EURECOM
Department of Mobile Communications
2229, route des Crêtes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-07-202

TitleOfReport

septemberth, 2007
Last update April th, 2008

Melek Önen, Refik Molva, Abdullatif Shikfa

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {onen,molva,shikfa}@eurecom.fr

[1]This research is partially supported by the HAGGLE european project

Homomorphic signatures for Network Coding

Melek Önen, Refik Molva, Abdullatif Shikfa

Abstract

Network coding allows intermediate nodes to mix data in order to achieve higher throughput and better reliability. Hence, nodes combine multiple packets into a single packet before forwarding it. Such scheme needs efficient authentication and integrity mechanisms in order to prevent pollution attacks whereby an attacker injects bogus messages into the system instead of network coded blocks. Such an attack has the potential impact of infecting all subsequent message exchanges between nodes that received polluted messages. In this paper, we propose $SigNCode$, a new integrity mechanism based on homomorphic operations allowing an on-the-fly verification of the integrity of a network coded packet and therefore preventing pollution attacks. Thanks to this new mechanism, any intermediate node is capable of constructing a correct signature for a linear combination of messages originating from the source. The proposed mechanism is based on the use of bilinear pairings and relies on a single communication channel. In order to evaluate the security of our signature scheme we also developed a new security definition and a proof model that encompass the extended integrity notion underlying network coding. This definition and the model help distinguish some legitimate forgery such as linear combinations of original data blocks from pure forgery such as injection of bogus data.

1 Introduction

Network coding is a fundamental concept that recently fostered new research in peer-to-peer or ad hoc networking. As introduced by the seminal paper [1] by Ahlswede et al, the core notion of network coding is to mix data at intermediate nodes in order to achieve higher throughput and better reliability to errors and data corruption. The network coding operation performed by each node consists in forwarding a linear combination of input messages to neighboring nodes. The elegance and performance advantages of network coding on the other hand come with a high price that is due to the sensitivity of the basic network coding mechanism to the injection of bogus combinations by malicious nodes through the so called *pollution attacks*. Due to the strong diffusion effect of network coding, an attacker that forwards bogus messages instead of well-formed combinations of authentic inputs can quickly jeopardize the operation of network coding in large portions of the network by preventing legitimate nodes from retrieving messages transmitted by the source.

One naturally seeks a solution to prevent such attacks in the area of data integrity mechanisms. Ideally, a typical data integrity solution would allow each node to tell apart legitimate messages resulting from network coding from bogus data injected by a pollution attack. The basic principle of classical data integrity consists in transmitting a redundant piece of information called integrity check value along with each data fragment or packet and allowing each recipient to compare the integrity check value with the result of some integrity verification function computed over the packet. These mechanisms thus allow recipients to detect if a message has been transmitted end-to-end without any modification or tampering by intruders or random errors. The underpinnings of classical data integrity mechanisms thus are digital signatures that allow legitimate source and recipient to respectively compute and verify the integrity check value as a short representative of each message while preventing an intruder from computing it.

In case of network coding, the first shortcoming of classical integrity techniques is due to the fact that by the very definition of network coding the messages are to be modified by legitimate linear combination operations. Network coding thus calls for a new concept of integrity whereby some modifications of messages that are conformant with network coding principles should be authorized whereas other modifications such as malicious tampering or injection of garbage that aim at jeopardizing network coding should be prevented.

The core question of data integrity with network coding is thus the design of signatures that are compatible with the network coding operation and that are also called homomorphic signatures. Using such functions legitimate nodes would be able to compute an integrity check value for a linear combination of two or several inputs using the individual integrity check values of each input. However, by their very definition, homomorphic signature schemes would not satisfy the security against existential forgery requirement because their main goal is to authorize a third node to compute valid signatures without knowing the secret key. Therefore,

homomorphic signatures require a new definition of security whereby some types of forgeries are authorized whereas all others are unauthorized. In the context of network coding, authorized forgeries consist of any linear combination of original blocks generated by the source: an adversary should not be able to compute a valid signature of a bogus message that is not a correctly computed combination of original blocks.

In this paper, we define a new security model for homomorphic signatures in the context of network coding and further suggest a data integrity mechanism which is an extension of an identity-based signature technique. Unlike some recent proposal, our scheme is very efficient in terms of bandwidth utilization in that it does not require the distribution of integrity check values for each original data block. Our scheme does not require prior distribution of secrets to networks nodes either. The scheme is implemented by a protocol called $SigNCode$ that allows any party to verify that an input message is a valid combination of two or several legitimate combinations of original data blocks only based on the ID of the source and the value of a check value that is transmitted along with the input message. $SigNCode$ also allows each party to compute a valid signature for any combination of two or several valid inputs based on the inputs, the ID of the source and the check value that is transmitted along with each input.

