear homomorphic property: for any vectors a, b and scalar coefficients α and β, it holds that $\mathsf{Sign}(\alpha a + \beta b) = \mathsf{Sign}(a)^{\alpha}\mathsf{Sign}(b)^{\beta}$. More formally, this means that the scheme is equipped with a $\mathsf{Combine}$ algorithm that given μ signatures $\sigma_1, \ldots, \sigma_\mu$ on vectors $w_1, \ldots, w_\mu$ respectively, and scalar coefficients $\alpha_1, \ldots, \alpha_\mu$, it can compute a signature σ which is valid with respect to the vector $w = \sum_{i=1}^{\mu} \alpha_i \cdot w_i$. Importantly, the combination operation does not require the secret key. The security notion for this primitive requires that an adversary who receives signatures on a set of vectors $w_1, \ldots, w_m$ should be able to generate only signatures on vectors that lie in the linear span of $(w_1, \ldots, w_m)$. It should be clear at this point how this primitive can be used to secure the network coding-based application (see Section 2.4 for a detailed description) and, more generally, enable authenticated computation of linear functions of signed data [2].

Related Work. Since our work focuses on homomorphic network coding signatures, in this section we describe the most significant works in this topic. The notion of homomorphic signature was first introduced by Johnson, Molnar, Song and Wagner in a more general setting [21] and only recently adapted to the particular application for network coding by Boneh, Freeman, Katz and Waters [7]. In their work, Boneh *et al.* propose an efficient construction over bilinear groups and prove its security from the CDH assumption in the random oracle model. One year later, Gennaro, Katz, Krawkzyk and Rabin [14] proposed another implementation of homomorphic network coding signatures based on RSA in the random oracle model. Moreover, as an additional contribution, they showed that even if the homomorphic signature works over a large finite field (or over the integers), it is possible to use small coefficients in the linear combinations, and this significantly improves the efficiency at the intermediate nodes in the network coding application. In [9] Boneh and Freeman give the construction of a homomorphic network coding signature based on lattices. As a new property, their scheme allows to authenticate vectors defined over binary fields, and is based on the problem of finding short vectors in integer lattices. The security of this construction relies on the random oracle heuristic. In addition, the same paper shows a scheme in the standard model, but this scheme is only k-time secure (a signing key can be used to issue only k signatures, where k is fixed in advance). In a subsequent work [8], Boneh and Freeman proposed the notion of homomorphic signatures for polynomial functions. While all previous works considered schemes whose homomorphic property allows to compute only linear

functions on the signed data, the scheme in [8] is capable to evaluate multivariate polynomials. Their construction uses ideal lattices and its security is proven in the random oracle model.

The problems associated to the use of the random oracles are well-known and significant research effort is invested in devising implementations that do not rely on this heuristic. For network coding such constructions proved elusive – and we are only aware of two such proposals [3,11][1].

In [3] Attrapadung and Libert give an implementation over bilinear groups of composite order, using the dual system techniques of Waters [24] to carry on the security proof. Unfortunately the scheme relies on the setting of composite order groups and is thus highly inefficient. Furthermore, even if the scheme were to be converted to gropus of prime order (as suggested, but not fully described in [3]), the efficiency gap between the resulting construction and those in the random oracle solutions is still significant.

The most recent proposal is by Catalano, Fiore and Warinschi who propose a homomorphic network coding signature as an application of the notion of Adaptive Pseudo-Free groups [11]. In particular, the concrete implementation is secure in the standard model under the Strong RSA assumption. While from the point of view of computation the efficiency of this scheme is not far from that of the random oracle construction of Gennaro *et al.* which also works in the RSA group, the signature's size in [11] is much worse than that in [14], as it is very affected by the large random exponent s (that is 1346 bits long if one considers 80 bits of security).

Our Contribution. In this work we design two new homomorphic network coding signatures with security proofs in the standard model. Our realizations outperform in efficiency the two currently known constructions in the standard model [3,11] and achieve computational and communication efficiency comparable to those of the random oracle implementations [7,14].

Our first scheme works over asymmetric bilinear groups of prime order p, and is secure under the q-Strong Diffie Hellman assumption (q-SDH for short) introduced by Boneh and Boyen [6]. The construction adapts ideas from the signature by Hofheinz and Kiltz [19] which in turn is based on the concept of Programmable Hash Functions. There, a signature is a random $r \in \mathbb{Z}_p$ and a group element X that is a solution of $X^{z+r} = H(M)$, where z is the secret key, and H is the programmable hash function. To obtain a solution for signing vector spaces along the same lines, we developed some non-trivial extensions which roughly speaking deal with the fact that in our case the same random exponent has to be reused for several signatures. In our construction, a signature on a vector $w = (u, v) \in \mathbb{F}_p^{m+n}$ consists of a random element $s \in \mathbb{Z}_p$ and the

[1] We mention that the random oracle based solution given in [7] might be turned into a scheme secure in the standard model if one is willing to give up the homomorphic property. This makes the resulting solution much less interesting in practice as the signer would need to sign all the vectors in the given subspace at once.

solution X to the following equation:

$$X^{z+\mathsf{fid}} = h^s h_1^{u_1} \cdots h_m^{u_m} g_1^{v_1} \cdots g_n^{v_n}$$

where $\mathsf{fid} \in \mathbb{Z}_p$ represents the random file identifier and z is the secret key. We can therefore achieve rather short signatures: one group element plus an element of $\mathbb{Z}_p$, that is, about 512 bits for 128 bits of security.

