Computer Networks 54 (2010) 28–40

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An efficient dynamic-identity based signature scheme for secure network coding

Yixin Jiang [a,b], Haojin Zhu [a], Minghui Shi [a], Xuemin (Sherman) Shen [a,*], Chuang Lin [b]

[a] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 3 February 2009
Received in revised form 15 May 2009
Accepted 7 August 2009
Available online 13 August 2009
Responsible Editor: W. Wang

Keywords:
Network coding
Identity-based cryptography
Signature
Pollution attacks

ABSTRACT

The network coding based applications are vulnerable to possible malicious pollution attacks. Signature schemes have been well-recognized as the most effective approach to address this security issue. However, existing homomorphic signature schemes for network coding either incur high transmission/computation overhead, or are vulnerable to random forgery attacks. In this paper, we propose a novel *dynamic-identity based signature scheme* for network coding by signing linear vector subspaces. The scheme can rapidly detect/drop the packets that are generated from pollution attacks, and efficiently thwart random forgery attack. By employing fast packet-based and generation-based batch verification approaches, a forwarding node can verify multiple received packets synchronously with dramatically reduced total verification cost. In addition, the proposed scheme provides one-way identity authentication without requiring any extra secure channels or separate certificates, so that the transmission cost can be significantly reduced. Simulation results demonstrate the practicality and efficiency of the proposed schemes.

1. Introduction

Network coding, as an efficient means of information dissemination, is a promising approach in many practical network applications, such as traditional multicast or broadcast networks [1], wireless sensor networks [2,3], and peer-to-peer content distribution networks [4–7]. Network coding was first introduced in [8] as an alternative to the traditional routing networks, and it has been shown that random linear coding can achieve the optimal throughput for multicast [1,9] and even unicast transmissions [10,11].

Unlike the traditional forwarding approach which requires duplicating every input message, network coding allows each intermediate node to encode packets en-route. Therefore, each output message sent to the downlink can be linear combination of input messages received from the uplinks, as illustrated in Fig. 1 [16]. Generally, network coding system consists of the transmission, encoding, and re-encoding of messages at intermediate nodes, such that the encoded messages can be decoded at their final destinations.

A primary benefit of network coding is that it can improve throughput and minimize the transmission delay of a network. Another compelling benefit is its robustness and adaptability. Practical network coding techniques, such as random linear coding, packet tagging, and buffering, allow the encoding and decoding to proceed in a distributed manner, even if asynchronous packets arrive and depart in arbitrarily varying rate, delay, and loss. Thus, network coding is well suited for dynamic network scenarios, where nodes only have partial information about the global network topology. In addition, network coding can minimize the amount of energy required per packet multicast in wireless networks.

* Corresponding author. Tel.: +1 519 888 4567 32691; fax: +1 519 746 3077.
E-mail addresses: yixin@bbcr.uwaterloo.ca (Y. Jiang), h9zhu@bbcr.uwaterloo.ca (H. Zhu), mshi@bbcr.uwaterloo.ca (M. Shi), xshen@bbcr.uwaterloo.ca (X. (Sherman) Shen), clin@csnet1.cs.tsinghua.edu.cn (C. Lin).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.08.006

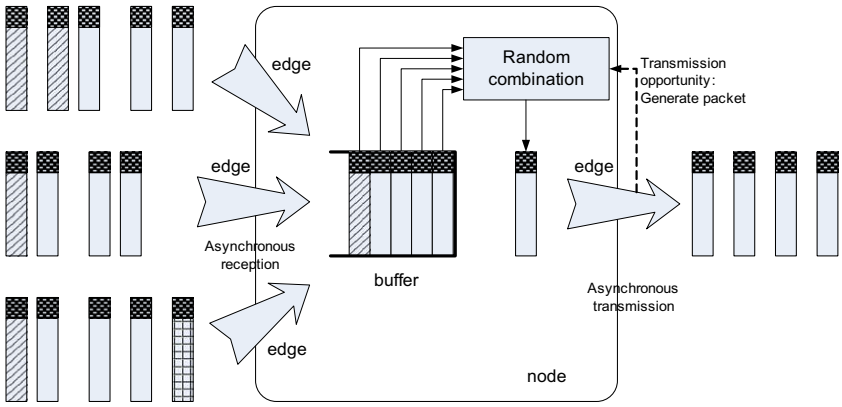

Fig. 1. Coding at network node.

