Signatures for Multi-source Network Coding

László Czap and István Vajda

Budapest University of Technology and Economics
Laboratory of Cryptography and Systems Security (CrySys)
2, Magyar Tudósok krt., Budapest 1117, Hungary
{czap,vajda}@crysys.hu

Abstract. We consider the problem of securing inter-flow network coding with multiple sources. We present a practical homomorphic signature scheme that makes possible to verify network coded packets composed of data originating from different sources. The multi-source signature scheme allows to circumvent the need of a secret key shared by all sources. Our solution is an extension of the pairing based homomorphic signature scheme by Boneh et al. We prove the security of the extended scheme by showing a reduction to the single-source case. We evaluated the performance of required computations and our results imply that the solution is applicable in practice.

1 Introduction

Network coding allows intermediate nodes to perform encoding on data packets. This can increase throughput and improve robustness [1]. Usually, in systems applying inter-flow network coding, packets originating from multiple different sources are encoded together. The methods of linear network coding [2–4] and random linear network coding [5,6] can be extended for systems with more than one source nodes as well, however, in general the multi-source case is more difficult to handle [7]. In these systems, in order to exploit the benefits of network coding, linear combinations of packets from different sources need to be produced.

We consider such multi-source networks using network coding in an adversarial setting, where some adversarial nodes (maybe even source nodes) may aim to hinder data transmission by launching a pollution attack. This means that the adversary may insert such encoded packets that are not a valid linear combination of original packets. Classical methods of integrity protection (cryptographic hash functions, digital signatures) are not applicable here, because they are designed to protect packets from modification on their way from the source to the destination, while network coding not only routes but inevitably modifies packets.

Pollution attack has serious effect, because recombining a polluted packet also results in a polluted packet. As a consequence, if intermediate nodes are not aware of the pollution of an incoming packet, all of their downstream nodes may receive and disseminate polluted packets. Furthermore, by decoding, even a single polluted packet may destroy the whole original data at the receivers.

The problem of pollution attack in the multi-source setting is investigated in [8]. The authors show a construction in which the homomorphic hash value of data vectors are signed using oridinary digital signatures. It is shown, that in the general case, we can not hope to find more efficient solution. Here, we deal with a special case, where only packets having the same identifier are combined together. In this setting we can exploit the homomorphic property of the signature and use shorter signatures than [8].

The signature on raw packets are computed by the sources, each using its own private key. In this way sources do not need to share a secret key or use secure channels, encoded packets can be verified using the public keys of the involved source nodes. This property requires that a part of the public key of the sources is common. Our solution shows how to extend the homomorphic signature scheme presented in [9] to fulfill this property and to handle multiple sources while remaining secure. Although the computations rely on public key operations, our performance analysis show that the computational overhead can be tolerable for practical applications also.

The rest of this paper is organized as follows. Section 2 overviews the related work, Section 3 describes the applied system model including the model of the adversary. The multi-source signature scheme is described in detail in Section 4. In Section 5 we give a proof of security and in Section 6 the results of our performance analysis are given. In Section 7 we summarize the paper.

2 Related Work

Several cryptographic primitives were proposed with homomorphic property for the sake of verification of network coded packets. A homomorphic hash function is introduced in [10] and extended for network coding in [11]. Using homomorphic hash function for integrity protection is generally not practical, especially in the multi-source scenario, because it requires a secure channel between each of the sources and each of the intermediate nodes.

Based on the homomorphic hash function a homomorphic signature scheme was introduced in [12]. It exploits that the base RSA signature also has homomorphic property, thus signing the homomorphic hash value of a message results in a homomorphic signature scheme. However, the authors were unaware that homomorphism introduces a serious security flaw that does not arise using common cryptographic hash functions. The flaw was published in [13]. Further errors of this solution were also discovered [14].

A computationally less expensive method is also proposed in [12], and the same scheme is published in [15] also. A homomorphic signature scheme for content distribution is introduced in [16]. Another approach using bilinear mapping was introduced in [17]. All these schemes share the property that they are limited to a single source and that the number of messages that can be signed before changing the keys is restricted.

