A Data Integrity Scheme Based on Homomorphic Hash Function for Multi-source Network Coding

Niu Shu-fen, Wang Cai-fen and Cao Su-zhen

Abstract

The integrity scheme for a single source based on homomorphic signature cannot handle a combined message's signature from multiple sources with different private keys. The main reason is that the signature schemes will not hold the homomorphism if the unique secret key is replaced by distinct private keys. This also means the forwarding nodes could not generate a valid signature for a combined message without knowing the source keys.

In this paper, taking advantage of vector Merge algorithm and homomorphic hash function, we propose an efficient data integrity scheme for multi-source securing network coding against pollution attacks. Firstly, each source node computes raw massage's hash values and uses a secure mechanism to sign the hash values, and then appends the hash values and its signatures to each message sending to forwarding nodes and sink nodes. The forwarder can verify the integrity of network coded data from different source nodes without knowing the sources private keys and generate the hash for the combined messages. The security of the scheme relies on the Discrete Logarithm problem and Co-Diffie-Hellman problem.

Keywords: Multi-source network coding, Data integrity, Aggregate Signatures, Homomorphic Hash Function

1. Introduction

Network coding was first introduced in [1] as an alternative to the traditional routing networks, and it has been shown to improve the capacity and achieve the optimal throughput in network [2],[3]. Furthermore, network coding can reduce the amount of transmissions in wireless networks[4],[5]. However, network coding may face potential pollution attack threats; if some routers in the network are malicious and forward invalid combinations of received packets, then these invalid packets get mixed with valid packets downstream and quickly pollute the whole network. Prior works for intermediate nodes verifying the validity of incoming vectors are based on one of two primitives: (collision-resistant) homomorphic hash functions [8][10] or homomorphic signature schemes[11][12][13]. In both cases, the homomorphic properties are used such that the signature (or hashing) operation on a linear combination of vectors results in a corresponding homomorphic combination of signatures (or hash values).

For multi-source networks using network coding, signature scheme is considerably harder than the single source problem. In this systems, packets originating from multiple different sources are encoded together, and the homomorphic signature schemes will not hold the homomorphism if the unique secret key is replaced by distinct private keys, which means that the forwarding nodes could not generate a valid homomorphic signature for a combined message without knowing the source keys.

The remainder of this paper is organized as follows. In Section 2, we briefly discuss related work, and we discuss the novelty and contributions of this paper in Section 3. Then we introduce complexity assumptions and cryptographic primitives used in our approach in Section 4. The detailed design and security analysis of our scheme are presented in Section 5 and 6. Finally, we evaluate the performance of the proposed scheme in Section 7,and conclude this paper in Section 8.

2. Related Work

Krohn [10] first presented a homomorphic hashing scheme, which allows a verifier to verify the integrity of rateless erase codes. This approach was extended by Gkantsidis and Rodriguez (denotes as GR's scheme)[15] for securing the file distribution systems built on top of network coding technique. However, one drawback of GR's scheme is that it needs an extra secure channel to broadcast the source hashes to all the forwarders.

Charles (denotes as CJL's scheme)[12] defined another similar model based on homomorphic signature scheme over elliptic curves. However, the pair-based operations used by CJL's scheme are much slower than the modular exponentiation operations adopted in GR's schemes.

Corresponding Author:
(*E-mail: {sfniu76,wangcf, suzcao}nwnu.edu.cn*)
College of Computer Science and Engineering,Northwest Normal University Northwest Normal University, Lanzhou, Gansu , 730070, P.R.C

Gennaro [14] proposed a homomorphic signature based on the RSA assumption and showed how to work with small coeffcients over the integers in networks of bounded size.

Catalano [16] introduced two new network coding signature schemes. Both of the schemes are provably secure in the standard model. Not only could the above existent signature schemes be used in single source network coding system, but they are limited to be applied to multi-source network coding system which is more prevalent in networks.