However, network coding may face potential security threats due to open multi-hop communications and the packet encoding at intermediate forwarders. Since network coding involves mixing of packet inside the network, several primary types of attacks, *pollution attacks, random forgery attacks* [38], and *entropy attacks* [5], are particularly relevant to network coding. The *pollution attack* is originated from any malicious behaviors of un-trusted forwarders or adversaries, such as injecting polluted information, modifying and replaying the disseminated messages, which could be fatal to the whole networks. Although this may also occur in a traditional network system without network coding, its effect is far more serious with network coding. If a junk message is mixed by a forwarder, the output messages of the forwarder will be contaminated. Such polluted messages should be detected and filtered as early as possible, since they may spread to all downstream nodes by re-encoding junk messages. The *random forgery attack is* related to homomorphic signature function itself. Jognson et al. [38] conclude that for an additive homomorphic signature function defined on the lattice $L = (\mathbb{Z}/m\mathbb{Z})^d$, if an adversary can derive signature $Sig(x_1), \ldots, Sig(x_d)$, where $x_1, \ldots, x_d$ are a basis for L, then it can launch successful random forgery attacks to the additive homomorphic signature function. The *entropy attack* can be considered as a special replay attack, where an adversary may use "stale" encoded packet vectors to forge non-innovative packets that are trivial linear combinations of existing packets at the forwarders. Although entropy attack does not destroy the linear algebraic constraint conditions between the original packet and the appended encoding vector, it reduces the decoding opportunities at sinks and the overall throughput rate. How to thwart entropy attacks exceeds the scope of this paper, which we have explored at length in [36].

For secure network coding, it is prerequisite to achieve efficient message integrity and validity. The non-cryptography based schemes [14,15] can only detect or filter out polluted messages at the sinks, but not at the forwarders. A well-recognized cryptography-based solution is to sign each message with a signature. However, the traditional hash function based signature schemes may be unsuitable for network coding, since the original source signatures can be destroyed by the subsequent encoding process, which is performed at each forwarder. The basic idea in existing cryptography-based schemes is to detect each packet before it gets mixed into the buffer, including a homomorphic hash scheme [5], a homomorphic signature scheme [12], and a secure random checksum scheme [5]. These solutions either require an extra secure channel [5], incur high computation overhead due to not supporting batch verification [12], suffer from relatively high extra transmission overhead [5,39], endure weak scalability [12,39], or are vulnerable to the random forgery attack [12,13], by which an adversary may arbitrarily forge signatures for a given message if sufficient signatures of "stale" messages are collected [38]. Recently, Yu et al. [13] propose an efficient homomorphic signature scheme based on the RSA signature scheme, with which the forwarders can achieve efficient verification at the expense of increased transmission overhead, since the size of a RSA signature is typically very large in the order of hundreds of bytes. Zhao et al. [39] also present a novel signature scheme for network coding by authenticating the vector sub-space. The significant drawback in this scheme is that the size of both the public signature information and public keys is at least the square root of the file size. Moreover, the scheme is not efficient for distributing multiple files with the same public key, which significantly impairs the system scalability. Finally, to calculate the public signature information, the scheme requires the source to buffer the entire file in advance. Therefore the scheme is not suitable for streaming live data, which are generated on-the-fly. These aforementioned deficiencies motivate us to explore a more efficient and scalable scheme for securing network coding.

In this paper, we propose an efficient *dynamic-identity based signature* scheme for secure network coding, which features the following notable properties: (1) **Efficiency**: The proposed signature scheme can support fast identity-based batch verification, and rapid signature generation for the output packets. By employing two optimized verification techniques, packet-based and generation-based batch verification methods, a node can quickly verify mul-

tiple received packets in batch such that the total verification cost can be dramatically reduced. Thus the proposed scheme effectively eliminates the performance bottleneck due to the greatly reduced computational overhead at forwarders. Moreover, with identity-based signature, both certificate management cost and the transmission overhead can be significantly reduced; (2) **Security**: To address the security and robustness of our scheme, a Multi-level Binary Authentication Tree (M-BAT) approach is proposed for detecting pollution attacks. In addition, with the one-way *dynamic-identity based signature* function, the scheme can efficiently thwart random forgery attack, which exists in most of reported homomorphic signature schemes for network coding. The proposed scheme also does not need any extra secure channel, and provides source authentication via one-way identity hash-chain. (3) **Scalability**: In the proposed scheme, the signature keys can be updated with one-way pseudo-identity refreshing in a natural way, while the public keys keep invariant. Therefore, the proposed scheme is more efficient for transmitting live data or distributing multiple files with the same public keys. Such features effectively improve deployment scalability of the proposed scheme.