It is only the signature scheme by Boneh et al. in [9] based on bilinear mapping that is general, not flawed yet and capable of signing unlimited number

of messages. We take this scheme as a basis and extend it for the multi-source network coding scenario.

The multi-source case is considered in [8]. The most general setting is investigated, where vectors with arbitrary identifiers are allowed to combined together. Their construction applies homomorphic hashing with ordinary digital signatures. Compared to this work, our solution is more restricted, because we require that vectors combined together share a common identifier. However, this allows us to make the signatures shorter.

There are also information theoretic solutions against pollution attacks such as [18, 19], however they rely on adding redundancy at the source, hence they are not applicable when there are more than one sources. A scheme proposed for sensor networks [20] deals with multiple sources, but its field of applicability is constrained to special scenarios.

3 System model

We consider a network coding scenario with m unit-capacity source nodes and any number of receivers. We do not consider here the question of designing a code, we only assume that intermediate nodes perform some linear network coding over a finite field of size p. For simplicity, we further assume that sources fragment their data into equal size packets consisting of N symbols. A packet is considered as a vector $\mathbf{v}$ of dimension N. An id is assigned to each packet, such that packets that have to be encoded together are assigned with the same id.

We further require that a PKI is available in the system that makes possible all nodes to have their own private-public key pair. Source i signs its packet $\mathbf{v}_i$ using its own private key α_i. Intermediate nodes first verify the signature on the received signed packets, then produce linear combinations of the verified packets and compute a signature for the encoded packet using the received signatures without the need to access the private keys. A signed packet is a tuple $(id, \beta, \mathbf{y}, \sigma)$, where $\mathbf{y}$ is the encoded payload, β is the corresponding encoding vector: $\mathbf{y} = \sum_{i=1}^{m} \beta_i \mathbf{v}_i$, and σ is the signature on the encoded packet. A signed packet can be verified with the signature on it, even though its payload is combined from vectors originating from different sources. Section 4 gives the specific algorithms of signing, combining and verification of packets.

3.1 Adversary

We assume that the adversary controls one or more nodes of the network and aims to provide the receivers with false information by forging the signature scheme. The forgery is successful if any receiver decodes incorrectly, i.e. some of the decoded packets do not equal the original source packets. It can happen if the receiver accepts an encoded packet that is not a valid linear combination of original source packets. More formally, the attack is successful, if a forged signed packet $(id^*, \beta^*, \mathbf{y}^*, \sigma^*)$ passes the signature verification, while $\mathbf{y} \neq \mathbf{0}$ and either id^* is not a valid id used by the sources, or $\mathbf{y}^* \neq \sum_{i=1}^{m} \beta_i^* \mathbf{v}_i$.

It is important to note that contrary to the usual model of digital signatures we have to deal with adversarial source nodes as well. This gives the adversary another possibility for the forgery. In the single source case, an adversarial source node can influence only its own data. However, in the multi-source scenario data of honest nodes can also be corrupted by adversarial behavior, even if the adversarial source node produces valid signatures only. The following toy example illustrates the situation. We have two sources, from which the first behaves adversarial. Assume that two different paths are available from the adversary to the receiver. On the first path, the adversary transmits a signed packet with data $x_1 = 1$, while on the other path she sends $x_1' = 2$. The other source node is consistent, it sends $x_2 = 7$. It can happen that the receiver gets two valid signed packets corresponding to the following system of linear equations:

$$x_1 + x_2 = 8$$
$$2x_1' + x_2 = 11$$

If the receiver is not aware of the inconsistency of the adversarial source, the result of decoding is: $x_1 = 3$, $x_2 = 5$. This small example illustrates that the adversarial behavior of a source effects the decoding of data sent by honest sources also. This problem makes even more challenging to design signatures for multi-source network coding.