Our second realization works over $\mathbb{Z}_N^*$ where N is the product of two safe primes pq. The scheme can be seen as an optimization of the construction by Catalano-Fiore-Warinschi where the random exponent s can now be taken as small as $2k$ bits (where k denotes the desired bit security). The signature on a vector $w = (u, v) \in \mathbb{F}^{m+n}$ is a random integer $s \in \mathbb{Z}_e$ and the solution x to the equation

$$x^e = g^s h_1^{u_1} \cdots h_m^{u_m} g_1^{v_1} \cdots g_n^{v_n} \bmod N$$

where e is a random prime representing the file identifier, and $g, h_1, \ldots, h_m$, $g_1, \ldots, g_n \in \mathbb{Z}_N^*$ are in the public key. As an additional improvement, we show how to do linear combinations $\pmod e$, allowing for the signature scheme to be used in networks with paths of any lengths. This was not the case in [11] and [14] where the parameters have to be set according to a bound L on the maximum length of a path between the source and the target nodes in the network.

A more detailed efficiency analysis of our schemes as well as comparisons with previous solutions, are given in Section 5.

Concurrent Work. In concurrent and independent work Freeman has proposed a semi-generic transformation for building linearly-homomorphic signatures from standard signature schemes [13]. This transformation yields new linearly homomorphic signature schemes that are secure in the standard model under a new security notion (introduced in [13]) which is slightly stronger than the one considered in our work. Our schemes are different from the ones obtained in [13] enjoy better efficiency. It is of future interest to check whether they also satisfy the stronger notion of security proposed in [13].

2 Background and Definitions

In what follows we will denote with $k \in \mathbb{N}$ a security parameter. We say that a function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is negligible if and only if for every positive polynomial $p(k)$ there exists a $k_0 \in \mathbb{N}$ such that for all $k > k_0$: $\epsilon(k) < 1/p(k)$. If S is a set, we denote with $x \xleftarrow{\$} S$ the process of selecting x uniformly at random in S. Let $\mathcal{A}$ be a probabilistic algorithm. We denote with $x \xleftarrow{\$} \mathcal{A}(\cdot)$ the process of running $\mathcal{A}$ on some appropriate input and assigning its output to x.

2.1 Computational Assumptions

An integer N is called *RSA modulus* if it is the product of two distinct prime numbers pq. The Strong RSA Assumption was introduced by Baric and Pfitzmann in [4]. Informally, the assumption states that given a public RSA modulus

N, and a random value $z \in \mathbb{Z}_N$, any PPT adversary cannot compute an e-th root of z for an $e \neq 1$ of its choice.

Definition 1 (Strong RSA Assumption). *Let N be a random RSA modulus of length k where $k \in \mathbb{N}$ is the security parameter, and z be a random element in $\mathbb{Z}_N$. Then we say that the Strong RSA assumption holds if for any PPT adversary $\mathcal{A}$ the probability*

$$Pr[(y, e) \leftarrow \mathcal{A}(N, z) : y^e = z \bmod N \wedge e \neq 1]$$

is negligible in k.

Let $\mathbb{G}, \mathbb{G}'$ and $\mathbb{G}_T$ be bilinear groups of prime order p such that $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_T$ is a bilinear map. The q-Strong Diffie-Hellman Assumption (q-SDH for short) was introduced by Boneh and Boyen in [5] and it is defined as follows.

Definition 2 (q-SDH Assumption). *Let $k \in \mathbb{N}$ be the security parameter, $p > 2^k$ be a prime, and $\mathbb{G}, \mathbb{G}', \mathbb{G}_T$ be bilinear groups of the same order p such that g and g' are the generators of $\mathbb{G}$ and $\mathbb{G}'$ respectively. Then we say that the q-SDH Assumption holds in $\mathbb{G}, \mathbb{G}', \mathbb{G}_T$ if for any PPT algorithm $\mathcal{A}$ and any $q = poly(k)$, the following probability (taken over the random choice of x and the random coins of $\mathcal{A}$) is negligible in k*

$$\Pr[\mathcal{A}(g, g^x, g^{x^2}, \cdots, g^{x^q}, g', (g')^x) = (c, g^{1/(x+c)})]$$

2.2 Background on Linear Network Coding

In linear network coding [1,23] a file to be transmitted is viewed as a set of n-dimensional vectors $(v^{(1)}, \ldots, v^{(m)})$ defined over the integers or over some finite field. To transmit a file $\mathcal{V} = (v^{(1)}, \ldots, v^{(m)})$ the source node creates m augmented vectors $(w^{(1)}, \ldots, w^{(m)})$ where each $w^{(i)}$ is obtained by prepending to $v^{(i)}$ a vector $u^{(i)}$ of length m, i.e., $w^{(i)} = (u^{(i)}, v^{(i)})$. Precisely, $(u^{(1)}, \ldots, u^{(m)})$ represents the canonical basis of $\mathbb{Z}^m$, that is $u^{(i)}$ is the i-th unitary vector, with 1 in position i and 0 elsewhere. This way, the vectors $w^{(1)}, \ldots, w^{(m)}$ form a basis of a subspace $\mathcal{W} \subset \mathbb{F}^{m+n}$. Vectors $w^{(i)}$ of the above form are called *properly augmented vectors* while $(w^{(1)}, \ldots, w^{(m)})$ is a *properly augmented basis*.