The remainder of the paper is organized as follows. In Section 2, preliminaries related to the proposed research are given, including the network coding model, adversary model, and the pairing concept. In Section 3, the proposed signature scheme is introduced in details. In Sections 4 and 5, the security analysis and performance evaluation are presented, respectively. In Section 6, the related works are discussed, followed by the conclusions in Section 7.

2. Preliminaries

In this section, we briefly present the practical network coding and the adversary model, followed by the introduction of bilinear pairing, which is the foundation of the proposed scheme.

2.1. Network coding model

The principle behind network coding is to allow intermediate nodes to re-encode the incoming packets. In practical network coding [16,35], the information source outputs a continuous stream of packets, which can be grouped into blocks with n source packets per block. Let all the code packets in the network related to the kth source block be denoted by *generation k*. To keep tracking of packets in same generation, each packet is tagged with its generation number k.

Fig. 1 illustrates a typical network node with three incoming links and one outgoing link. Packets with *generation number* (shown as shade) arrive sequentially through each link and are put into a buffer sorted by generation, where the packets with the "*active generation*" at the head of the queue. Once there is a transmission opportunity for an outgoing link, an outgoing packet is formed by taking a random linear combination of packets with the active generation.

In the following, we introduce the *general algebraic model* of network coding. Let an acyclic network (V, E, c) be denoted by a set of nodes (or vertices) V, a set of directed links (or edges) E with unit capacity edges, i.e., $c(e) = 1$ for all $e \in E$, which means that each edge can carry one symbol per unit of time. Assume that each symbol is an element of a finite field $\mathbb{Z}_q$, where q is a primer.

In the proposed scheme, we consider a single source $s \in V$ and a set of sinks $T \subseteq V$(or a single node $t \in V$). Let $n = MinCut(s, T)$ be the multicast capacity. Assume that s attempts to send some information blocks to the sinks in $T \subseteq V$. For one block with a given *generation number*, it can be divided into n packets $\{\boldsymbol{B}_1^r, \boldsymbol{B}_2^r, \ldots, \boldsymbol{B}_n^r\}$ per block. Similar to the setting in [5,12], each packet $\boldsymbol{B}_i^r$ $(i = 1, \ldots, n)$ is further divided into m symbols, which can be denoted as a vector such as $\boldsymbol{B}_i^r = [b_{i,1}^r, b_{i,2}^r, \ldots, b_{i,m}^r]$, where $b_{i,j}^r \in \mathbb{Z}_q (1 \leqslant j \leqslant m)$ is the original symbols. The auxiliary variables t_r and g_r are the time-stamp and the generation number, respectively, and $h(\cdot)$ is a one-way hash function such as SHA-1.

For each edge e emanating from a node v, let $y(e) \in \mathbb{Z}_q$ denote the symbol carried on e, which can be computed as a linear combination of the symbols $y(e')$ carried on edges e' entering node v, namely, $y(e) = \sum_{e'} \beta_{e'}(e) y(e')$. The coefficients $\beta_{e'}(e)$ form a *local encoding vector* $\boldsymbol{\beta}(e) = [\beta_{e'}(e)]$ on edge e.

In practical networks, symbols flow sequentially over the edges, and they are grouped into packets. Corresponding to the source packet $\boldsymbol{B}_i^r = [b_{i,1}^r, b_{i,2}^r, \ldots, b_{i,m}^r]$, each packet in the network can be considered as a vector $Y_r(e) = [y_1^r(e), y_2^r(e), \ldots, y_m^r(e)]$. Thus, each packet $Y_r(e)$ on edge e can be computed as a linear combination of the packets $Y_r(e')$ on the preceding edges or, alternatively, as a linear combination of the source packets $\{\boldsymbol{B}_1^r, \boldsymbol{B}_2^r, \ldots, \boldsymbol{B}_n^r\}$ by induction

$$\begin{aligned} \boldsymbol{Y}_r(e) &= \sum_{e'} \beta_{e'}(e) \boldsymbol{Y}_r(e') = \sum_{i=1}^{n} g_i(e) \cdot \boldsymbol{B}_i^r \\ &= \left[\sum_{i=1}^{n} g_i(e) \cdot b_{i,1}^r, \ldots, \sum_{i=1}^{n} g_i(e) \cdot b_{i,m}^r \right]. \end{aligned} \tag{1}$$