The main features of our approach are:

- an extended notion of integrity suited to network coding whereby any linear combination of authentic data blocks are deemed legitimate;
- a data integrity mechanism that achieves the extended integrity notion without direct feedback from the source.

In order to evaluate the security of the signature scheme we also developed a proof model that encompass the extended integrity notion. This definition and the model help distinguish some legitimate forgery such as linear combinations of original data blocks from pure forgery such as injection of bogus data.

2 Problem Statement

2.1 Network coding

Network coding allows intermediate nodes to mix the information in packets before forwarding them. For such mechanisms, the network is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges between these nodes. When a source $S \in \mathcal{V}$ wishes to send a file F to a set $T \subseteq \mathcal{V}$ of vertices, F is subdivided into n blocks such as $F = b_1|b_2|..|b_n$. The source computes a different linear combination of the blocks for each following intermediate node and sends the result of this linear combination with the vector of coefficients. Each intermediate node $N_i \in \mathcal{V}$ receiving packets from several nodes, computes a

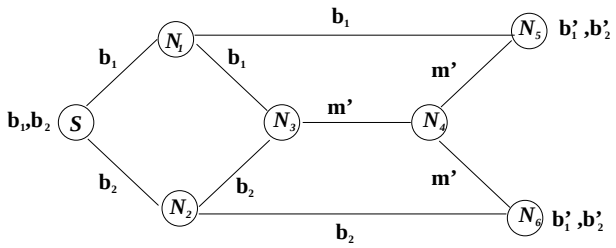

Figure 1: Network coding: Example

linear combination of the received values and obtains a new linear combination of initial blocks as follows:

$$y_i = \sum_{1 \leq j \leq n} \alpha_{i,j} b_j$$

y_i is sent to the next edge with the vector $< \alpha_{i,1}, \alpha_{i,2}, .., \alpha_{i,n} >$.

In order to illustrate this mechanism, we define a network with seven nodes represented in figure 2.1. In this particular network, the source node S wishes to send a file $F = b_1|b_2$ to nodes N_5 and N_6 and thus sends b_1 to N_1 and b_2 to node N_2. In this particular network, node N_3 performs a simple linear combination whereby the coefficients corresponding to original blocks are $< 1, 1 >$. Thanks to this operation, nodes N_5 and N_6 are able to perform the decoding operation in order to retrieve b_1 and b_2.

2.2 Structure of a network coded message

Consider a source node S that wishes to send a file F. F is first subdivided into n blocks such as $F = b_1|b_2|..|b_n$ where $|$ denotes the concatenation. Then, each block is subdivided into k sub-blocks such as $b_l = (b_{l,1}, .., b_{l,k})$. Therefore, F can be represented by the following matrix:

$$\begin{pmatrix} b_{1,1} & .. & b_{1,k} \\ . & . & . \\ . & . & . \\ . & . & . \\ b_{n,1} & ... & b_{n,k} \end{pmatrix}$$

A network coded message is defined as a vector regrouping both coefficients that are used in the computation of the linear combination of original block and k sub-blocks. We thus have the following representation:

$$m_i = (\alpha_{i,1}, .., \alpha_{i,n}, m_{i,1}, .., m_{i,k})$$

where

$$m_{i,j} = \alpha_{i,1} b_{1,j} + .. + \alpha_{i,n} b_{n,j}$$

with $1 \leq j \leq k$.

Let m_1 and m_2 be two network coded messages defined as follows:

$$m_1 = (\alpha_{1,1}, .., \alpha_{1,n}, m_{1,1}, .., m_{1,k})$$
$$m_2 = (\alpha_{2,1}, .., \alpha_{2,n}, m_{2,1}, .., m_{2,k})$$

The result of a linear combination of these two network coded messages with coefficients β_1 and β_2 is a new message m_3 defined as follows:

$$m_3 = (\alpha_{3,1}, .., \alpha_{3,n}, m_{3,1}, .., m_{3,k})$$

for each i and j such that $1 \leq i \leq n$ and $1 \leq j \leq k$, $\alpha_{3,i}$ and $m_{3,j}$ are defined as follows:

$$\alpha_{3,i} = \beta_1 \alpha_{1,i} + \beta_2 \alpha_{2,i}$$
$$m_{3,j} = \beta_1 m_{1,j} + \beta_2 m_{2,j}$$

2.3 Pollution attacks

Network coding has been theoretically proved to maximize network throughput [1–3] and recent implementations demonstrate its practical benefits [4]. However, new security issues appear with network coding. Indeed, in the case where some malicious node injects a single bogus packet, all the information turns out to be contaminated and therefore receiving nodes end up with an incorrect decoding operation. Because any message can be defined as a linear combination of original blocks, a bogus network coded message if there is no concordance between the coefficients and the result of the linear combination (the message sub-blocks).