4 A multi-source homomorphic signature scheme

Before giving specific algorithms we define the special properties that a multi-source signature should have. Signing a raw packet does not have special requirements, we only need to define the properties of verification. Besides, one additional operation is also needed:

1. *Verification of encoded packets.* The verification must fail for an encoded packet that is not a valid linear combination of original source packets. A forged signed packet $(id^*, \beta^*, \mathbf{y}^*, \sigma^*)$ has to fail the signature verification, if $\mathbf{y} \neq \mathbf{0}$ and either id^* is not a valid id used by the sources, or $\mathbf{y} \neq \sum_{i=1}^{m} \beta_i \mathbf{v}_i$.
2. *Combining packets.* If all sources are honest, given a series of valid signed packets with the same id: $\{P_\vartheta = (id, \beta_\vartheta, \mathbf{y}_\vartheta, \sigma_\vartheta)\}$, $\vartheta = 1, 2, \ldots, k$, a node has to be able to produce a novel valid signed packet with any chosen encoding vector ϱ of length k without accessing any private keys. The resulting signed packet has to have the form $(id, \sum_{\vartheta=1}^{k} \varrho_\vartheta \beta_\vartheta, \sum_{\vartheta=1}^{k} \varrho_\vartheta \mathbf{y}_\vartheta, \sigma)$.

Below, we introduce a multi-source signature scheme with the desired properties based on the homomorphic signature scheme by Boneh et al. [9].

The signature scheme operates over a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi)$, where

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic multiplicative groups of the same prime order. The discrete logarithm problem is assumed to be computationally infeasible in these groups,

– $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable mapping with bilinear and non-degenerating property,
– $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism.

The setup of the system consists of the following steps:

1. Generate a bilinear group tuple $\mathcal{G}$. Let p denote the order of groups, p is prime, and $p > 2^\kappa$, where κ is the security parameter of the system.
2. Generate $N - m$ random numbers $g_1, g_2, \ldots, g_{N-m} \in \mathbb{G}_1 \setminus \{1\}$.
3. Choose a random number $u \in \mathbb{G}_2 \setminus \{1\}$.
4. Output the tuple of public system parameters, these are the public parameters that all source nodes share: $(\mathcal{G}, g_1, g_2, \ldots, g_{N-m}, u, H)$, where H is a known hash function.
5. Each source i sets up its own key pair. Source i selects randomly a private key $\alpha_i \in \mathbb{F}_p$. The public keys h_i of the sources are computed such that $u = h_i^{\alpha_i}$ holds for all i. The public key of the ith source is hence $PK_i = u^{(\alpha_i^{-1})} = h_i$. Computing the inverse of α_i and modulo exponentiation are all computationally feasible operations.
6. Output public keys.

Given this key setup, we show the different operations of the signature scheme.

4.1 Signing

Nodes can produce their signatures as if the single source scheme [9] was used, only the id has to be shared. Source i computes signature σ_i on vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,N}) \neq \mathbf{0}$:

$$\sigma_i = \left(\prod_{k=1}^{m} H(id||k)^{v_{i,N-m+k}} \prod_{j=1}^{N-m} g_j^{v_{i,j}} \right)^{\alpha_i}.$$

4.2 Combining

The signature σ of an encoded packet $(id, \beta, \mathbf{y}, \sigma)$ is produced by simply concatenating the signatures σ_i produced by the corresponding sources on $\mathbf{v}_i$, for which $\beta_i \neq 0$. It means that the encoded packet is verified by the tuple of signatures computed on the vectors that are encoded together. Formally, if m packets are encoded together, σ is an m length vector, and the ith element of the vector is σ_j, if β_j is the ith non-zero element of β.

At this point, the combining process has to check if all input packets that contain a signature for any $\mathbf{v}_i$ contain the same value of σ_i. If not, the combining process fails and an adversarial source is recognized.

4.3 Verification

The verification of a signed packet takes three steps as below:

1. Compute
$$\Gamma_1 = \gamma_1(id, \beta, \sigma) = \prod_{i=1}^{m} e(\sigma_i, h_i)^{\beta_i}.$$
2. Compute
$$\Gamma_2 = \gamma_2(id, \mathbf{y}) = \\ = e\left(\prod_{k=1}^{m} H(id||k)^{y_{N-m+k}} \prod_{j=1}^{N-m} g_j^{y_j}, u\right)$$
3. If $\Gamma_1 = \Gamma_2$ the verification succeeds, otherwise it fails.