The coefficients of this combination form a *global encoding vector* $\boldsymbol{G}(e) = [g_1(e), g_2(e), \ldots, g_n(e)]$ on edge e, which can be computed recursively as $\boldsymbol{G}(e) = \sum_{e'} \beta_{e'(e)\boldsymbol{G}(e')}$, using the local encoding vectors $\beta(e)$. The vector $\boldsymbol{G}(e)$ represents the symbol $\boldsymbol{Y}_r(e)$ in terms of the source symbols $\{\boldsymbol{B}_1^r, \boldsymbol{B}_2^r, \ldots, \boldsymbol{B}_n^r\}$.

To facilitate the decoding at the sinks, each packet carried on edge e is appended with its global encoding vector $\boldsymbol{G}(e)$. This can be achieved by prefixing the ith packet vector $\boldsymbol{B}_i^r$ with the ith unit vector $\boldsymbol{U}_i$ and applying the algebraic operations to the resulting vector, i.e., $[\boldsymbol{Y}_r(e), \boldsymbol{G}(e)] = \sum_{e'} \beta_{e'}(e) [biY_r(e'), \boldsymbol{G}(e')] = \sum_{i=1}^{n} g_i(e) \cdot [\boldsymbol{B}_i^r, \boldsymbol{U}_i]$. Therefore, the augmented source packet $\widetilde{\boldsymbol{B}}_i^r$ and corresponding encoded packet $\widetilde{\boldsymbol{Y}}_r(e)$ are denoted as

$$\begin{aligned} \widetilde{\boldsymbol{B}}_i^r &= [\boldsymbol{B}_i^r, \boldsymbol{U}_i] \\ &= [\underbrace{b_{i,1}^r, \ldots, b_{i,m}^r}_{m}, \underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{n-i}] \\ &= [\underbrace{\tilde{b}_{i,1}^r, \ldots, \tilde{b}_{i,m}^r}_{m}, \underbrace{\tilde{b}_{i,m+1}^r, \ldots, \tilde{b}_{i,m+n}^r}_{n}], \end{aligned} \tag{2}$$

$$\begin{aligned}\widetilde{\boldsymbol{Y}}_r(e) &= \sum_{i=1}^{n} g_i(e)\cdot\tilde{\boldsymbol{B}}_i^r \\ &= \sum_{i=1}^{n} g_i(e)\cdot[\boldsymbol{B}_i^r,\boldsymbol{U}_i] \\ &= [\underbrace{y_1^r(e),\ldots,y_m^r(e)}_{m},\underbrace{g_1(e),\ldots,g_n(e)}_{n}] \\ &= [\underbrace{\tilde{y}_1^r(e),\ldots,\tilde{y}_m^r(e)}_{m},\underbrace{\tilde{y}_{m+1}^r(e),\ldots,\tilde{y}_{m+n}^r(e)}_{n}],\end{aligned} \tag{3}$$

Tagging each packet with the corresponding global encoding vector allows the distributed decoding procedure and requires no knowledge of encoding functions for the nodes. Furthermore, once a sink $t \in T$ receives n packets $\boldsymbol{Y}_r(e_1), \boldsymbol{Y}_r(e_2), \ldots, \boldsymbol{Y}_r(e_n)$, which can be denoted as

$$\begin{aligned}\begin{bmatrix}\boldsymbol{Y}_r(e_1)\\ \cdots \\ \boldsymbol{Y}_r(e_n)\end{bmatrix} &= \begin{bmatrix} g_1(e_1) & g_2(e_1) & \cdots & g_n(e_1)\\ \cdots & \cdots & \cdots & \cdots \\ g_1(e_n) & g_2(e_n) & \cdots & g_n(e_n)\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{B}_1^r\\ \cdots \\ \boldsymbol{B}_n^r\end{bmatrix} \\ &= G_t\cdot\begin{bmatrix}\boldsymbol{B}_1^r\\ \cdots \\ \boldsymbol{B}_n^r\end{bmatrix},\end{aligned}$$

where the $\boldsymbol{G}(e_i)$ is the global encoding vector associated with $\boldsymbol{Y}_r(e_i)$. Then sink t can recover the n source packets. The matrix $\boldsymbol{G}_t$ is invertible with high probability, if the coefficients in local encoding vectors are chosen randomly from $\mathbb{Z}_q$ with an adequately larger size than that of the network [17,18].