Recently, Agrawal et al. [19] and [18] proposed their schemes to defend against pollution attacks in multi-source network coding. Agrawal et al.[19] introduced a merge algorithm into their work to generate the public keys and signatures at intermediate nodes. Laszlo's work [20] is built on bilinear pairing, and the way they sign the packets is similar to Jonathan's [11] method. Both [19] and [18], however, have a common drawback that the size of signature grows linearly with the number of the sources. If a packet is mixed by l original packets, then the length of the signature on this packet is l times the signature length in single source network coding. That is unpractical.

Shao [20] proposed a new network coding signature scheme to be proven-secure in the standard model. In Yang's work[21], the intermediate nodes therein can verify the integrity and origin of the encoded messages received without having to decode them, and the receiver nodes can check them out and discard the messages that fail the verification. By this way, the pollution is canceled out before reaching the destinations.

3. Our Contribution

In the paper, we present a new integrity approach which is different from an existing scheme, because of taking the Merge algorithm [19] as a basis and exploiting homomorphic hash function [10], and the scheme overcome drawback that the homomorphism can not hold if the unique secret key is replaced by distinct private keys in multiple-source network coding. The scheme works as follows : First, each source node computes raw massage's hash values and uses a secure mechanism to sign the hash values, and then appends the hash values, and its signature to each message sends to intermediate nodes and sink nodes. The forwarder can verify the integrity of network coded data from different source nodes without knowing the sources private keys, and generate the hash for the combined messages.

Our security proof shows that creating a valid hash for a vector outside the linear span of the original message vectors is difficult, and forging a signature for hash is at least as hard as solving the Co-Gap Diffie-Hellman problem. The proposed scheme also does not need any extra secure channel to broadcast the source hashes to all the forwarders. Computational results show that the algorithm outperforms the Laszlo's scheme in verification process and the whole runtime of the algorithm when source nodes $m > 300$.

4. Definitions and Preliminaries

In this section, we first introduce bilinear maps and several complexity assumptions. Then we briefly describe several cryptographic primitives we use in this paper.

4.1 Bilinear Map

Let G_1, G_2 be multiplicative cyclic groups of prime order p; let g_1, g_2 be generators of G_1. A bilinear map is a map $e: G_1 \times G_1 \to G_T$ with the following properties:

- *Computability*: There exists an efficiently computable algorithm for computing map e.
- *Bilinearity*: For all $u, v \in G_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(a,b)^{ab}$.
- *Non-degeneracy*: $e(g_1, g_2) = 1$

4.2. Complexity Assumptions

Definition 1 *Discrete Logarithm Problem* For $a \in Z_p$, given $g, h = g^a \in G_1$, output a.

The Discrete Logarithm assumption holds in G_1 if no t- time algorithm has advantage at least ε in solving the Discrete Logarithm problem in G_1, which means it is computational infeasible to solve the Discrete Logarithm problem in G_1.

Definition 2 *Computational Co-Diffie-Hellman* For $a \in Z_p$, given $g_2, g_2^a \in G_2$ and $h \in G_1$ compute $h^a \in G_1$.

The co-CDH assumption holds in G_1 and G_2 if no t-time algorithm has advantage at least ε in solving the co-CDH problem in G_1 and G_2.

4.3. Homomorphic vector Hash Function (VH)

A homomorphic hash function was introduced in [10] and extended for network coding in [15]. Let G be a cyclic group of prime order p in which the discrete logarithm problem is hard, and the public parameters contain a description of G and d random generators $g_1, g_2, ..., g_d \in G$ Then a homomorphic

hashing on message $\boldsymbol{v} = (v_1, v_2, ..., v_d) \in Z_p$ can be computed as: $H(\boldsymbol{v}) = \underline{\underline{def}} \prod_{j=1}^{d} g_j^{v_j}$ .It is easy to verify that the hashing functions satisfy following properties:

- *Homomorphism* For any two messages scalars m_1, m_2 and $\boldsymbol{w_1}, \boldsymbol{w_2}$, it holds that $H(w_1\boldsymbol{m}_1 + w_2\boldsymbol{m}_2) = H(\boldsymbol{m}_1)^{w_1} H(\boldsymbol{m}_2)^{w_2}$.
- *Collision Resistance* There is no probabilistic polynomial-time (PPT) adversary capable of forging $\boldsymbol{m}_3$ satisfying both $\boldsymbol{m}_3 \neq w_1\boldsymbol{m}_1 + w_2\boldsymbol{m}_2$ and $H(\boldsymbol{m}_3) = H(\boldsymbol{m}_1)^{w_1} H(\boldsymbol{m}_2)^{w_2}$.