Consider again the previously defined file $F = b_1|..|b_n$. Let m'_3 defined as follows:

$$m'_3 = (\alpha'_{3,1}, .., \alpha'_{3,n}, m'_{3,1}, .., m'_{3,k})$$

m'_3 is defined as bogus when at least one subblock is not the result of a linear combination using the coefficients $\{\alpha_{3,i}\}$, that is when for at least one j such that $1 \leq j \leq k$ we have :

$$m'_{3,j} \neq \alpha'_{3,1} b'_{1,j} + .. + \alpha'_{3,n} b_{n,j}$$

Such an attack has the potential impact of infecting subsequent message exchanges between nodes that received polluted messages. In the example illustrated in figure 2.3, when an attacker injects a bogus message m', this message would impact the incorrect decoding operation by all receivers.

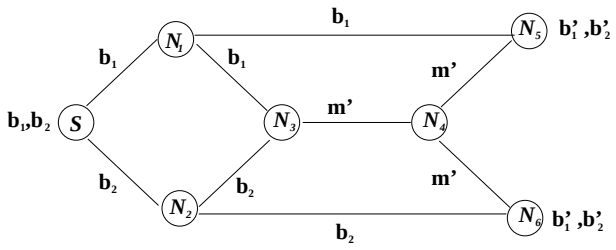

Figure 2: Pollution attacks against network coding

Since intermediate nodes do not have knowledge of original blocks, they cannot verify the correctness of received network coded blocks. Therefore, network coding mechanisms need efficient authentication and integrity mechanisms in order to prevent such pollution attacks. Indeed, before encoding received packets, a node should first verify the integrity of incoming packets and the concordance between coefficients and message blocks. The use of cryptographic signatures is a solution to provide integrity. However, each packet should be authenticated by the source itself because intermediate nodes can be malicious and only the source is trusted. In this case, since packets are inherently modified at each intermediate node, the integrity code resulting from linear combinations originating from the source should also take into account such modification.

Therefore, integrity mechanisms dedicated to network coding applications should be based on homomorphic operations. Thanks to this property, any intermediate node would be able to construct a correct signature over both coefficients and message blocks for a new linear combination of messages originating from the source without having access to the source's private key.

However, homomorphism inherently does not satisfy security against existential forgery requirement. Therefore, in the next section, we formally define a new security model for homomorphic signatures dedicated to network coding applications.

3 Homomorphic Signatures for Network Coding

3.1 Definition

A homomorphism is defined as a map $\phi : X \rightarrow Y$ such that:

$$\phi(x \cdot y) = \phi(x) \circ \phi(y)$$

where $\cdot$ and $\circ$ respectively are the operations in X and Y.

Therefore, given two network coded messages m_1 and m_2 and their respective signatures σ_1 and σ_2, σ is homomorphic if it is easy to compute the signature σ_3 of a new network coded message m_3. Indeed, if m_3 is defined as $m_3 = \beta_1 m_1 + \beta_2 m_2$ then, its signature σ_3 can be computed in a similar way, that is: $\sigma_3 = \beta_1\sigma_1 + \beta_2\sigma_2$.

3.2 Security with Homomorphic signatures for network coding

The standard notion of security against existential forgeries is too strong for homomorphic signatures: no homomorphic signature scheme could even satisfy it, because given two signatures on messages m_1, m_2 one can generate a signature on the new message $m_3 = m_1 + m_2$ without asking the signer for a signature on m_3 explicitly.

We therefore need a new definition of security. Authors in [5] define the security of homomorphic signature as follows:

Definition 1 *We say that a homomorphic signature scheme Sig is (t, q, ϵ)-secure against existential forgeries with respect to $\oplus$ if every adversary $\mathcal{A}$ making at most q chosen-message queries and running in time at most t has advantage $Adv\mathcal{A} \leq \epsilon$. The advantage of an adversary $\mathcal{A}$ is defined as the probability that, after queries on the messages $x_1, .., x_q$, $\mathcal{A}$ outputs a valid signature $< x', y' >$ on some message $x' \notin span_{\oplus}(x_1., .., x_q)$.*

In this definition, some types of forgeries are authorized whereas all others are unauthorized. Authorized forgeries consist of combinations of legitimate messages under the homomorphic operation $\oplus$ to result in another legitimate message. All other types of forgeries are deemed unauthorized.

In the context of network coded messages, authorized forgeries consist of any linear combination of original blocks generated by the source. Hence, an adversary should not be able to compute a valid signature of a bogus network coded message including both coefficients and message blocks $m'_3 = (\alpha'_{3,1}, .., \alpha'_{3,n}, m'_{3,1}, .., m'_{3,k})$ if there exists at least one j such that $1 \leq j \leq k$ and $m'_{3,j} \neq \alpha'_{3,1} b'_{1,j} + .. + \alpha'_{3,n} b'_{n,j}$.