2.2. Adversary model

In the proposed scheme, we assume that a source, which utilizes the networking systems to provide content distribution service for multiple sinks, is always trusted while the forwarders may not be trusted since they may potentially disrupt the normal coding operation by sending/injecting invalid packets. Therefore, network coding based applications may be vulnerable to various potential attacks. Generally, the possible attacks considered in this paper include Pollution Attacks and Forgery Attacks, which can further classified into general forgery attacks and random forgery attacks.

(1) **Pollution attacks**: The pollution attack can be defined as that a malicious intermediate node can inject junk packets into the network to pollute the output, and further contaminate the entire downstream, preventing proper decoding. Formally, we describe the pollution attacks as follows. A packet $\widetilde{\boldsymbol{Y}}_r(e)$ is a polluted one if the vector $\widetilde{\boldsymbol{Y}}_r(e)$ is not equal to the product of the original augmented packet vector $[\tilde{\boldsymbol{B}}_1^r, \tilde{\boldsymbol{B}}_2^r, \ldots, \tilde{\boldsymbol{B}}_n^r]$ and the global encoding vector, i.e.,

$$\widetilde{\boldsymbol{Y}}_r(e) \neq \sum_{i=1}^{n} g_i(e)\tilde{\boldsymbol{B}}_i^r \quad \text{or} \quad \widetilde{\boldsymbol{Y}}_r(e) \neq \sum_{j=1}^{m+n}\sum_{i=1}^{n} g_i(e)\tilde{b}_{ij}^r, \tag{4}$$

where $\boldsymbol{G}(e) = [g_1(e), g_2(e), \ldots, g_n(e)]$ is the global encoding vector embedded into packet $\widetilde{\boldsymbol{Y}}_r(e)$.

(2) **Forgery attacks**: The attackers can also try to forge signatures to prevent the intermediate nodes from detecting the forged packets, which is defined as general forgery attack. A variant of forgery attack is random forgery attack, which can be defined as that, given signatures on a small set of known messages, the adversary can forge signatures for other possible messages [38]. In the context of network coding, random forgery attacks mean that an attacker attempts to generate valid signatures for arbitrarily false encoded packets based on the collected signatures for stale encoded packets.

2.3. Identity-based cryptography and bilinear pairing

Identity-based cryptography (IBC) is a type of public-key cryptography in which the public key of a user is its unique identity information. The primary IBC schemes include Boneh et al.'s pairing-based scheme [19], Cocks's quadratic-residue based scheme [37], etc. As an important IBC scheme, the pairing-based IBC scheme can offer lower transmission cost compared with the traditional RSA-based schemes due to the smaller signature overhead. We briefly introduce the bilinear pairing as follows.

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ respectively be a cyclic additive group and a cyclic multiplicative group generated by P with the same prime order q, i.e., $|\mathbb{G}| = |\mathbb{G}_{\mathbb{T}}| = q$. Let $\hat{e}: \mathbb{G}\times\mathbb{G}\to\mathbb{G}_{\mathbb{T}}$ be a bilinear map, which satisfies the following properties:

(1) **Bilinear**: $\forall P, Q, R \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_q$, $\hat{e}(Q, P+R) = \hat{e}(P+R, Q) = \hat{e}(P, Q)\cdot\hat{e}(R, Q)$. Especially, $\hat{e}(aP, bP) = \hat{e}(P, bP)^a = \hat{e}(aP, P)^b = \hat{e}(P, P)^{ab}$.
(2) **Non-degenerate**: $\exists P, Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_{\mathbb{T}}}$.
(3) **Computable**: $\forall P, Q \in \mathbb{G}$, there is an efficient algorithm to calculate $\hat{e}(P, Q)$.

Such a bilinear map $\hat{e}$ can be constructed by the modified Weil [19] or Tate pairings [20] on elliptic curves, on which the Decisional Diffie–Hellman (DDH) problem is easy to be solved while the Computational Diffie–Hellman (CDH) problem is believed hard [21].

3. An efficient dynamic-identity based signature scheme for network coding

In this section, we propose an efficient *dynamic-identity based signature* scheme for network coding, where each node can rapidly tag/drop packets from pollution attacks and thwart random forgery attacks.