In this setting, the *source* node sends these vectors as packets in the network. Whenever a node in the network receives $(w^{(1)}, \ldots, w^{(\mu)})$ on its μ incoming edges, it computes a linear combination $\hat{w}$ of the received vectors and transmits $\hat{w}$ in the network through its outgoing edges. The coefficients used in the linear combination can be fixed by the application, established by a central authority, or they can be randomly chosen by each node. The latter is the case considered in our work and it is called "random network coding". As shown in [12,16,18], random network coding performs almost as well as linear network coding with ad-hoc coefficients. To recover the original file a node must receive m (valid) vectors $\hat{w}^{(1)}, \ldots, \hat{w}^{(m)}$ of the form described before, i.e., $\hat{w}^{(i)} = (\hat{u}^{(i)}, \hat{v}^{(i)})$. In particular, in order for the file to be reconstructed, the vectors $(\hat{u}^{(1)}, \ldots, \hat{u}^{(m)})$

need to be linearly independent. Let denote with $\hat{U}$ the matrix whose rows are the vectors $(\hat{u}^{(1)}, \ldots, \hat{u}^{(m)})$ and with $\hat{\mathcal{V}}$ the matrix whose rows are the vectors $(\hat{v}^{(1)}, \ldots, \hat{v}^{(m)})$. Then, the original file can be retrieved by computing

$$\mathcal{V} = \hat{U}^{-1} \cdot \hat{\mathcal{V}}.$$

Although the above described approach solves the problem of recovering the information in network coding, as we mentioned in the introduction, the main issue in this approach is that it is susceptible to *pollution attacks* where malicious nodes may inject invalid packets in the network so that the reconstruction of the original file becomes impossible. This is particularly sensitive also because a single error introduced by a (malicious) node can be propagated by honest nodes.

Before describing solutions, we observe how two trivial approaches do not solve the problem. First, the source node cannot simply sign the transmitted packets as the receivers are likely to get modified versions of them (by the effect of the linear combinations). Second, the source could sign the entire file. This would prevent the receivers to accept incorrect files, but it does not provide an efficient way for the receivers to recover the correct file as malicious nodes can still inject invalid packets to mount a DoS attack.

To mitigate the effect of pollution attacks two main approaches have been proposed. They can be divided into two categories: *information-theoretic* and *computational*.

Information theoretic approaches [17,18,20] use error-correction techniques to introduce redundancy in the transmitted vectors so that it is possible to reconstruct the original file as long as the number of compromised vectors is not too big. These methods have the advantage of not relying on computational assumptions, but, unfortunately, they introduce a significant overhead in the communication.

On the other hand, approaches based on computational assumptions use cryptographic techniques to provide a way for honest nodes to verify that the received packets are correct. The main tool to achieve this goal are *network coding signature schemes*. Roughly speaking, the basic requirement of such schemes is that they allow to efficiently check if a given vector is valid, i.e., it has been generated as linear combination of initial (valid) vectors $w^{(1)}, \ldots, w^{(m)}$. Two classes of network coding signatures are known: those based on homomorphic hashing [22,16,7], and those using homomorphic signatures [21,7,14,11].

In our work, we focus on the second class of schemes, that is homomorphic network coding signatures. We give relevant definitions in the following section.

2.3 Network Coding Signatures

In this section we give the definition of a network coding signature scheme and its security notion, as done by Boneh *et al.* in [7]. As we mentioned before, a network coding signature scheme allows to sign a subspace $\mathcal{W} \subset \mathbb{F}^N$ so that any vector $w \in \mathcal{W}$ is accepted, whereas vectors $w \notin \mathcal{W}$ are rejected. In particular, in

our work we focus on subspaces $\mathcal{W}$ that are described by a properly augmented basis.

We assume that a file is associated with a file identifier fid that is chosen by the source node before the transmission. In general, such fid can be the filename. Though, in our systems we need such file identifiers to be randomly chosen by the source node. Thus we think of fid as an element of an efficiently samplable set $\mathcal{I}$.

Definition 3 (Network Coding Signatures). *A* network coding signature *is defined by a triple of algorithms (*$\mathsf{NetKG}, \mathsf{NetSign}, \mathsf{NetVer}$*) such that:*

$\mathsf{NetKG}(1^k, m, n)$ *On input the security parameter k and two integers m, n, this algorithm outputs* $(\mathsf{vk}, \mathsf{sk})$ *where* sk *is the secret signing key and* vk *is the public verification key. m defines the dimension of the vector spaces while n is an upper bound to the size of the signed vectors. We assume that the public key implicitly defines the field $\mathbb{F}$ over which vectors and linear combinations are defined.*

$\mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, \mathcal{W})$ *The signing algorithm takes as input the secret key* sk*, a random file identifier* fid *and a properly augmented basis of a m-dimensional subspace $\mathcal{W} \subset \mathbb{F}^{m+\ell}$ (with $1 \leq \ell \leq n$), and it outputs a signature σ.*

$\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w, \sigma)$ *Given the public key* vk*, a file identifier* fid*, a vector $w \in \mathbb{F}^{m+\ell}$ (for $1 \leq \ell \leq n$) and a signature σ, the algorithm outputs 0 (reject) or 1 (accept).*

For correctness, we require that for all honestly generated key pairs $(\mathsf{vk}, \mathsf{sk})$, all identifiers $\mathsf{fid} \in \mathcal{I}$, all $1 \leq \ell \leq n$, and all $\mathcal{W} \subset \mathbb{F}^{m+\ell}$, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{fid}, \mathcal{W})$ then $\mathsf{Ver}(\mathsf{vk}, \mathsf{fid}, w, \sigma) = 1 \ \forall w \in \mathcal{W}$.