Furthermore, since authorized forgeries are limited to the span space of signatures of network coded blocks, signature keys should also depend on these blocks. If a source wishes to distribute two different files that are defined in two different span spaces, one attacker can succeed to find a signature of a bogus network coded message that is a correct one for the second distribution. In order to avoid such attacks, homomorphic signatures for network coded messages should be based on the use of one-time signature keys. Each time a source wishes to distribute a different file that is a set of different blocks, then a new signature key should also be generated. Hence, in addition to the identity, signature keys should also depend on the files.

4 SigNCode

We now describe a homomorphic signature scheme that detects pollution attacks in the context of network coding as defined previously.

4.1 Preliminaries

The proposed scheme is based on elliptic curve cryptography. Therefore, we first define the useful security primitives. Consider an additive cyclic group G_1 of prime order q and a cyclic multiplicative group G_2. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a map which satisfies the following properties:

- **bilinear:** for all $P, Q \in G_1$ and all $a, b \in Z$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$
- **non-degenerate** The map does not send all pairs from $G_1 \times G_1$ to the identity in G_2.
- **computable** there is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Such bilinear maps are considered as admissible bilinear maps and can be obtained by the Weil or Tate pairing [6] over supersingular elliptic curves or abelian varieties.

The security of encryption or integrity schemes that are based on bilinear maps relies on the hardness of the following problems.

Let G_1 be a cyclic group of prime order q and P a generator of G_1.

- The Decisional Diffie-Hellman Problem (DDH) in G_1 is to distinguish between the distributions $< P, aP, bP, abP >$ and $< P, aP, bP, cP >$ where a, b, c are random in Z_q
- The Computational Diffie-Hellman Problem (CDH) in G_1 is to compute abP given $< P, aP, bP >$

A gap Diffie-Hellman group (GDH) is defined as a group where the DDH problem is efficiently solved and where there exists no algorithm which can solve the CDH problem with non-negligible probability within polynomial time. Our scheme relies on such groups.

Bilinear maps inherently provide homomorphism thanks to their property of bilinearity. Indeed, let Q_1, Q_2 and P_1, P_2 be four elements of an additive cyclic group G_1 and α_1 and α_2 two scalar coefficients. We have the following property:

$$\hat{e}(\alpha_1 Q_1 + \alpha_2 Q_2, P) = \hat{e}(Q_1, P)^{\alpha_1} \hat{e}(Q_2, P)^{\alpha_2}$$

$$\hat{e}(Q_1, \alpha_1 P_1 + \alpha_2 P_2) = \hat{e}(Q_1, P_1)^{\alpha_1} \hat{e}(Q_1, P_2)^{\alpha_2}$$

4.2 SigNCode Description

We propose a new homomorphic integrity mechanism based on bilinear pairings allowing an on-the-fly verification of the integrity of a network coded packet

in order to prevent pollution attacks. Thanks to this new mechanism any intermediate node is capable of constructing a correct signature for a linear combination of messages originating from the source. This scheme is based on a signature scheme proposed in [7], but this existing scheme is modified and adapted to network coding mechanisms.

SigNCode is ID-based and therefore there is no need for a preliminary phase of key distribution. Identities are public and thus public keys that are used for signature verification can be derived from the identity of the source and a file id. Moreover, in order not to let an adversary to use signatures of previous instances originating from other files in order to compute new signatures, SigNCode are based on one-time signatures: a different set of keys is defined for each different file.

The proposed scheme consists of four algorithms: *Setup*, *Extract*, *Sign* and *Verify*. In order to describe these algorithms we first remind that a network coded message m_i regrouping both coefficients and subblocks is represented as follows:

$$m_i = (\alpha_{i,1}, ..., \alpha_{i,n}, m_{i,1}, .., m_{i,k})$$

- **Setup:** Given a GDH group G and its generator P, pick a random s and set $P_{pub} = sP$. Let $F : Z_q \times G \rightarrow Z_q$ be a linear function and $H : \{0,1\}^* \rightarrow G^*$ be a hash function. The system parameters are (P, P_{pub}, F, H). The master key is s.

- **Extract**: This algorithm deals with secret generation. As previously explained, signature keys strongly depend on both identities and the file. Therefore, given an identity ID and a file F_i represented as described previously, the algorithm first defines a subblock identity $BID_{i,j}$ for each element of the vector including coefficients corresponding to a network-coded message. These subblock identities are unique and cannot be reused for another file distribution. Then, the algorithm computes $Q_j = H(ID|BID_{i,j})$ for each j such that $1 \leq j \leq n + k$ and $D_j = sQ_j$. The algorithm outputs $\{D_j\}_{1 \leq j \leq n+k}\}$ as the set of private keys corresponding to identity ID and message m_i.