3.1. Dynamic-identity based signature scheme

The proposed signature scheme is based on identity-based cryptography [19]. There are three parties in the system: the forwarders (signer and verifier), the sinks (verifier), and the source (verifier). The source is responsible of generating public/private security parameters, and the public security parameters can be published with the trusted third party's signature.

The basic scheme mainly consists of three algorithms: **setup**, **sign**, and **verifying**.

Setup: In this phase, the source needs to set up the basic security parameters and to generate the following private/public key pairs, pseudo identity and the resultant identity-aware signature keys.

(1) Bilinear map parameters: Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be a cyclic additive group and a cyclic multiplicative group, where $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ are generated by P with the same order q. Let $\hat{e}:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_{\mathbb{T}}$ be a bilinear map. $H(\cdot)$ is a MapToPoint hash [21] function such that $H:\{0,1\}^*\to\mathbb{G}$.

(2) The source generates $m+n$ random numbers $\{s_1,s_2,\ldots,s_{m+n}\}\in\mathbb{Z}_q^*$ as its secret master keys. The source also derives a temporary pseudo identity PID from its real identity ID. the source pre-computes the following $m+n$ temporary secret signature keys:

$$\boldsymbol{SK}=\{SK_i|SK_i=s_iH(PID),\ 1\leqslant i\leqslant m+n\},\tag{5}$$

where $H(\cdot)$ is a MapToPoint hash [21] function such as $H:\{0,1\}^*\to\mathbb{G}$. Note that, to preserve the privacy of signature keys SK_i, the temporary pseudo identity PID will be changed, once a source has sent $\lfloor(m+n-1)/n\rfloor\cdot n$ linear independent packet vectors. Specially, for the k^{th} identity refreshment, we can introduce one-way forward hash chain to update the pseudo identity PID as

$$PID=h^k(ID),\tag{6}$$

where $h(\cdot)$ is a one-way hash function such that MD5 or SHA-1. With the parameter $\{ID,k\}$, a node can easily verify the authenticity of pseudo identity PID by checking $PID=h^k(ID)$.

(3) The source uses $m+n$ master keys $\{s_1,s_2,\ldots,s_{m+n}\}\in\mathbb{Z}_q^*$ to compute the following public keys:

$$\boldsymbol{PK}=\{PK_i|PK_i=s_iP,\ 1\leqslant i\leqslant m+n\}.\tag{7}$$

Finally, the source publicizes the security parameters $\{\mathbb{G},\mathbb{G}_{\mathbb{T}},q,P,\boldsymbol{PK},ID\}$ to all nodes.**Sign**: According to the network coding model, the source calculates the signatures for its packets $\{\widetilde{\boldsymbol{B}}_1^r,\widetilde{\boldsymbol{B}}_2^r,\ldots,\widetilde{\boldsymbol{B}}_n^r\}$, respectively. Let $\boldsymbol{H}_S(\cdot)$ denote the homomorphic signature function. For $\widetilde{\boldsymbol{B}}_i^r$, the corresponding signature$\boldsymbol{H}_S(\widetilde{\boldsymbol{B}}_i^r)$can be defined as

$$\boldsymbol{H}_S(\widetilde{\boldsymbol{B}}_i^r)=\sum_{j=1}^{m+n}\left\{\tilde{b}_{i,j}^rSK_j\right\}=\sum_{j=1}^{m+n}\{\tilde{b}_{i,j}^rs_jH(PID)\}.\tag{8}$$

Then, the source constructs and delivers a packet $\{PID,k,\widetilde{\boldsymbol{B}}_i^r,\boldsymbol{H}_S(\widetilde{\boldsymbol{B}}_i^r)\}$ to the downstream nodes. Similarly, the signature of encoded packet $\widetilde{Y}_r(e)$ in Eq. (5) can also be denoted as

$$\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))=\sum_{j=1}^{m+n}\{\tilde{y}_j^r(e)SK_j\}=\sum_{j=1}^{m+n}\{\tilde{y}_j^r(e)s_jH(PID)\}.\tag{9}$$