We show that for correctly computed packets the verification eventually succeeds. To simplify the notations, let us introduce

$$\mathcal{H}(\mathbf{v}_i, id) = \prod_{k=1}^{m} H(id||k)^{v_{i,N-m+k}} \prod_{j=1}^{N-m} g_j^{v_{i,j}}.$$

$\mathcal{H}(,)$ is essentially a homomorphic hash function, for which $\mathcal{H}\left(\sum_{i=1}^{m} \beta_i \mathbf{v}_i, id\right) = \prod_{i=1}^{m} \mathcal{H}(\mathbf{v}_i, id)^{\beta_i}$. Let $n = N - m$.

Given this,

$$\begin{aligned}
\Gamma_1 = \gamma_1(id, \beta, \sigma) = \prod_{i=1}^{m} e(\sigma_i, h_i)^{\beta_i} = \\
= \prod_{i=1}^{m} e\left(\mathcal{H}(\mathbf{v}_i, id)^{\alpha_i}, h_i\right)^{\beta_i} = \prod_{i=1}^{m} e\left(\mathcal{H}(\mathbf{v}_i, id)^{\beta_i}, h_i^{\alpha_i}\right) = \\
= \prod_{i=1}^{m} e\left(\mathcal{H}(\mathbf{v}_i, id)^{\beta_i}, u\right) = e\left(\prod_{i=1}^{m} \mathcal{H}(\mathbf{v}_i, id)^{\beta_i}, u\right) = \\
= e\left(\mathcal{H}\left(\sum_{i=1}^{m} \beta_i \mathbf{v}_i, id\right), u\right) = e(\mathcal{H}(\mathbf{y}, id), u) = \\
= \gamma_2(id, \mathbf{y}) = \Gamma_2
\end{aligned}$$

We exploited the bilinear property of e, i.e. $e(x_1^{y_1}, x_2^{y_2}) = e(x_1, x_2)^{y_1 y_2}$, and the fact that $h_i^{\alpha_i} = u$.

4.4 Batch verification

The presented multi-source signature scheme supports batch verification as homomorphic signature schemes usually do. The verifier computes a random linear

combination from the signed encoded packets in the batch, resulting in a single signed encoded packet as if it computed a combined packet for forwarding. If the verification for this combined packet succeeds, all signed encoded packets are accepted as valid. If the verification fails, there is eventually at least one polluted packet in the batch. To find out which one, the verifier needs further verifications, e.g. it can use binary-checking, that tests the half of the batch in each step.

If there are some inconsistent packets in the batch, the batch verification fails, because the combining process will not be able to produce the valid signed packet that contains the random linear combination of packets in the batch. This is the property that makes it possible to handle adversarial sources. From this it follows that batch verification is not only an option to improve the performance of the verification process, but it is essential for the receiver to perform a batch verification on the received set of packets before decoding in order to be aware of adversarial source nodes.

It is an important practical advantage of our scheme that it not only recognizes inconsistency, but also provides an evidence against the adversary, because the verifier can find two encoded packets with two different valid signatures from the adversary. These inconsistent packets provide eligible proof against the adversary. Based on this, e.g. the certificate authority of the system can revoke the key pair of the malicious node. This property ensures that an adversary may reduce the performance of the system by inconsistent behavior for a while, but she can not do so continuously.

5 Security

To prove the security of the presented signature scheme we show a reduction to the single-source case. If there is only a single source, our scheme is almost identical to the network coding signature scheme $\mathbf{NCS_1}$in [9]. Theorem 6 of [9] proves that the probability of a successful forgery is negligible in κ, assuming the co-CDH problem is infeasible in $(\mathbb{G}_1, \mathbb{G}_2)$. Based on this result it is enough to show a reduction of the multi-source signature scheme to $\mathbf{NCS_1}$to prove computational security.