***Theorem** 1* The homomorphic hashing functions is secure assuming the discrete logarithm problem in G is hard.

4.4. Multi-source Network Coding

We model multi-source network as a directed graph (V, E) consisting of a set of nodes V and a set of edges E . There are d source nodes $s = (s_1, s_2, ..., s_d) \subset V$ and they wish to send a set of messages to a set of destination nodes $T \subset V$ .Before transmission, the source $S_k (k = 1, ..., d)$ split the data F into a set of m vectors $\boldsymbol{v}_i^{(k)} \in F_p^n \ (i = 1, ..., m)$ each source node then appends a unit vector of length m to the vectors $\boldsymbol{v}_i^{(k)}$ to create m augmented vectors $\tilde{\boldsymbol{v}}_i^{(k)}$,given by

$$\tilde{\boldsymbol{v}}_i^{(k)} = (-v_i^{(k)} - \underbrace{\overbrace{0, 0, ..., 1}^{m}}_{i}, 0, 0, ..., 0) \in F_p^{n+m} \quad ,$$

where $k = 1, ..., d; i = 1, ..., m$. i.e., each original vector $\boldsymbol{v}_i^{(k)}$ is prepended by the vector of length m containing a single '1' in the ith position.These augmented vectors are then sent by the source as packets in the network.

Each forwarding node in the network processes packets as follows. Upon receiving packets $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_l$ on its l incoming communication edges, a node computes the vector $\boldsymbol{y} = \sum_{i=1}^{l} \beta_i \boldsymbol{w}_i$, where each $\beta_i \in F_p$. The resulting vector $\boldsymbol{y}$ is then transmitted on the node's outgoing edges. That is, each forwarding node transmits a linear combination of the packets it receives.

5. Homomorphic Hash-based Integrity Scheme

The proposed integrity scheme is based on homomorphic hash cryptography. For a raw message $\boldsymbol{v}_i$, source nodes S_i computes raw hash value $h_i = H(\boldsymbol{v}_i)$, and uses a mechanism to sign h_i , in a way that allows verifiers to verify the authenticity of h_i . When a verifier receives an linear combination message $\boldsymbol{y} = \sum_{i=1}^{l} \boldsymbol{v}_i w_i$ along with m pairs of (raw hashing value, weight) (h_i, w_i) , it first determines whether the hashing values are valid, and then verifies the message's integrity by checking whether:

$$\prod_{i=1}^{m} h_i^{w_i} = H(\boldsymbol{y})$$

We also exploit aggregate signatures scheme [17](denoted as BGL's scheme)to authenticate h_i . The B GL'S scheme base on bilinear maps provides general aggregation, where anyone can combine signatures into an aggregate at any time without the cooperation of the signers.

Agrawal et al [19] proposed an algorithm *Merge* that merges the lists of identifiers contained in aggregate vectors and adjusts the vectors' augmentations. The algorithm does not itself linearly combine vectors, but rather it prepares aggregate vectors to be mixed together. For our multi-source network coding, we have

$$\tilde{\boldsymbol{v}}_i^{(k)} = (-v_i^{(k)} - a_i^{(k)}) \in F_p^{n+m}$$

Where $a_i^{(k)}$ denotes augmentation components. If the vectors come from different files and we must introduce additional augmentation before we can linearly combine the vectors. In this case we define:

$$\tilde{\boldsymbol{v}}_i^{(1)} = (-\boldsymbol{v}_i^{(1)} - \boldsymbol{a}_i^{(1)}, \overbrace{\boldsymbol{0}, \boldsymbol{0}, ..., \boldsymbol{0}}^{d-1}) \in F_p^{n+dm}$$