For securing network coding, each node needs to append signature $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ to its output packet $\widetilde{\boldsymbol{Y}}_r(e)$. Due to the homomorphism of the signature function, it is not required to compute $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ such as Eq. (10). For an output packet $\widetilde{\boldsymbol{Y}}_r(e)=[\boldsymbol{Y}_r(e),\boldsymbol{U}(e)]=\sum_{i=1}^n g_i(e)\cdot\widetilde{\boldsymbol{B}}_i^r$, its signature can also be calculated as

$$\begin{aligned}\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))&=\boldsymbol{H}_S\left(\sum_{i=1}^n g_i(e)\widetilde{\boldsymbol{B}}_i^r\right)\\&=\sum_{j=1}^{m+n}\left(SK_j\sum_{i=1}^n g_i(e)\tilde{b}_{i,j}^r\right)\\&=\sum_{i=1}^n\left(g_i(e)\left(\sum_{j=1}^{m+n}\tilde{b}_{i,j}^rSK_j\right)\right)\\&=\sum_{i=1}^n g_i(e)\boldsymbol{H}_S(\widetilde{\boldsymbol{B}}_i^r)\quad(\because\text{Eq. (9)}).\end{aligned}\tag{10}$$

Verifying: To efficiently thwart the pollution attacks and the random forgery attacks, the forwarders or sinks perform the following dynamic-identity-based packet verification procedure.

Step (1) On receiving the encoded packet $\{PID,k,\widetilde{\boldsymbol{Y}}_r(e'),\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))\}$ from an incoming edge $e\prime$, the forwarders or sinks with the parameters $\{\mathbb{G},\mathbb{G}_{\mathbb{T}},q,P,\boldsymbol{PK},ID\}$ verify the authenticity of both pseudo identity PID and the corresponding signature by checking if

$$PID=h^k(ID),\tag{11}$$

$$\hat{e}(\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e')),P)=\hat{e}\left(H(PID),\sum_{j=1}^{m+n}\tilde{y}_j^r(e')PK_j\right).\tag{12}$$

Eq. (12) holds since

$$\begin{aligned}&\hat{e}(\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e')),P)\\&=\hat{e}\left(\sum_{j=1}^{m+n}\tilde{y}_j^r(e')SK_j,P\right)=\hat{e}\left(\sum_{j=1}^{m+n}\tilde{y}_j^r(e')s_jH(PID),P\right)\\&=\hat{e}\left(H(PID),\sum_{j=1}^{m+n}\tilde{y}_j^r(e')s_jP\right)=\hat{e}\left(H(PID),\sum_{j=1}^{m+n}\tilde{y}_j^r(e\prime)PK_j\right).\end{aligned}\tag{13}$$

The bogus packets are discarded, and the valid packets are accepted and further used for encoding or decoding. Eq. (12) indicates that the computation cost to verify a signature primarily consists of two pairing, $m+n$ point multiplications, and one MapToPoint hash operation. The computation cost of a pairing operation is much higher than that of a MapToPoint or point multiplication one.

According to the bilinear property of pairing, the verification cost in Eq. (12) could also be reduced by pre-computation optimization as follows:

$$\begin{aligned}\hat{e}(\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e')),P)&=\prod_{j=1}^{m+n}\hat{e}(\tilde{y}_j^r(e')H(PID),PK_j)\\&=\prod_{j=1}^{m+n}\hat{e}(H(PID),PK_j)^{\tilde{y}_j^r(e')}=\prod_{j=1}^{m+n}\delta_j^{\tilde{y}_j^r(e\prime)},\end{aligned}\tag{14}$$

where $\delta_j = \hat{e}(H(PID), PK_j)$ is pre-computed and distributed in advance. Thus the time-consuming pairing operation is replaced with comparable low-cost exponential operation.

3.2. Batch verification

We can further reduce the computation overhead and accelerate the verification process of identity-based signatures by using batch verification [22–28], which can verify all received signatures synchronously instead of sequentially.

As shown in Fig. 1, all packets entering a node will be tagged by *generations*. Each emanated packet is formed by taking a random linear combination of packets with the same generation. In the following, we introduce two forms of batch verification, packet-based batch verification and generation-based batch verification, to optimize the performance. The generation-based batch verification is well suitable for the *interleaving generation* coding policy introduced in [35], which can effectively reduce the *delay spread*.