5.1 The $\mathbf{NCS_1}$scheme

Before giving the proof of security we briefly describe $\mathbf{NCS_1}$. It uses the same public parameters as the multi-source scheme, the only difference is that there is only one source with a key-pair (α, h). The source has m vectors to send and it signs these vectors with its private key. The computation of this signature σ'_i on a vector $\mathbf{v}_i$ is the same as in the multi-source case, i.e.

$$\sigma'_i = \mathcal{H}\left(\mathbf{v}_i, id\right)^{\alpha}.$$

Computing the signature of an encoded packet differs and is as follows in the case of $\mathbf{NCS_1}$:

$$\sigma' = \prod_{k=1}^{m} (\sigma'_i)^{\beta_k}.$$

The verification process is also similar, but the computation of Γ'_1 differs:

$$\Gamma'_1 = \gamma'_1(id, \beta, \sigma') = e(\sigma', h).$$

The computation of Γ'_2 is the same as Γ_2.

5.2 Proof of security

Theorem 1. *The presented multi-source signature scheme is secure, if the network coding signature scheme* ($\mathbf{NCS_1}$) *in [9] is secure.*

Proof. We show that if there exists an efficient algorithm $\mathcal{A}^*$ that can produce a forged signed packet in the multi-source signature scheme, we can construct an efficient algorithm $\mathcal{A}$ that produces a forged signed packet in $\mathbf{NCS_1}$.

The input of algorithm $\mathcal{A}^*$ is the tuple of public system parameters as well as the public key of all the m sources together with a valid identifier id, and the set of corresponding signed packets from the sources. The signed packet from the ith source is: $(id, \mathbf{e}_i, \mathbf{v}_i, \sigma_i)$, where $\mathbf{e}_i$ is the ith unit vector. Algorithm $\mathcal{A}^*$ outputs a packet $(id^*, \beta^*, \mathbf{y}^*, \sigma^*)$ for which $\mathbf{y}^* \neq \mathbf{0}$ and the verification succeeds, but either $id^* \neq id$ or $id^* = id$ and $\mathbf{y}^* \neq \sum_{i=1}^{m} \beta_i^* \mathbf{v}_i$.

Let us assume that a source node using the $\mathbf{NCS_1}$scheme is given with the system parameters $(\mathcal{G}, g_1, g_2, \ldots, g_{N-m}, u, H)$ and a key pair (h, α), for which $h^\alpha = u$. (Here m means the dimension of the subspace the source signs.)

Algorithm $\mathcal{A}$ operates as follows:

1. Construct the multi-source signature scheme with m sources. The key pair of the first source is $(h_1 = h, \alpha_1 = \alpha)$, and this single-source key setup is extended with $m-1$ additional key pairs in the following way. We choose $m-1$ random numbers from $\mathbb{F}_p$: $x_1, x_2, \ldots, x_{m-1}$. The public key of the ith source is $h_i = h^{x_{i-1}}$ for $i > 1$. The corresponding private keys are $\alpha_i = x_{i-1}^{-1}\alpha$ for $i > 1$.
2. Call algorithm $\mathcal{A}^*$ with the parameters constructed above. If required, a signed packet from the ith source can be constructed from a single-source signed packet, although private keys α_i are unknown: if $(id, \mathbf{e}_i, \mathbf{v}_i, \sigma'_i)$ is the corresponding single-source packet we compute a packet as if it came from the ith source: $(id, \mathbf{e}_i, \mathbf{v}_i, \sigma_i = (\sigma'_i)^{x_{i-1}^{-1}})$.
   The output of $\mathcal{A}^*$ is a forged signed packet in the multi-source scheme: $(id^*, \beta^*, \mathbf{y}^* = \sum_{i=1}^{m} \beta_i^* \mathbf{v}_i^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_m^*))$. (For simplicity, we assume that all sources are involved, but the proof does not rely on this.)
3. Output the packet $(id^*, \beta^*, \mathbf{y}^*, \sigma^{**})$, where $\sigma^{**} = (\sigma_1^*)^{\beta_1^*} \prod_{i=2}^{m} (\sigma_i^*)^{\beta_i^* x_{i-1}}$.