- **Sign**: Given identity ID and message m_i, the algorithm first generates a random number $r_i \in Z_q$ and then computes the signature $\sigma_i = (U_i, V_i)$ such that:

$$\begin{cases} U_i = r_i \sum_{j=1}^{n+k} Q_j \\ V_i = r_i \sum_{j=1}^{n+k} D_j + \sum_{j=1}^{n} \alpha_{i,j} D_j + \sum_{j=1}^{k} m_{i,j} D_{n+j} + f_i \sum_{j=1}^{n+k} D_j \end{cases}$$

where $f_i = F(\sum_{j=1}^{k} m_{i,j}, U_i)$.

- **Verify**: For a given message $m_i = (\alpha_{i,1}, .., \alpha_{i,n}, m_{i,1}, .., m_{i,k})$, identity ID and a signature $\tilde{\sigma_i} = (\tilde{U_i}, \tilde{V_i})$, $\tilde{\sigma_i}$ is valid if

$$\hat{e}(\tilde{V_i}, P) = \hat{e}(\tilde{U_i} + \sum_{j=1}^{n} \alpha_{i,j} Q_j + \sum_{j=1}^{k} m_{i,j} Q_{n+j} + f_i \sum_{j=1}^{n+k} Q_j, P_{pub})$$

where $f_i = F(\sum_{j=1}^{k} m_{i,j}, U_i)$.

4.3 Homomorphism with SigNCode

In this section, we show that the proposed scheme provides homomorphism. Let two network coded messages m_1 and m_2 be defined as follows:

$$m_1 = (\alpha_{1,1}, .., \alpha_{1,n}, m_{1,1}, .., m_{1,k})$$
$$m_2 = (\alpha_{2,1}, .., \alpha_{2,n}, m_{2,1}, .., m_{2,k})$$

Given $\{\beta_1, \beta_2\} \in Z_q$, we define a new network coded message m_3 as the result of a linear combination of the previous two messages with these coefficients. We thus have:

$$\alpha_{3,i} = \beta_1 \alpha_{1,i} + \beta_2 \alpha_{2,i}$$
$$m_{3,j} = \beta_1 m_{1,j} + \beta_2 m_{2,j}$$

where $1 \leq i \leq n$ and $1 \leq j \leq k$.

Since F is a linear function we have $f_3 = \beta_1 f_1 + \beta_2 f_2$.

Now, let $\sigma_1 = (U_1, V_1)$ and $\sigma_2 = (U_2, V_2)$ the signatures respectively corresponding to m_1 and m_2 resulting from $SigNCode$ given an identity ID. We thus have:

$$\begin{cases} U_1 = r_1 \sum_{j=1}^{n+k} Q_j \\ V_1 = r_1 \sum_{j=1}^{n+k} D_j + \sum_{j=1}^{n} \alpha_{1,j} D_j + \sum_{j=1}^{k} m_{1,j} D_{n+j} + f_1 \sum_{j=1}^{n+k} D_j \end{cases}$$

$$\begin{cases} U_2 = r_2 \sum_{j=1}^{n+k} Q_j \\ V_2 = r_2 \sum_{j=1}^{n+k} D_j + \sum_{j=1}^{n} \alpha_{2,j} D_j + \sum_{j=1}^{k} m_{2,j} D_{n+j} + f_2 \sum_{j=1}^{n+k} D_j \end{cases}$$

where f_1 and f_2 are defined as:

$$f_1 = F(\sum_{j=1}^{k} m_{1,j}, U_1)$$
$$f_2 = F(\sum_{j=1}^{k} m_{2,j}, U_2)$$

The signature σ_3 corresponding to m_3 would have been computed by the source as follows:

- r_3 is defined as $r_3 = \beta_1 r_1 + \beta_2 r_2$;
- $U_3 = r_3 \sum_{j=1}^{n+k} Q_j = (\beta_1 r_1 + \beta_2 r_2) \sum_{j=1}^{n+k} Q_j$;
- $f_3 = F(\sum_{j=1}^{k} m_{3,j}, U_3)$;
- $V_3 = r_3 \sum_{j=1}^{n+k} D_j + \sum_{j=1}^{n} \alpha_{3,j} D_j + \sum_{j=1}^{k} m_{3,j} D_{n+j} + f_3 \sum_{j=1}^{n+k} D_j$

Any other intermediate node can compute this signature by only using U_1, U_2, f_1, and f_2 as follows:

$$\begin{cases} U_3 = \beta_1 U_1 + \beta_2 U_2 \\ f_3 = \beta_1 f_1 + \beta_2 f_2 \\ V_3 = \beta_1 V_1 + \beta_2 V_2 \end{cases}$$

Consequently, from existing correctly signed messages, any node can compute new signatures for messages that are the result of any linear combination without the knowledge of private keys.