$$\tilde{\boldsymbol{v}}_i^{(2)} = (-\boldsymbol{v}_i^{(2)} -, \boldsymbol{0}, \boldsymbol{a}_i^{(1)}, \overbrace{\boldsymbol{0}, \boldsymbol{0}, ..., \boldsymbol{0}}^{d-1}) \in F_p^{n+dm}$$

$$\cdots$$

$$\tilde{\boldsymbol{v}}_i^{(d)} = (-\boldsymbol{v}_i^{(d)} -, \overbrace{\boldsymbol{0}, \boldsymbol{0}, ..., \boldsymbol{0}}^{d-1}, \boldsymbol{a}_i^{(d)}) \in F_p^{n+dm}$$

Where $\boldsymbol{0}$ denotes a length m zero vector. We assume forwarding node receives c message from different source nodes, and L denotes a c element subset of $[1, n]$. Forwarding node divides set L into d subset $L_1, L_2, ..., L_d$, where L_i is the subset of receiving message from source node S_i . And the number of message in subset L_i is c_i , and we have

$c = \sum_{i=1}^{d} c_i, L = L_1 \cup L_2 \cdots L_d$.

Given set of weights of every source nodes $\beta_{ij} (i = 1,2,...,d; j \in L_i)$ and c vectors from different source nodes, where $c = \sum_{i=1}^{d} c_i$, we have:

$$\boldsymbol{y} = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{\boldsymbol{v}}_j$$

$$= (y_1, y_2, ..., y_n, ..., y_{n+m}, ..., y_{n+dm})$$

The data components are mixed together, but the augmentation coefficients remain separate.

There are three parties in the system: the sources, the forwarders, and the sinks; each source is identified by an identifier. The basic scheme mainly consists of five algorithms: *Setup*, *KeyGen*, *Sign*, *Combine*, *Verifying* . Let NS = (*Setup*, *KeyGen*, *Sign*, *Combine*, *Verifying*) be a signature scheme, and our network coding signature scheme NS is as follows:

- ***Setup*** : Given a security parameter 1^k, the system does:
  1). Generate bilinear group tuple (G_1, G_2, G_T, e), where G_1, G_2, G_T have prime order p that efficiently support a bilinear mapping $e: G_1 \times G_2 \to G_T$ .Choose generators $g_i \leftarrow G_1 \setminus \{1\}$ for $i = 1,2,...,n$ and $g \leftarrow G_2 \setminus \{1\}$.
  2). Let $H_1 : \{0,1\}^* \to G_1$ be a hash function.
- *KeyGen*: The source S_i picks random $x_i \in Z_p$ and computes $p_i = g^{x_i}$. The source's public key is p_i , and private key is $x_i \in Z_p$.
- ***Sign***: The source S_i does :
  1). Computes $h_i = H(\boldsymbol{v}_i) = \prod_{j=1}^{n} g_j^{v_{ij}}$.
  2). Computes $t_i = H_1(h_i, id) \in G_1$.
  3). Computes $\sigma_i = t_i^{x_i} \in G_1$.

Then source nodes append the h_i and its signature σ_i to every message sending to the network.

- ***Combine*** : We assume forwarding node receives c message from different source nodes, L denotes a c element subset of $[1,n]$. Forwarding node divides set L into d subsets: $L_1, L_2, ..., L_d$, where L_i is the subset of receiving message from source node S_i. And the number of message in subset L_i is c_i . We have $c = \sum_{i=1}^{d} c_i, L = L_1 \cup L_2 \cdots L_d$, computes $\boldsymbol{y} = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{\boldsymbol{v}}_j$.
- *Verify*: By a verifier with the knowledge of $\tilde{\boldsymbol{v}}_i$, vector's hash h_i , and theirs signature σ_i , $\boldsymbol{y} \in F_p$.

  1). The verifier computes $\sigma = \prod_{i=1}^{c} \sigma_i$.