Security of Network Coding Signatures. The security notion of network coding signatures is defined by the following game between a challenger and an adversary $\mathcal{A}$:

Setup. The adversary chooses positive integers m, n and gives them to the challenger. The challenger runs $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{NetKG}(1^k, m, n)$ and gives vk to $\mathcal{A}$.

Signing queries. The adversary can ask signatures on vector spaces $\mathcal{W}_i \subset \mathbb{F}^{m+\ell}$ (with $\ell \leq n$) of its choice, specified by giving to the challenger a properly augmented basis describing $\mathcal{W}_i$. The challenger chooses a random file identifier fid_i, runs $\sigma_i \xleftarrow{\$} \mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}_i, \mathcal{W}_i)$ and returns σ_i to $\mathcal{A}$.

Forgery. The adversary outputs a tuple $(\mathsf{fid}^*, w^*, \sigma^*)$.

We say that the adversary wins this game if $\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}^*, v^*, \sigma^*) = 1$ and either one of the following cases holds: (1) $\mathsf{fid}^* \neq \mathsf{fid}_i$ for all i (*type-I forgery*); (2) $\mathsf{fid}^* = \mathsf{fid}_i$ for some i, but $w^* \notin \mathcal{W}_i$ (*type-II forgery*).

We define the advantage of $\mathcal{A}$ into breaking a network coding signature scheme, $\mathbf{Adv}^{NC}(\mathcal{A})$, as the probability that $\mathcal{A}$ wins the above security game, and we say that a network coding signature is secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{NC}(\mathcal{A})$ is at most negligible in the security parameter.

Finally, we give the formal definition of *homomorphic network coding signature*.

Definition 4 (Homomorphic Network Coding Signatures). *A* homomorphic network coding signature scheme *is defined by a 4-tuple of algorithms (*$\mathsf{NetKG}, \mathsf{NetSign}, \mathsf{NetVer}, \mathsf{Combine}$*) such that:*

$\mathsf{NetKG}(1^k, m, n)$**.** *On input the security parameter k and two integers $m, n \geq 1$, this algorithm outputs $(\mathsf{vk}, \mathsf{sk})$ where sk is the secret signing key and vk is the public verification key. Here, m defines the dimension of the vector spaces and $n + m$ is an upper bound to the size of the signed vectors. We assume that the public key implicitly defines the field $\mathbb{F}$ over which vectors and linear combinations are defined, and that it contains the description of an efficiently samplable distribution for fid.*

$\mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, w)$**.** *The signing algorithm takes as input the secret key sk, a file identifier in the support of fid and a vector $w \in \mathbb{F}^{\ell+m}$ (with $1 \leq \ell \leq n$) and outputs a signature σ.*

$\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w, \sigma)$**.** *Given the public key vk, a file identifier fid, a vector $w \in \mathbb{F}^{\ell}$ and a signature σ, the algorithm outputs 0 (reject) or 1 (accept).*

$\mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{(w^{(i)}, \alpha_i, \sigma_i)\}_{i=1}^{\mu})$**.** *This algorithm takes as input the public key vk, a file identifier fid, and a set of tuples $(w^{(i)}, \alpha_i, \sigma_i)$ where σ_i is a signature, $w^{(i)} \in \mathbb{F}^{\ell}$ is a vector and $\alpha_i \in \mathbb{F}$ is a scalar. This algorithm outputs a new signature σ such that: if each σ_i is a valid signature on vector $w^{(i)}$, then σ is a valid signature for w obtained from the linear combination $\sum_{i=1}^{\mu} \alpha_i \cdot w^{(i)}$.*

For correctness, we require that for all $m, n \geq 1$, all honestly generated pairs of keys $(\mathsf{vk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{NetKG}(1^k, m, n)$ the following hold:

- For all $\mathsf{fid} \in \mathcal{I}$ and all $w \in \mathbb{F}^{m+\ell}$, if $\sigma \stackrel{\$}{\leftarrow} \mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, w)$, then $\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w, \sigma) = 1$.
- For all $\mathsf{fid} \in \mathcal{I}$, any $\mu > 0$, and all sets of triples $\{(w^{(i)}, \alpha_i, \sigma_i)\}_{i=1}^{\mu}$, if $\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w^{(i)}, \sigma_i) = 1$ for all i, then it must be the case that

$$\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, \sum \alpha_i w^{(i)}, \mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{(w^{(i)}, \alpha_i, \sigma_i)\}_{i=1}^{\mu}) = 1.$$

As noticed by Boneh et al. [7], homomorphic network coding signatures are a special case of network coding signatures.

2.4 An Efficient Linear Network Coding Scheme

In this section we specify the linear network coding scheme considered in our work. Basically, it is the random network coding solution described in the previous section except that we consider some optimizations recently proposed by Gennaro *et al.* in [14]. The scheme works as follows.