5 Security Analysis

5.1 Security Model for SigNCode

In the security definition for $SigNCode$, authorized forgeries consist of any linear combination of original blocks generated by the source. Based on this exten-

sion of the concept of existential forgery in the context of homomorphic functions, we came up with a new security model that is defined as follows:

Definition 2 *$SigNCode$ is said to be existentially unforgeable if no probabilistic polynomial time adversary has a non-negligible advantage in the following game:*

1. ***Setup**: The challenger runs the algorithm **Setup** of $SigNCode$ and obtains both the public parameters and the master secret. The adversary is given these public parameters but the master secret is kept by the challenger.*

2. ***Queries**: The adversary adaptively makes a number of different queries to the challenger. Each query can be one of the following.*

   - ***Extract query**: the adversary can ask for the private key of any identity ID and message m. The challenger responds by running the algorithm **Extract** of $SigNCode$ and forwards private keys $\{D_j\}_{1\leq j\leq n+k}$ to the adversary.*
   - ***Sign query**: the adversary can ask for the signature of any identity ID on any message m. The challenger responds by first running **Extract** for (ID, M) to obtain private keys $\{D_j\}_{1\leq j\leq n+k}$, and then running **Sign** to obtain a signature, which is forwarded to the adversary.*
   - ***Other queries**: the adversary can ask the result of F or H on respectively a tuple (M, U) or a message m.*

3. ***Forgery**: The adversary outputs $(ID^*, M^*, f^*, r^*, U^*, V^*)$. The adversary succeeds if the following hold true:*

   *(a) ID^* does not appear in any of the **Extract** queries;*

   (b) M^* is not the result of a linear combination of any set of messages that appear in the different queries;

   (c) $U^* = r^* \sum_{j=1}^{n+k} Q_j^*$;

   (d) $f^* = F(\sum_{j=1}^{k} m_{,j}^*, U^*)$;

   (e) and finally:

$$\hat{e}(V^*, P) = \hat{e}(U^* + \sum_{j=1}^{n} \alpha_j^* Q_j^* + \sum_{j=1}^{k} m_j^* Q_{n+j}^* + f^* \sum_{j=1}^{n+k} Q_j^*, P_{pub})$$

5.2 Reductionist security proof

We should insist in the fact that keys are used only once!!!!

Now that the security model for our particular scheme is defined, we associate $SigNCode$ to the Computational Diffie-Hellman Problem which is assumed to be hard and the security proof consists in showing that if an adversary can break $SigNCode$ then one can efficiently solve CDHP. This approach is referred to as the provable security paradigm.

Therefore, in this section, we provide the proof of the following theorem:

Theorem 1 *If there exist an adversary $\mathcal{A}$ that has an advantage ϵ in forging our scheme in an attack modeled by the game that is previously described when running in a time t and asking q_H queries to random oracles H, q_E queries to the key extraction oracle and q_S queries to the signature oracle, then the CDH problem can be solved with an advantage $\epsilon' > (1 - \frac{1}{q_E})\epsilon$ and within time $t' < t + ((n + k)q_H + (n + k)q_E + nq_F(n + k + 3)q_S)t_m$ where t_m denotes the time to compute a scalar multiplication in G_1.*

Proof 1 *A challenger $\mathcal{C}$ is given aP, bP and tries to compute abP by simulating all possible oracles for the adversary $\mathcal{A}$. After receiving aP and bP, the challenger $\mathcal{C}$ sets $P_{pub} = aP$ and the adversary $\mathcal{A}$ can proceed to different queries at all phases of the proposed scheme.*

Queries: *$\mathcal{C}$ simulates all oracles and responds to each type of query as follows:*

- *H* ***queries****: when an identity ID and a message m is submitted to the H oracle, the challenger $\mathcal{C}$ flips a coin $c \in \{0, 1\}$ where $Pr[c = 0] = \frac{1}{q_E}$ and picks $x_j \in Z_q$ for each $1 \leq j \leq n + k$. Then, $\mathcal{C}$ returns the following:*
  - *$\{x_jP\}_{1\leq j\leq n+k}$ if $c = 0$;*
  - *$\{x_jbP\}_{1\leq j\leq n+k}$ if $c = 1$;*

  In both cases, $\mathcal{C}$ inserts a tuple $< ID, m, \{x_j\}_{1\leq j\leq n+k}, c >$ in a list $List_1$ to keep track of the way it answered the query.
- ***Key extraction queries****: when $\mathcal{A}$ requests the private key associated to an identity and a message m, $\mathcal{C}$ recovers the corresponding $< ID, m, \{x_j\}_{1\leq j\leq n+k}, c >$ from $List_1$. We assume that for any key extraction query or signature query involving an identity, a H oracle query was previously issued for the same identity and message.*
  - *if $c = 1$, $\mathcal{C}$ outputs "failure" and halts because it is unable to coherently answer the query;*
  - *if $c = 0$, $\mathcal{C}$ outputs $\{D_j = x_jP_{pub}\}_{1\leq j\leq n+k}$*