  2). if **(a)** : $e(\sigma, g) = \prod_{i=1}^{c} e(t_i, p_i)$

  and

  (b): $\prod_{i=1}^{d} \prod_{j \in L_i} h_j^{y_{n+(i-1)m+j}} = \prod_{j=1}^{n} g_j^{y_j}$

then outputs 0 (accept), otherwise outputs 1 (reject).

Correctness:

(a) Verification the signature of message's hash

$$e(\sigma, g) = e(\prod_{i=1}^{c} \sigma_i, g) = e(\prod_{i=1}^{c} t_i^{x_i}, g) = \prod_{i=1}^{c} e(t_i^{x_i}, g)$$

$$= \prod_{i=1}^{c} e(t_i, p_i)$$

(b) Verification the integrity of message

$$\boldsymbol{y} = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{\boldsymbol{v}}_j$$

$= (y_1, y_2, ..., y_n, ..., y_{n+m}, ..., y_{n+dm})$, where

$$y_1 = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{\boldsymbol{v}}_{1j}$$

$$\cdots$$

$$y_n = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{\boldsymbol{v}}_{nj}$$

$$y_{n+(i-1)m+j} = \beta_{ij}$$

$$\prod_{i=1}^{d} \prod_{j \in L_i} h_i^{\beta_{ij}}$$

$$= \prod_{i=1}^{d} \prod_{j \in L_i} (\prod_{k=1}^{n} g_k^{v_{ik}})^{\beta_{ij}} = \prod_{k=1}^{n} g_k^{\sum_{i=1}^{d} \sum_{j \in L_i} v_{ik} \beta_{ij}}$$

$$= \prod_{j=1}^{n} g_j^{y_j}$$

6. Security Analysis

The complication arises from the fact that the forwarding nodes wish to combine vectors from files produced by different sources, but each source knows nothing of what the other sources are doing.

***Theorem** 2 The data integrity scheme **NS** is secure assuming that **VH** is a secure vector hash function, and **BGL** is a secure signature scheme. In particular, we assume there exists a polynomial-time adversary breaks NS. Then there exists a polynomial-time adversary*
polynomial-time algorithm
space hash VH, such that :

NS_Adv[A, NS] ≤ Sig _ Adv[B, BGL] + Hash_Adv[X, VH]

Where NS_Adv[A, NS] is the probability that
scheme NS.

6.1. Signature Forging

In the scheme, data integrity is verified by homomorphic hashing； then the forwarder uses the aggregate signature [17] to verify the authenticity of all raw hash values. For each hash's signature $\sigma_i^{(k)}$, we have $e(\sigma_i^{(k)}, g) = e(t_i^{(k)}, p_i)$ .Theorem 1 of [17] proved the existential unforgeability of the signature $c_i^{(k)}$ under a chosen message attack in the random oracle model assuming G_1 and G_2 are a co-gap group pair for computational Co-Diffie-Hellman. For aggregate signature $\sigma = \prod_{i=1}^{c} \sigma_i$, we have $e(\sigma, g) = \prod_{i=1}^{c} e(t_i, p_i)$, and the aggregate signature scheme is proven secure in the random oracle model, on the assumption of hardness of computational based on Co-Diffie-Hellman problem. Therefore, we can conclude that our signature scheme is secure enough to defend against signature forging.

6.2. Hash Collision

(a) Attack at source nodes

Adversary may either generate a hash collision for the massage vector or forge a message vector which can pass the verification. According to *Theorem 1*, for each source node, it is computationally infeasible to find two different messages $\boldsymbol{v} \in F_p^{m+n}$ and $\tilde{\boldsymbol{v}} \in F_p^{m+n}$ such that $\prod_{j=1}^{n} g_j^{v_j} = \prod_{j=1}^{n} g_j^{\tilde{v}_j}$, since a discrete logarithm problem is hard. Therefore, we can conclude that any source node can not forge a message for other source's vectors which can pass the verification. In the second case, the vector's hash (together with a unique file identifier ID) can use a secure traditional standard signature scheme.