We show that the output of $\mathcal{A}$ is a forged signed packet in $\mathbf{NCS_1}$. First, we show that it passes the verification process than prove that the packet is forged.

$$\Gamma_1' = e\left((\sigma_1^*)^{\beta_1^*}\prod_{i=2}^{m}(\sigma_i^*)^{\beta_i^* x_{i-1}}, h\right) = e\left((\sigma_1^*)^{\beta_1^*}, h_1\right)\prod_{i=2}^{m} e\left((\sigma_i^*)^{\beta_i^* x_{i-1}}, h_1\right) = \quad (1)$$

$$= e(\sigma_1^*, h_1)^{\beta_1^*}\prod_{i=2}^{m} e\left(\sigma_i^*, h_1^{x_{i-1}}\right)^{\beta_i^*} = \quad (2)$$

$$= \prod_{i=1}^{m} e(\sigma_i^*, h_i)^{\beta_i^*} = e\left(\mathcal{H}\left(\sum_{i=1}^{m}\beta_i^*\mathbf{v}_i^*, id\right), u\right) = \quad (3)$$

$$= e\left(\mathcal{H}\left(\mathbf{y}^*, id\right), u\right) = \Gamma_2' \quad (4)$$

Equation (3) holds because the output of $\mathcal{A}^*$ passes the multi-source verification. It is important to note that $\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_m^*$ are not known and may not unique, but they eventually exist. This equation proves that the packet passes the verification of $\mathbf{NCS_1}$.

Now we have to show, that this packet is a forged signed packet in $\mathbf{NCS_1}$. The packet is forged if $\mathbf{y}^* \neq \mathbf{0}$ and either $id^* \neq id$, or $\mathbf{y}^* \neq \sum_{i=1}^{N}\beta_i^*\mathbf{v}_i$. According to the properties of $\mathcal{A}^*$ this condition is satisfied, consequently, the output of $\mathcal{A}$ is a forged packet in $\mathbf{NCS_1}$. □

Theorem 1 and Theorem 6 of [9] together prove that the multi-source signature scheme is computationally secure.

6 Performance results

Like all known public key cryptographic operations, the presented scheme also involves computationally expensive operations. Let us overview the number of operations the different algorithms require. Here we ignore the cost of network coding operations and focus only on the overhead caused by the signature scheme.

- Signing a raw packet requires N exponentiations and $N-1$ multiplications.
- Combining is extremely simple, signatures of the input packets need only be put one after another. Besides, the process only needs to check if signatures corresponding to the same data vector equal.
- Verification is computationally more expensive, if m packets are combined together, computing Γ_1 requires m mappings, m exponentiations and $m-1$ multiplications, while Γ_2 requires N modulo exponentiations, $N-1$ multiplications and one mapping.

Of course, using signatures has not only computational, but communication overhead as well. In our case, the length of a signature depends on the size of $\mathbb{G}_1$ and on the number of source packets combined together in the encoded packet. It follows from the operation of the signature scheme that the amount of overhead is proportional to the number of packets combined.

In contrary to usual signatures, in the case of network coding, the verification process has to be performed not only by the receiver, but intermediate nodes also, before they produce a novel combination. This property makes verification performance a key issue of a homomorphic signature scheme. Note that in this extent the multi-source setting does not differ from the single-source case.

In order to provide also numerical results, we implemented the signature scheme and measured the performance of the signing and verification operations as well as the communication overhead in different cases. We omitted measuring the performance of the combining process, because its overhead is negligible. Our implementation was written in C using the Pairing-Based Cryptography Library (libpbc) [21]. For the computations we used the curve groups implemented in the libpbc library. The main properties of different tested parameters is summarized on Table 1. For further details on different types of pairings we refer the reader to [21].

Type	**Base field (bits)**	**Dlog security (bits)**	**Signature length (bytes)**
a	512	1024	128
f	160	1920	40
d159	159	954	40

Table 1. Main properties of tested pairings

We performed measurements in two different settings that differs in the size of signed data packet. In the first scenario the length of a packet was set to 1500 bytes, that is the length of a typical IP packet. In the second case, larger data blocks were signed, the packet length was 1 MB. In both cases 100 packets were generated with random elements. Different key pairs were generated for every packet, and raw packets were signed using their corresponding private key. Using the signed raw packets, we produced signed encoded packets. The average number of combined packets Δ was set in advance, then 100 combined packets were produced. The number of packets to combine together was selected randomly from the range $[1; 2\Delta - 1]$ for each combined packet, where the coefficients and the packets to combine were also selected randomly. Afterward, the produced signed combined packets were verified one by one. We measured the throughput of the signing and the verification process not counting the size of appended signatures. The computations were run on PC with 2.5 GHz CPU frequency and 2 GB of RAM, using Linux operating system.