The application specifies four global parameters $m, n, M, p' \in \mathbb{N}$ such that $m, n \geq 1$, and p' is a prime. In this setting, a file $\mathcal{V}$ to be transmitted is always encoded as a set of m vectors $(v^{(1)}, \ldots, v^{(m)})$ where each $v^{(i)}$ takes values in $\mathbb{F}_M^{\ell}$

where M is a bound on the initial magnitude of each coordinate and $\ell \leq n$. Since m is fixed in advance by the application, at the time of the transmission, once the size of the file $\mathcal{V}$ is known, the total length of information in every vector $v^{(i)}$ is determined. Thus, ℓ can be chosen accordingly as any number between 1 and n. The freedom in choosing ℓ is important as different choices have different impact on the efficiency of the scheme: a smaller ℓ saves bandwidth, while a larger ℓ saves computation (see [14] for more details). The parameter p' specifies the domain $P = \{0, \ldots, p'-1\}$ from which the network nodes sample the coefficients for the linear combination. Linear combinations can then be performed either over the integers, or modulo some large prime p (which is specified by the application or by the signature scheme). Gennaro et *al.* show that taking a small p' (e.g., $p' = 257$) allows to improve the performances of the network coding scheme as well as to keep a good decoding probability. In particular, they show that this holds in both cases when the linear combinations are done over the integers, or over some large prime $p > M$. Precisely, in the latter case, the performances remain better (than the case when coefficients are chosen in $\mathbb{F}_p$) as long as the bit-size of p' is negligible compared to the bit-size k of the prime p.

Global application parameters: $m, n, M, p' \in \mathbb{N}$ as specified above.

Key Generation: Each source node generates a pair of keys $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{NetKG}(1^k, m, n)$ of a homomorphic network coding signature scheme.

File transmission: On input a file $\mathcal{V}$ represented by m vectors $v^{(1)}, \ldots, v^{(m)} \in \mathbb{F}_M^\ell$ (with $\ell \leq n$), the source node generates augmented vectors $w^{(1)}, \ldots, w^{(m)}$, i.e., $w^{(i)} = (u^{(i)}, v^{(i)})$ where $u^{(i)}$ is the i-th unity vector. Next, it chooses a random file identifier $\mathsf{fid} \xleftarrow{\$} \mathcal{I}$ (recall that $\mathcal{I}$ is specified by vk), and for $i = 1$ to m, it generates $\sigma_i \xleftarrow{\$} \mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, w^{(i)})$. Finally, it sends the tuples $(\mathsf{fid}, w^{(i)}, \sigma_i)$ on its outgoing edges.

Intermediate nodes: When a node receives μ vectors $w^{(1)}, \ldots, w^{(\mu)}$ and signatures $\sigma_1, \ldots, \sigma_\mu$, all corresponding to file fid, it proceeds as follows. First, it checks that $\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w^{(i)}, \sigma_i) = 1$, for $i = 1$ to μ. It discards all the vectors (and signatures) that did not pass the check. For the remaining vectors (for simplicity, let they be $w^{(1)}, \ldots, w^{(\mu)}$), the node chooses $\alpha_1, \ldots, \alpha_\mu \xleftarrow{\$} P$, and computes: $w = \sum_{i=1}^{\mu} \alpha_i \cdot w^{(i)}$, $\sigma \leftarrow \mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{(w^{(i)}, \alpha_i, \sigma_i)\}_{i=1}^{\mu})$. Finally, the node sends $(\mathsf{fid}, w, \sigma)$ on its outgoing edges.

Target node: Once a node obtains linearly independent vectors $w^{(1)}, \ldots, w^{(m)}$ together with the respective signatures and the same file identifier fid, it first checks that they are all valid, i.e., it verifies that $\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w^{(i)}, \sigma_i) = 1$, $\forall i = 1, \ldots, m$. Given m valid vectors, the node can reconstruct the original file $(v^{(1)}, \ldots, v^m)$ as described in Section 2.2.

3 A Construction Based on SDH

In this section we propose the construction of a network coding homomorphic signature based on the Strong Diffie-Hellman assumption.

Recall that we are in the setting of the linear network coding application described in the previous section. A file $\mathcal{V}$ is represented as a set of m vectors $(v^{(1)}, \ldots, v^{(m)})$ such that each $v^{(i)} \in \mathbb{F}_p^\ell$ where p is a (publicly known) prime specified by the key generation algorithm and $\ell \leq n$. Notice that all the operations with the vectors are thus defined over the finite field $\mathbb{F}_p$, i.e., $\bmod p$. Moreover, the space for file identifiers is the set $\mathbb{Z}_p^*$ where p is the same prime specified in the key generation.

Below we give a precise description of the scheme's algorithms[2]:

$\mathsf{NetKG}(1^k, n, m)$**:** Let $\mathbb{G}, \mathbb{G}', \mathbb{G}_T$ be bilinear groups of prime order p such that $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_T$ is a bilinear map and $g \in \mathbb{G}, g' \in \mathbb{G}'$ are two generators. Pick a random $z \xleftarrow{\$} \mathbb{Z}_p$ and set $Z = (g')^z$. Choose random elements $h, h_1, \ldots, h_m, g_1, \ldots, g_n \xleftarrow{\$} \mathbb{G}$. Output the public verification key $\mathsf{vk} = (p, g, g', Z, h, h_1, \ldots, h_m, g_1, \ldots, g_n)$ and the secret key $\mathsf{sk} = z$.

$\mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, w)$**:** Let $w = (u, v) \in \mathbb{F}_p^{m+n}$ be a properly augmented vector, and let fid be randomly chosen in $\mathbb{Z}_p^*$. The signing algorithm proceeds as follows. Pick a random $s \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$X = \left(h^s \prod_{i=1}^{m} h_i^{u_i} \prod_{i=1}^{n} g_i^{v_i} \right)^{\frac{1}{z+\mathsf{fid}}}$$

Finally, output $\sigma = (X, s)$.