(b) Attacks at forwarding nodes

We assume each source's massage vector belongs to one subspace

$$V = span\{\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, ..., \tilde{\boldsymbol{v}}_m\} \subseteq F_p^{n+dm}.$$

For any message vector $\boldsymbol{y}$ at forwarding nodes:

$y \in V \Leftrightarrow y = \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{v}_j$,

An adversary may break our scheme at forwarding nodes by the following methods.

1). Generating a forged hash for a given message vector .

The collision resistance ensures that all packets with the same hash belong to the same vector space. If an adversary wants to forge a hash $\bar{h}$ for a linear combination $\boldsymbol{y}$, from the viewpoint of adversary, the probability is that $\bar{h} = h$ is $1/p^l$, which is negligible.

2). Forging a message vector

A smart adversary may attempt to forge a message $\boldsymbol{y}'$, such that $y \neq \sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{v}_j$, but $h(y) = h(\sum_{i=1}^{d} \sum_{j \in L_i} \beta_{ij} \tilde{v}_j)$. If the forging message $\boldsymbol{y}'$ can pass the verification, it must satisfy: $\prod_{j=1}^{n} g_j^{y_j'} = \prod_{j=1}^{n} g_j^{y_j}$. It is infeasible to find different messages vector $\boldsymbol{y}'$ since a discrete logarithm problem is hard.

7. Performance Valuation

In this section, we first evaluate the proposed scheme in terms of computation overheads from theoretical analysis. After that, we present the experimental results.

7.1 Computation Cost

We now compare the performance of our scheme with previous works [7] and [18] in terms of computation overheads. We define the computation cost of the primarily cryptographic operations as follows. Let C_{me} denote the time cost to perform one exponentiations operation; C_{par} denote the time cost to a pairing operation; C_{mul} denote the time cost to a multiplications operation; **MSP** denote multi-source supporting. We neglect all the operations such as addition operations for the sake of simplicity. We also assume the verifier nodes combine c vectors from different sources; n and d denote the dimension of vectors and the number of source nodes, respectively. According to linear network coding signature theory: $n \geq d, n \geq c$ and $d \geq c$.

Table 1: Computation Overhead of Verification Operation

Scheme	C_{me}	C_{mul}	C_{par}	**MSP**
[7]	$d+n$	$d+n-1$	2	NO
[18]	$2d+n$	$d+n+c-1$	$c+1$	YES
Ours	$n+c$	$d+2c-1$	$c+1$	YES

Table 1 shows the dominant operations of the three signature schemes in terms of verifying an encoded packet. The verification in our scheme requires $c+1$ pairing computation and $n+c$ exponentiations, so there has the optimal computation complexity on average. The more important is that our scheme exploits the vector *Merge* algorithm combining difference messages coming from multiple sources, so there has perfect extension in the multi-source network coding signature scheme.

7.2 Experimental Results

In order to provide numerical results, we implement the performance of the signing and verification operations in different cases. Our implementation was written in C using the Pairing-Based Cryptography Library(libpbc)[22]. The computations are run on PC with 2.9 GHz CPU frequency and 4 GB of RAM, using Linux operating system. The main properties of different tested parameters are summarized in Table 2.

In the experiment, we utilize two elliptic curves: One is *Type a* with a base field size of 512 bits and an embedding degree 2, and the other is *Type e* with a base field size of 1024 bits and an embedding degree 1. The security levels are chosen to be $|p|=512$ and $|p|=1024$, respectively.

Table 2: Main properties of tested pairing

Type	Base field(bits)	Dlog security(bits)	degree of curve
a	512	1024	2
e	1024	1024	1

Table 3: Dimension of Vector Corresponding message Bytes

n	5	10	50	100
Type a	320bytes	640bytes	3200bytes	6400bytes
Type e	5KB	10KB	50KB	100KB

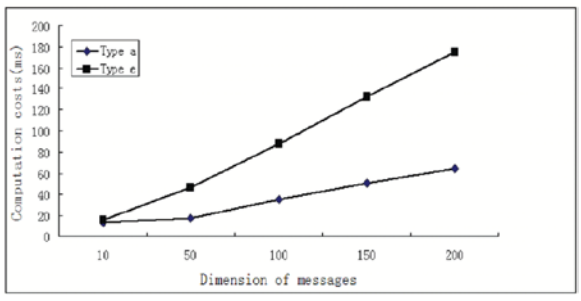
In Figures and Tables,

- d denotes the number of source nodes;
- n denotes the dimension of message vectors;