The performance results of the signing operation is presented on Table 2, while the throughput results of the verification process are plotted on Fig. 1 and 2 corresponding to the two different packet lengths tested.

The result heavily depends on both the packet length and the selected type of pairing. The verification is clearly more costly than signing. In both cases

Packet length	Pairing Type	Throughput	% overhead
1500 bytes	a	84.2 MB/s	8.5
	f	107.2 MB/s	2.6
	d159	99.0 MB/s	2.6
1 MB	a	141.8 MB/s	0.0128
	f	149.2 MB/s	0.004
	d159	130.6 MB/s	0.004

Table 2. Throughput of the signing process

the larger packet size results better performance. For signing the improvement is $\sim 60\%$ for Type 'a' pairing. For the verification, the difference is much more significant (note that the scaling of vertical axes differ on the two figures). It is because computing the pairing is the most costly operation, and if the packet size is smaller, the pairing function is used more often.

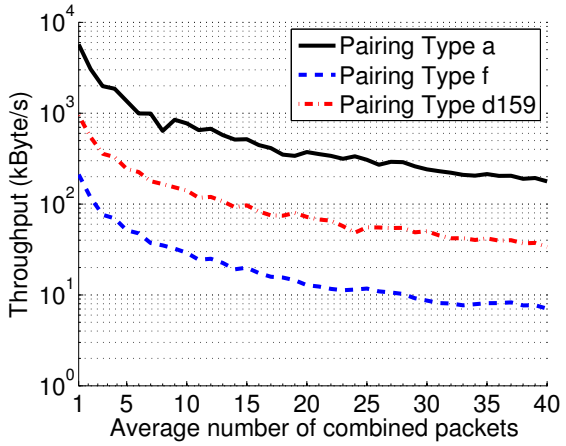

Fig. 1. Throughput results of the verification process, packet size is 1500 Bytes

While by signature generation the different types of pairing give similar result - the difference is at most 27%, see 3rd column of Table 2 -, in the signature verification almost one order of magnitude difference rises between them. There is also a considerable difference in the degree of performance degradation as the number of packets combined together increases. In this aspect type 'a' pairing gives the best performance, in the case of the larger packet size, there is only a slight difference between its throughput results.

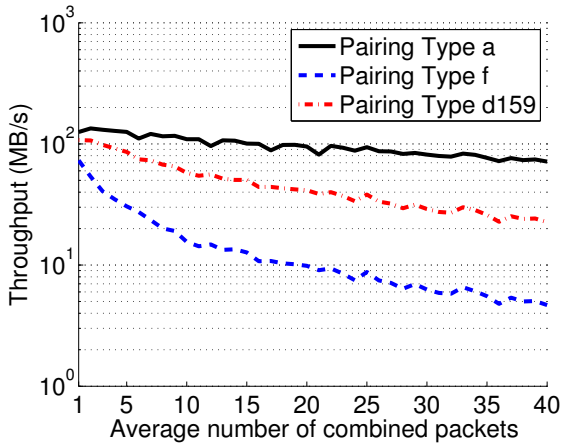

Fig. 2. Throughput results of the verification process, packet size is 1 MB

Regarding communication overhead, it is quite obvious that the larger is the packet size the lower is the overhead of one signature. In the case of an encoded packet, this overhead grows proportionally with the number of packets involved.

The conclusion of our investigation is that both the packet size and the applied pairing have to be selected carefully considering the performance requirements of the system. The cost of verification is the most important parameter, because verification is performed much more often than signing. Thus, from the tested pairings, type 'a' pairing seems to be more attractive than others, despite the larger signature size and the slightly slower signing operation.

Our results show, that although costly public key cryptographic operations are involved, the performance of the verification process can be in the order of several tens of MB/s, thus it can be practical for many applications, even if the computational power of an intermediate router is more constrained than that of the desktop computer on which we ran our measurements.