$\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w, \sigma)$**:** Let $\sigma = (X, s) \in \mathbb{G} \times \mathbb{Z}_p$. This algorithm checks whether σ is a valid signature on a vector $w = (u, v)$ w.r.t. the file identifier fid. If the following equation holds, then output 1, otherwise output 0:

$$e(X, Z \cdot (g')^{\mathsf{fid}}) = e(h^s \prod_{i=1}^{m} h_i^{u_i} \prod_{i=1}^{n} g_i^{v_i}, g').$$

$\mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{w^{(i)}, \alpha_i, \sigma_i\}_{i=1}^{\mu})$**:** Recall that $w^{(i)} = (u^{(i)}, v^{(i)})$ where $u^{(i)} \in \mathbb{F}_p^m$ and $v^{(i)} \in \mathbb{F}_p^n$, and that $\alpha_i \in \mathbb{F}_p$ is a randomly chosen coefficient, for all $i \in \{1, \ldots, \mu\}$. Moreover, recall that in our application this algorithm is run when every σ_i has been verified as a valid signature on $w^{(i)}$ w.r.t. fid. The algorithm computes

$$X = \prod_{i=1}^{\mu} (X_i)^{\alpha_i}, \quad s = \sum_{i=1}^{\mu} \alpha_i \cdot s_i \bmod p$$

and outputs $\sigma = (X, s)$.

Efficiency. A signature consists of one element of $\mathbb{G}$ and one element of $\mathbb{Z}_p$. Signing costs a multi-exponentiation in $\mathbb{G}$, whereas verification needs to compute two pairings, one exponentiation in $\mathbb{G}'$, i.e., $(g')^{\mathsf{fid}}$, and one multi-exponentiation.

[2] For ease of exposition, in our description we assume that the vectors w have the maximum length $m+n$. In fact, in our scheme any shorter vector with $\ell < n$ can be augmented by appending $n - \ell$ zeros.

We state the following theorem (for lack of space its proof appears in the full version of this paper [10])

Theorem 1. *If the q-SDH assumption holds in* $(p, \mathbb{G}, \mathbb{G}', \mathbb{G}_T)$ *for any polynomial* q*, then the scheme described above is a secure network coding signature.*

4 A (Strong) RSA Based Realization

In this section we describe our strong-RSA based implementation. We stress that the file to be signed is encoded as a set of vectors $(v^{(1)}, \ldots, v^{(m)})$ of ℓ components each where $\ell \leq n$ for some pre-specified bound n. Before being signed and transmitted, such vectors will be prepended with m unitary vectors $u^{(i)}$ (each having m components). We denote with $w^{(i)}$ the resulting vectors. Our implementation uses a parameter λ to specify the space $\mathcal{I}$ for the file identifiers. If M is the bound on the initial magnitude of each vector component, then $2^\lambda > M$ and $\mathcal{I}$ is the set of prime numbers of (exactly) $\lambda + 1$ bits, greater than 2^λ.

Finally, we notice that in this scheme the exact finite field over which are done the linear combinations is different for each file. In particular, it will be $\mathbb{F}_e$ where $e = \mathsf{fid}$ (e is a prime number) is the file identifier chosen by the sender. More precisely, this means that whenever a vector space $\mathcal{W}$ has to be signed, a file identifier $\mathsf{fid} = e$ is chosen (as a sufficiently large prime) and it is associated to $\mathcal{W}$. Thus, linear combinations are done $\bmod e$ and $w \notin \mathcal{W}$ implies that w cannot be written as a linear combination $\bmod e$ of vectors of $\mathcal{W}$.

A precise description of our network coding scheme $\mathsf{NetPFSig} = (\mathsf{NetKG}, \mathsf{NetSign}, \mathsf{NetVer}, \mathsf{Combine})$ follows.

$\mathsf{NetKG}(1^k, \lambda, m, n)$**.** The NetKG algorithm chooses two random (safe) primes p, q of length $k/2$ each. It sets $N = pq$ and proceeds by choosing $g, g_1, \ldots, g_n$, $h_1, \ldots, h_m$ at random (in $\mathbb{Z}_N^*$). In addition to k, here we assume an additional security parameter λ which specifies the space $\mathcal{I}$ of file identifiers as described before. The public key is set as $(N, g, g_1, \ldots, g_n, h_1, \ldots, h_m)$, while the secret key is (p, q).

$\mathsf{NetSign}(\mathsf{sk}, \mathsf{fid}, w)$**.** The signing algorithm proceeds as follows. Let $w = (u, v) \in \mathbb{F}_M^{m+n}$ and let fid be a random file identifier, which is a prime number of the form specified before. For ease of exposition, let $e = \mathsf{fid}$. The signer chooses a random element $s \in \mathbb{Z}_e$ and uses its knowledge of p and q to solve the following equation

$$x^e = g^s \prod_{j=1}^{m} h_j^{u_j} \prod_{j=1}^{n} g_j^{v_j} \bmod N$$

Finally, it outputs the signature $\sigma = (s, x)$.

$\mathsf{NetVer}(\mathsf{vk}, \mathsf{fid}, w, \sigma)$**.** To verify a signature $\sigma = (s, x)$ on a vector w, the verification algorithm proceeds as follows. Let $e = \mathsf{fid}$.

- Check that e is an odd number of the right size (i.e. $\lambda + 1$ bits).
- Check that all the u's, v's and s are in $\mathbb{Z}_e$.