7.2.1 Performance of our scheme

We demonstrate the effectiveness and efficiency of our proposed mechanism in the signature process and verification process at different pairing parameters. For each source node, computation costs for the signature mainly rely on the message size. In this experiment, we set the number of files $d=50$, and the dimension of messages set to $n=10,50,150,200$, whereas the pairing parameter is fixed at *Type a* and *Type e*. For the sink node's verification, the efficiency of computation mainly depends on both the number of files and the message size. Thus, in this experiment, we set the length of message at 3200 bytes, and the number of files d is set to 10, 50, 100,200, 300.

The run time results of the verification process and signature process are plotted on Fig. 1. From Fig. 1, we can see that the computation cost of the algorithm heavily depends on the selected Type of paring parameters; and for each paring parameters, the computation cost increases with the increasing the dimension of message vectors.



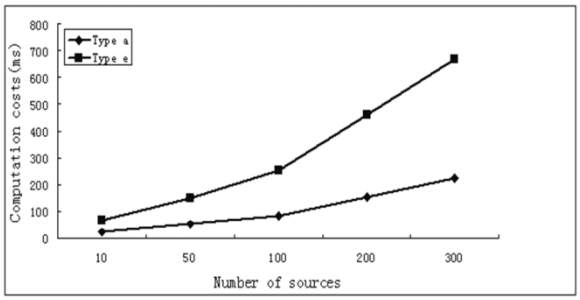


Figure 1: Computation costs of the signature and verification operation

7.2.2 Comparison analysis

We now compare the performance of our scheme with previous work, *Làszlò* 's scheme [18] in the experiments. Table 4 shows our algorithm outperforms the *Làszlò* 's scheme when source node $d \geq 200$.

Table 4: Total run time(s) for algorithm in *Type a*

d	L´aszl´o's scheme	Our Algorithm
50	5.70	5.69
100	11.16	11.20
200	21.61	22.20
300	33.25	32.69
350	38.26	37.95
400	44.98	41.46
600	66.51	62.07

8. Summary

We propose a homomorphic hash-based integrity scheme that provides a cryptographic solution against pollution attacks for systems using inter-flow network coding with multiple sources. Our scheme overcomes that single source cannot handle mixing of packets from multiple sources, and homomorphism can not hold if the unique secret key is replaced by distinct private keys. Because it can combine different messages coming from multiple sources, this implies that the scheme is suitable for practical purposes.

Acknowledgement

The work was supported by the National Natural Science Foundation of China under grant (61163038, 61261015) and Foundation of Northwest Normal University(NWNU-LKQN-13-12).

References

[1]. R. Ahlswede, N. Cai, S. Li ,et al, Network information flow. IEEE Transactions on Information Theory, 2000, pp, 1204-1216.

[2]. Y. Zhu, B. Li, J. Guo, Multicast with network coding in applicationlayeroverlay networks. IEEE Journal on Selected Areas in Communications, January 2004.

[3]. S. Li, R. Yeung, N. Cai, Linear network coding. IEEE Transactions on Information Theory, 49 (2) 2003, pp, 371-381.

[4]. Z. Li, B. Li, Network coding: the case of multiple unicast sessions. In Proc. of 42th annual allerton conference on communication, control and computing, 2004.

[5]. D.S. Lun, M. Medard, R. Koetter, Network coding for efficient wireless unicast. In Proc. of 2006 International Zurich Seminar on Communications (IZS06), Zurich, Switzerland, 2006.

[6]. R. Koetter, M. M´edard, An algebraic approach to network coding. IEEE/ACM Transactions on Networking, 2003, pp, 782-795.

[7]. D. Boneh, D. Freeman, J. Katz et al, Signing a linear subspace: Signature schemes for network coding. In Public Key Cryptography: PKC 2009. LNCS, vol.5443, Springer-Verlag, Heidelberg, 2009, pp, 68-87.