- F ***queries***: *$\mathcal{C}$ keeps a list $List_2$ for such queries. When a tuple (M, U) is submitted to the F oracle, $\mathcal{C}$ first scans the list $List_2$ to check whether F was already defined for that input. If it was, the previously defined value is returned. If it is the result of linear combinations of previous values than $\mathcal{C}$ returns the result of the corresponding linear combination. Otherwise, $\mathcal{C}$ picks a random $f \in Z_q$, stores the tuple (M, U, f) in the list $List_2$ and returns f to $\mathcal{A}$;*

- ***Signature queries***: *$\mathcal{C}$ keeps a third list $List_3$ for such queries. When $\mathcal{A}$ queries the signature oracle on a message M for an identity ID, $\mathcal{C}$ first recovers the previously defined $\{Q_j\}_{1 \leq j \leq n+k}$ from $List_1$. $\mathcal{C}$ looks at $List_2$ if f is already defined. If f is already defined, then $\mathcal{C}$ outputs "failure" and halts because it is unable to coherently answer the query. If it is not the case, then $\mathcal{C}$ checks $List_3$ and verifies if M is not a linear combination of existing queries. Otherwise, it picks a new $f \in Z_q$ and inserts the corresponding tuple in $List_2$. Then, if m is a linear combination of existing queries, $\mathcal{C}$ returns the result of the same linear combination on U values and V values. In all other cases, $\mathcal{C}$ picks $t \in Z_q^*$ and sets:*

  - $V = tP_{pub}$;
  - $U = tP - \sum_{j=1}^{n} \alpha_j Q_j - \sum_{j=1}^{k} m_j Q_{n+j} - f \sum_{j=1}^{n+k} Q_j$;

  The pair (U, V), is returned to $\mathcal{A}$ and appears as a valid signature from the latter's point of view since the verification is correct:

$$\hat{e}(V, P) = \hat{e}(U + \sum_{j=1}^{n} \alpha_j Q_j + \sum_{j=1}^{k} m_j Q_{n+j} + f \sum_{j=1}^{n+k} Q_j, P_{pub})$$

  $\mathcal{C}$ stores the tuple $< ID, m, t, U, V, f >$ in $List_3$.

Solving CDHP: *When $\mathcal{C}$ receives the valid tuple $(ID^*, M^*, f^*, r^*, U^*, V^*)$ from the adversary $\mathcal{A}$, it recovers abP as follows:*

- $P_{pub} = aP$;
- *recover $\{Q_j^*, x_j^*\}_{1 \leq j \leq n+k}\}$ corresponding to ID^*; if $c^* = 0$ then $\mathcal{C}$ halts and outputs failure. Otherwise, $\mathcal{C}$ proceeds to the computation phase;*
- *$\mathcal{C}$ computes* $U^* + \sum_{j=1}^{n} \alpha_j^* Q_j^* + \sum_{j=1}^{k} m_j^* Q_{n+j}^* + f^* \sum_{j=1}^{n+k} Q_j^*$
- *since the new signature is a valid one we have:*

$$\hat{e}(V^*, P) = \hat{e}(U^* + \sum_{j=1}^{n} \alpha_j^* Q_j^* + \sum_{j=1}^{k} m_j^* Q_{n+j}^* + f^* \sum_{j=1}^{n+k} Q_j^*, P_{pub})$$

We already know that $P_{pub} = aP$. Thanks to bilinearity we obtain:

$$\hat{e}(V^*, P) = \hat{e}(a(U^* + \sum_{j=1}^{n} \alpha_j^* Q_j^* + \sum_{j=1}^{k} m_j^* Q_{n+j}^* + f^* \sum_{j=1}^{n+k} Q_j^*), P)$$

We thus have:

$$\begin{aligned} V^* &= a(U^* + \sum_{j=1}^{n} \alpha_j^* Q_j^* + \sum_{j=1}^{k} m_j^* Q_{n+j}^* + f^* \sum_{j=1}^{n+k} Q_j^*) \\ &= a(r^* \sum_{j=1}^{n+k} x_j^* bP + \sum_{j=1}^{n} \alpha_j^* x_j^* bP + \sum_{j=1}^{k} m_j^* x_{n+j}^* bP \\ &\quad + f^* \sum_{j=1}^{n+k} x_j^* bP) \\ &= abP(r^* \sum_{j=1}^{n+k} x_j^* + \sum_{j=1}^{n} \alpha_j^* x_j^* + \sum_{j=1}^{k} m_j^* x_{n+j}^* \\ &\quad + f^* \sum_{j=1}^{n+k} x_j^*) \end{aligned}$$

From this equation, we can recover abP as follows:

$$abP = V^*((r^* + f^*) \sum_{j=1}^{n+k} x_j^* + \sum_{j=1}^{n} \alpha_j^* x_j^* + \sum_{j=1}^{k} m_j^* x_{n+j}^*)^{-1}$$

Consequently, the challenger $\mathcal{C}$ can resolve CDHP only if $\mathcal{A}$ was successful in his attack and if $c = 1$ for the corresponding (ID^*, M^*). Therefore $\mathcal{C}$ can resolve CDHP with probability at least $(1 - \frac{1}{q_E})\epsilon$. Finally, Theorem 1 is proved.