– Check that the equation $x^e = g^s \prod_{j=1}^{m} h_j^{u_j} \prod_{j=1}^{n} g_j^{v_j} \bmod N$ is satisfied by the given x.
– If all the checks above are satisfied, output 1, otherwise 0.

$\mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{w^{(i)}, \alpha_i, \sigma_i\}_{i=1}^{\mu})$. To combine signatures σ_i, corresponding to vectors $w^{(i)}$ sharing the same fid, the algorithm proceeds as follows.

– It computes

$$w = \sum_{i=1}^{\mu} \alpha_i \cdot w^{(i)} \bmod e,\ w' = (\sum_{i=1}^{\mu} \alpha_i \cdot w^{(i)} - w)/e$$

$$s = \sum_{i=1}^{\mu} \alpha_i s_i \bmod e,\ s' = (\sum_{i=1}^{\mu} \alpha_i s_i - s)/e$$

Let $w' = (u', v')$. It outputs $\sigma = (s, x)$ where x is obtained by computing:

$$x = \frac{\prod_{i=1}^{\mu} x_i^{\alpha_i}}{g^{s'} \prod_{j=1}^{m} h_j^{u_j'} \prod_{j=1}^{n} g_j^{v_j'}} \bmod N$$

To complete the description of the scheme we show its correctness. In particular, while the correctness of the signatures returned by the signing algorithm can be easily checked by inspection, we pause to show that also the signatures obtained from the $\mathsf{Combine}$ algorithm are correct. Assume that for $i = 1$ to μ, $\sigma_i = (x_i, s_i)$ is a valid signature on the vector $w^{(i)} = (u^{(i)}, v^{(i)})$, and let α_i be the integer coefficients of the linear combination. Let $\sigma = (x, s)$ be the signature as computed by $\mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, \{w^{(i)}, \alpha_i, \sigma_i\}_{i=1}^{\mu})$. We have that:

$$x^e = \frac{\prod_{i=1}^{\mu} (x_i^e)^{\alpha_i}}{(g^{s'} \prod_{j=1}^{m} h_j^{u_j'} \prod_{j=1}^{n} g_j^{v_j'})^e} \tag{1}$$

$$= \frac{g^{\sum_{i=1}^{\mu} s_i \alpha_i} \prod_{j=1}^{m} h_j^{\sum_{i=1}^{\mu} u_j^{(i)} \alpha_i} \prod_{j=1}^{n} g_j^{\sum_{i=1}^{\mu} v_j^{(i)} \alpha_i}}{(g^{s'} \prod_{j=1}^{m} h_j^{u_j'} \prod_{j=1}^{n} g_j^{v_j'})^e} \tag{2}$$

$$= g^{(\sum_{i=1}^{\mu} s_i \alpha_i - s'e)} \prod_{j=1}^{m} h_j^{(\sum_{i=1}^{\mu} u_j^{(i)} \alpha_i - u_j' e)} \prod_{j=1}^{n} g_j^{(\sum_{i=1}^{\mu} v_j^{(i)} \alpha_i - v_j' e)} \tag{3}$$

$$= g^s \prod_{j=1}^{m} h_j^{u_j} \prod_{j=1}^{n} g_j^{v_j} \tag{4}$$

which shows correctness as desired. Above, equation (2) is justified by that each σ_i is valid, and equation (4) follows from the definition of s' and $w' = (u', v')$ as computed in the $\mathsf{Combine}$ algorithm.

Efficiency. Each signature consists of an element of $\mathbb{Z}_N$ and one integer of λ bits. Signing costs one full exponentiation and one multi-exponentiation in $\mathbb{Z}_N$ with

λ-bits exponents, plus the sampling of a random prime number (which is dominated by the cost of prime verification). The verification needs an exponentiation with a $(\lambda + 1)$-bits prime, x^e, and one multi-exponentiation with λ-bits exponents.

Here we state the following theorem (again for lack of space the proof appears in [10]).

Theorem 2. *Under the Strong-RSA assumption, the scheme described above is a secure homomorphic network coding signature.*

5 Efficiency and Comparisons

In this section we discuss the efficiency of our two constructions and compare it to that of other known homomorphic network coding signatures. As we already mentioned, there are not that many schemes in the literature realizing this primitive: a few constructions [7,14,9,8] rely on random oracles, and a couple of more recent schemes [3,11,13] are proven secure in the standard model. We should also mention that there are other schemes in the standard model based on homomorphic hashing. However these are less appealing in practice mainly because the basis vectors have to be signed all at once, which means that in the network coding application the source node must know the entire file before sending the first packet. This is not desirable in several applications, e.g. a source node which is a sensor collecting data in some time interval, or streaming applications. Moreover, the authentication information to be sent along with the packets is quite long.

Therefore, we compare our constructions with the schemes in the standard model, and later in this section we will briefly discuss a comparison with the random oracle based ones.

In the scheme by Attrapadung and Libert [3] a signature consists of three group elements where the bilinear groups have composite order N, with N product of three primes. To compute a signature, the scheme needs to perform two multi-exponentiations and one exponentiation, whereas the verification time is dominated by the computation of four pairings in such composite order groups. Even if one applies standard techniques to convert the scheme in prime order groups (as suggested in [3]), the overhead would still remain significant.

In [13] Freeman proposes a general framework, that can be seen as a generalization of the Attrapadung and Libert methodology, for converting signature schemes with certain properties into linearly homomorphic ones. There are two appealing features in Freeman's work. First, his model allows for a stronger adversary than the one we consider. Second, the proposed approach is general enough to work with several currently known signature schemes. However, all the resulting (linearly homomorphic) signatures are less efficient than those given in this paper.