[8]. F. Zhao, T. Kalker, M´edard M, et al, Signatures for content distribution with network coding. In Proc. of International Symposium on Information Theory, 2007.

[9]. K. Han, T. Ho, R. Koetter, et al, On network coding for security. In Military Communications Conference(Milcom), 2007.

[10]. M. Krohn, M. Freedman, D. Mazieres, On the-fly verification of rateless erasure codes for efficient content distribution. In Proc. of IEEE Symposium on Security and Privacy, 2004, 226-240.

[11]. R. Johnson, D. Molnar, D. Song, et al, Homomorphic signature schemes. In Proc. of CT-RSA, LNCS, vol.2271, Springer-Verlag, Heidelberg, 2002, pp, 244-262.

[12]. D. Charles, K. Jain, K. Lauter, Signatures for network coding, In 40th Annual Conference on Information Sciences and Systems(CISS '06), 2006.

[13]. Z. Yu, Y. Wei, B. Ramkumar, et al, An Efficient Signature based Scheme for Securing Network Coding against Pollution Attacks. In Proc. of IEEE INFOCOM, 2008.

[14]. R. Gennaro, J Katz, H. Krawczyk,et al. Secure Network Coding over the Integers. In Public Key Cryptography PKC 2010, LNCS, vol.6056, Springer-Verlag, Heidelberg, 2010, pp, 142-160.

[15]. C. Gkantsidis, P. Rodriguez. Cooperative security for network coding file distribution. In Proc. of IEEE INFOCOM, 2006, pp, 1-13.

[16]. D. Catalano, D. Fiore, B. Warinschi. Efficient network coding signatures in the standard model. In Public Key Cryptography PKC 2012 (2012),pp, 680-696.
[17]. D. Boneh, C. Gentry, B. Lynn, et al, Aggregate and verifiably encrypted signatures from bilinear maps. In Proc. of Eurocrypt 2003, LNCS, vol. 2656, Springer-Verlag, Heidelberg, 2003, pp, 416-432.
[18]. L´aszl´o Czap, I. Vajda, Signatures for Multi-source Network Coding, Cryptology ePrint Archive, Report 2010/328. http://eprint.iacr.org/2010/328.
[19]. S. Agrawal, D. Boneh, X. Boyen, et al, Preventing pollution attacks in multi-source network coding. In Public Key Cryptography PKC 2010, LNCS, vol.6056, Springer-Verlag, Heidelberg, 2010, pp, 161-176.
[20]. J.Shao, J. Zhang, Y.Ling, et al. Multiple Sources Network Coding Signature in the Standard Model[M]//Internet and Distributed Computing Systems. Springer Berlin Heidelberg, 2013, pp,195-208.
[21]. H.Yang, M.Yang. An Unconditionally Secure Authentication Code For Multi-Source Network Coding[J]. International Journal of Wireless and Microwave Technologies (IJWMT), 2012, 2(1): 45.
[22]. :.The pairing-based cryptography library. http://crypto.stanford.edu/pbc/.
[23]. D. Boneh, B. Lynn, H. Shacham. Short signatures from theWeil pairing. In Proc. of Asiacrypt 2001, LNCS, vol.2248, Springer-Verlag, Heidelberg ,2001, pp, 514-532.

Shu-fen Niu is an associate professor in Northwest Normal University (China). She received her M.Sc. (2007) degree from Shanghai University, in operation research. She received the Ph.D. degree in Northwest Normal University. Her current research interests include cloud security, wireless sensor networks and network coding.

Cai-fenWang is a professor and a vice dean of the College of Computer Science and Engineering, Northwest Normal University. She received the Ph.D. degree in Computer Science from Xidian. Her current research interests include wireless sensor networks, network security.

Su-zhen Cao is an associate professor in Northwest Normal University. She received her M.Sc. (2010) degree from Lanzhou Jiaotong University in Computer Science. Her current research interests include wireless sensor networks and network coding.