6 Related Work

The problem of the impact of pollution attacks on network coding applications was first analyzed in [8] whereby authors propose the use of homomorphic functions proposed in [9]. In the proposed solution, homomorphic hash functions are

computationally expensive operations and therefore authors propose to reduce this cost by providing a probabilistic verification. Moreover, as opposed to $SigNCode$, the paper only deals with homomorphic hash functions and not signatures. Therefore, the system relies on the existence of a preliminary phase whereby the source first hashes original blocks and signs each hash block with a common signature algorithm such as RSA [10]. Furthermore, $SigNCode$ does not require the communication of initial integrity values before starting the network coding operation as opposed to [8]. Indeed, the verification of messages are performed on-the-fly and rely on combination of signatures originating from the source sent together with the network coded messages. An intermediate node receiving a message m_1 only requires the identity ID of the source and the block identities $BID_{1,j}$ in order to verify the signature σ_1 and thus can conclude that this message is a result of legitimate linear combination of blocks originating from the source. Therefore there is a single communication channel and the source does not initially send integrity values of each individual block.

In [11], similarly to our scheme, authors propose a new homomorphic signature scheme that allows nodes to sign any linear combination of the incoming packets without contacting the source. Their scheme slightly modifies the Aggregate Signature version of the Boneh-Lynn-Shacham signature scheme [12] by replacing the hash value of a message by the message or a block of the message itself. Authors do not provide a security analysis of the result of this modification and claim that the security of their proposed scheme relies on the security of this particular signature. Therefore the security of such scheme is not convincing. Moreover, in order to implement this particular signature scheme, the source needs to distribute a large set of keys that are represented by distinct points of p-torsion elliptic curves. Since $SigNCode$ uses ID-based cryptographic scheme, intermediate nodes do not need to previously receive any information from the source before the reception of messages.

In [13], authors propose another approach to analyze the impact of bogus injection that is also defined as Byzantine errors. Indeed, authors consider an information-theoretically solution for such attacks. Their scheme provides a distributed solution that is proved to be rate-optimal and can be implemented in polynomial time. Authors consider different type of adversaries and adapt their solution for each type of adversaries. However, even if their solution is proved to be rate-optimal, it significantly increases the communication overhead due to the redundant messages or flows.

7 Conclusion

In this paper, we propose a new approach to the data integrity problem in network coding using a new homomorphic signature scheme derived from an ID-based signature. This scheme mainly addresses the new integrity problem akin to network coding whereby some modification on original data, namely linear com-

binations of authentic original blocks should be authorized whereas all other modification attempts including data tampering and injection of bogus messages should be detected. Classical integrity mechanisms definitely fall short of meeting such a subtle notion of integrity. Recent research results have taken a similar direction to our solution by looking for homomorphic integrity functions but they suffer from several shortcomings such as the need for the source to distribute all nodes hash values for each original data block prior to the network coding operation. Our scheme is definitely more efficient in that each linear combination can be verified solely based on the ID of the source, the value of the linear combination and the signature thereof. The second contribution of our paper is the security definition and the model for the reductionist proof that encompass the new notion of extended integrity. Our protocol is proven in the suggested model with the assumption of the Computational Diffie Hellman Problem.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions of Information Theory*, 2000.

[2] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Transactions of Information Theory*, 2003.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transaction on Networking*, 2003.

[4] S. Katti, H. Rahul, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding for wireless environments," in *Proceedings of ACM SIGCOMM*, 2006.

[5] D. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *CT-RSA*, 2002, pp. 244–262.

[6] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," in *Advances in Cryptology - Asiacrypt'01*, vol. LNCS 2139. Springer-Verlag, 2001, pp. 213–229.

[7] J. Cha and J. Cheon, "An identity based signature from gap diffie-hellman groups," in *Proceedings of PKC 2003*, vol. LNCS 2567. Springer-Verlag, 2003, pp. 18–30.

[8] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distributions," in *Proceedings of INFOCOM'06*, 2006.

[9] M. Krohn, M. Freedman, and D. Menezies, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proceedings of IEEE Symposium on Security and Privacy*, Berkeley, CA, 2004.

[10] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM, 21(2):120-126*, vol. 21(2), pp. 120–126, 1978.

[11] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proceedings of the 4th annual conference on Information Schemes and Systems*, Princeton, NJ, 2006.

[12] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology- Asiacrypt'01*, vol. LNCS 2729. Springer-Verlag, 2003, pp. 382–398.

[13] S. Jaggi, M. Landberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzatine adversaries," in *Proceedings of INFOCOM'07*, 2007.