In the scheme by Catalano, Fiore and Warinschi [11] each signature consists of an element of $\mathbb{Z}_N^*$ and an integer s of $\lambda_s = 3k+|N|$ bits, where k is the security parameter and $|N|$ is the bit size of the RSA modulus N (which is related to k). Signing and verifying both need one multi-exponentiation (where all exponents have size λ, except one of size λ_s) and one exponentiation. Since in this scheme the linear combinations are done over the integers, it can support only a limited number of linear combinations, that in the network coding application translates to supporting only networks with paths of predetermined bounded length. Technically, the reason of such bound is that the vector coordinates cannot be let grow more than the size of the prime e.

In this scenario, our solution based on q-SDH seems the most efficient in terms of both bandwidth and computation. In fact, recall that in our case a signature is one group element plus one element of $\mathbb{Z}_p$: 512 bits in total, if one considers $k = 128$ bits of security and asymmetric pairings. The operations for signing and verifying are similar in all the schemes, but our SDH construction has the advantage that such operations can be performed over prime order groups. Our RSA realization, can be seen as a significant optimization of the Catalano-Fiore-Warinschi's scheme [11]. There are two main improvements. First, our scheme allows for a much smaller exponent s. In fact, in our case s can be of λ bits, that is even more than 10 times shorter than in [11], if one considers 128 bits of security. Intuitively, the reason of using a large s in [11] is that in the real scheme s is truly random, while in the simulation it is used to hide some information of $2k + |N|$ bits, which decreases its entropy down to k bits. So, there s is taken sufficiently large to keep it within negligible statistical distance from a uniform value of λ_s bits. In our case, s is in $\mathbb{Z}_e$, and we take advantage of modular reduction to obtain a uniformly distributed s also in the simulation. Notice that having such a short s saves in both bandwidth and computation. Second, our idea of computing all the linear combinations $\pmod e$ avoids the problem that the vector coordinates may grow beyond e. In this way we can support networks with paths of any lengths, which was not the case in the previous RSA-based schemes [11] and [14].

Finally, we consider the schemes in the random oracle model that work over similar algebraic settings, i.e., bilinear groups [7] and RSA [14]. Compared to them, our solutions are (not surprisingly) slightly worse. The main difference is the size of the public key that in our case is linear in $m+n$, whereas in [7,14] it is constant (because $O(m + m)$ group elements are generated on-the-fly using the random oracle). On the other hand, the size of a signature and the time needed to sign and verify are somewhat comparable. In this sense, we believe that our solutions offer a good compromise if one does not want to rely on the random oracle heuristic.

Acknowledgements. The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The authors would like to thank Dennis Hofheinz and Eike Kiltz for helpful discussions in the early stage of this work.

References

1. Ahlswede, R., Cai, N., Li, S., Yeung, R.W.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012), http://eprint.iacr.org/2011/096
3. Attrapadung, N., Libert, B.: Homomorphic Network Coding Signatures in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
4. Barić, N., Pfitzmann, B.: Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
5. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology 21(2), 149–177 (2008)
7. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
8. Boneh, D., Freeman, D.M.: Homomorphic Signatures for Polynomial Functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
9. Boneh, D., Freeman, D.M.: Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
10. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. Cryptology ePrint Archive, http://eprint.iacr.org/2011/696
11. Catalano, D., Fiore, D., Warinschi, B.: Adaptive Pseudo-free Groups and Applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (2011)
12. Chou, P.A., Wu, Y., Jain, K.: Practical network coding. In: 41st Allerton Conference on Communication, Control and Computing (2003)
13. Freeman, D.M.: Improved Security for Linearly Homomorphic Signatures: A Generic Framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012), http://eprint.iacr.org/2012/060
14. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure Network Coding over the Integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
15. Gkantsidis, C., Rodriguez, P.: Network coding for large scale content distribution. In: Proc. of IEEE INFOCOM 2005, pp. 2235–2245 (2005)
16. Ho, T., Koetter, R., Médard, M., Karger, D., Effros, M.: The benefit of coding over routing in a randomized setting. In: Proc. of International Symposium on Information Theory (ISIT), p. 442 (2003)
17. Ho, T., Leong, B., Koetter, R., Médard, M., Effros, M., Karger, D.: Byzantine modification detection in multicast networks using randomized network coding. In: Proc. of International Symposium on Information Theory (ISIT), pp. 144–152 (2004)

18. Ho, T., Médard, M., Koetter, R., Karger, D.R., Effros, M., Shi, J., Leong, B.: A random linear network coding approach to multicast. IEEE Transactions on Information Theory 52, 4413–4430 (2006)
19. Hofheinz, D., Kiltz, E.: Programmable Hash Functions and Their Applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
20. Jaggi, S., Langberg, M., Katti, S., Ho, T., Katabi, D., Médard, M., Effros, M.: Resilient network coding in the presence of byzantine adversaries. IEEE Transactions on Information Theory 54, 2596–2603 (2008)
21. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
22. Krohn, M., Freedman, M., Mazieres, D.: On the-fly verification of rateless erasure codes for efficient content distribution. In: 2004 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 9–12, pp. 226–240. IEEE Computer Society Press (2004)
23. Robert-Li, S.-Y., Yeung, R.Y., Cai, N.: Linear network coding. IEEE Transactions on Information Theory 49(2), 371–381 (2003)
24. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)