7 Summary

We proposed a homomorphic signature scheme that provides a cryptographic solution against pollution attacks for systems using inter-flow network coding with multiple sources. A combined signed packet can be verified using the combined signature and the public keys of the corresponding sources. The scheme is an extension of the signature scheme in [9]. We provide a proof for the security of the novel scheme based on the security of the single-source case. Our scheme can handle even the case when some source nodes are adversarial, moreover, it can not only detect their attack but it provides a proof against the adversary.

We implemented and measured the performance of the signing and verification process. The results imply that the scheme is suitable for practical purposes.

References

1. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network information flow. IEEE Transactions on Information Theory **46**(4) (July 2000) 1204–1216
2. Li, S.Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Transactions on Information Theory **49**(2) (2003) 371–381
3. Kötter, R., Médard, M.: An algebraic approach to network coding. IEEE/ACM Transactions on Networking **11** (October 2003) 782–795
4. Fragouli, C., Soljanin, E.: Network Coding Fundamentals. Volume 2. Foundations and Trends in Networking (2007)
5. Ho, T., Kötter, R., Médard, M., Karger, D.R., Effros, M.: The benefits of coding over routing in a randomized setting. In: Proceedings of the IEEE Information Theory Symposium (ISIT). (June 2003)
6. Ho, T., Médard, M., Kötter, R., Karger, D.R., Effros, M., Leong, B., Shi, J.: A random linear network coding approach to multicast. IEEE Transactions on Information Theory **52**(10) (2006) 4413–4430
7. Yeung, R.W.: Information Theory and Network Coding. Information Technology: Transmission, Processing and Storage. Springer (2008)
8. Boneh, D., Agrawal, S., Boyen, X., Freeman, D.: Preventing pollution attacks in multi-source network coding. In: Proceedings of PKC. (2010)
9. Boneh, D., Freeman, D., Katz, J., Waters, B. In: Signing a Linear Subspace: Signature Schemes for Network Coding. Volume 5443/2009. Springer (March 2009) 68–87
10. Krohn, M.N., Freedman, M.J., Mazieres, D.: On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proceedings of 2004 IEEE Symposium on Security and Privacy. (2004)
11. Gkantsidis, C., Rodriguez, P.: Cooperative security for network coding file distribution. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). (2006)
12. Yu, Z., Wei, Y., Ramkumar, B., Guan, Y.: An efficient signature-based scheme for securing network coding against pollution attacks. In: Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM). (2008)
13. Wang, Y.: Insecure "provable secure network coding" (October 2009)
14. Yun, A., Cheon, J., Kim, Y.: On Homomorphic Signatures for Network Coding. IEEE Transactions on Computers (2009)
15. Kang, H.J., Vasserman, E.Y., Lee, H.T., Cheon, J.H., Kim, Y.: Secure network coding for a P2P system. ACM Conference on Computer and Communications Security, Poster (2009)
16. Zhao, F., Kalker, T., Médard, M., Han, K.J.: Signatures for content distribution with network coding. In: Proceedings of 2007 IEEE International Symposium on Information Theory (ISIT '07). (June 2007)
17. Lauter, K.E., Charles X, D., Jain, K.: Signatures for network coding. In: Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS '06). (March 2006)
18. Jaggi, S., Sanders, P., Chou, P.A., Effros, M., Egner, S., Jain, K., Tolhuizen, L.: Polynomial time algorithms for multicast network code construction. IEEE Transactions on Information Theory **51**(6) (June 2005) 1973–1982
19. Ho, T., Leong, B., Kötter, R., Médard, M., Effros, M., Karger, D.R.: Byzantine modification detection in multicast networks using randomized network coding. In: Proceedings of the 2004 IEEE International Symposium on Information Theory (ISIT). (June 2004)

20. Buttyán, L., Czap, L., Vajda, I.: Securing coding based distributed storage in wireless sensor networks. In: Proceedings of the IEEE Workshop on Wireless and Sensor Network Security (WSNS), Atlanta, USA (2008)
21. : The pairing-based cryptography library. http://crypto.stanford.edu/pbc/