Packet-based batch verification: For each outgoing edge e at a forwarder or sink v, let $\widetilde{\boldsymbol{Y}}_r(e)$ denote the packet carried on e. Packet $\widetilde{\boldsymbol{Y}}_r(e)$ can be computed as a linear combination of the packet $\widetilde{\boldsymbol{Y}}_r(e')$ on edges e' entering a forwarder, namely, $\widetilde{\boldsymbol{Y}}_r(e) = \sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_r(e')$. Due to the homomorphic signature function, the signature of this new packet tagged with generation r can be obtained as

$$\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) = \sum_{e'} \beta_{e'}(e) H_S(\widetilde{\boldsymbol{Y}}_r(e')). \tag{15}$$

Eq. (15) holds since

$$\begin{aligned}
\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) &= \boldsymbol{H}_S\left(\sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_r(e')\right) \\
&= \boldsymbol{H}_S\left(\sum_{e'} \beta_{e'}(e)\left(\sum_{i=1}^{n} g_i(e')\widetilde{\boldsymbol{B}}_i^r\right)\right) \\
&= \sum_{j=1}^{m+n}\left(SK_j \sum_{e'}\left(\beta_{e'}(e)\sum_{i=1}^{n} g_i(e')\tilde{b}_{i,j}^r\right)\right) \\
&= \sum_{e'}\left(\beta_{e'}(e)\left(\sum_{j=1}^{m+n} SK_j \sum_{i=1}^{n} g_i(e')\tilde{b}_{i,j}^r\right)\right) \\
&= \sum_{e'}\left(\beta_{e'}(e)\left(\sum_{i=1}^{n} g_i(e')\sum_{j=1}^{m+n} SK_j\tilde{b}_{i,j}^r\right)\right) \\
&= \sum_{e'}\left(\beta_{e'}(e)\left(\sum_{i=1}^{n} g_i(e')\boldsymbol{H}_S(\widetilde{\boldsymbol{B}}_i^r)\right)\right) \\
&= \sum_{e'} \beta_{e'}(e)\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e')) \quad (\because \text{Eq. (12)}).
\end{aligned} \tag{16}$$

Consider $\widetilde{\boldsymbol{Y}}_r(e) = \sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_r(e') = [\tilde{y}_1^r(e), \ldots, \tilde{y}_{m+n}^r(e)]$, the forwarder or sink can verify the authenticity of the corresponding signatures $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ by checking if

$$\hat{e}(\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)), P) = \hat{e}(H(PID), \sum_{j=1}^{m+n} \tilde{y}_j^r(e) PK_j). \tag{17}$$

The packet-based batch verification equation can be proved similar to that of Eq. (13).

Let $in(v) = \{e | out(e) = v\}$ and $|in(v)|$ denote the edge set and the average number of edges entering a node v, respectively. From the batch verification equation, the computation cost to verify such $|in(v)|$ signatures is dominantly comprised of $(m + n + |in(v)|)$ point multiplication operations, one MapToPoint hash operation, and two pairing operations. Compared to the sequential verification using Eq. (12), the number of time-consuming pairing operation is reduced to two from $2|in(v)|$, and the number of point multiplication operations is reduced to $(m + n + |in(v)|)$ from $|in(v)|(m + n)$.

Generation-based batch verification: To further reduce the verification cost, each forwarder can aggregate the multiple packet-based signatures associated to the same pseudo-identity *PID*, and then perform the *generation-based batch verification* on the aggregated signature. In our scheme, the aggregate signature is equal to $\sum_{i=1}^{k} \omega_i$, given any k distinct generate-based signatures sent by the same signer, $\omega_1, \omega_2, \ldots, \omega_k$. For example, as shown in Fig. 1, a forwarder receives three types of packets tagged with generation $\{u, v, w\}$ during a given period. Instead of separately verifying the three emanated packets denoted by $\{\widetilde{\boldsymbol{Y}}_u(e), \widetilde{\boldsymbol{Y}}_v(e), \widetilde{\boldsymbol{Y}}_w(e)\}$, each forwarder can verify them in batch as follows, by aggregating these three signatures with same source pseudo-identity *PID*

$$\begin{aligned}
&\hat{e}(\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_u(e)) + \boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_v(e)) + \boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_w(e)), P) \\
&\quad = \hat{e}\left(H(PID), \sum_{j=1}^{m+n}(\tilde{y}_j^u(e) + \tilde{y}_j^v(e) + \tilde{y}_j^w(e))PK_j\right),
\end{aligned} \tag{18}$$

where $\{\tilde{y}_j^u(e), \tilde{y}_j^v(e), \tilde{y}_j^w(e)\}$ is the element in vectors $\{\sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_u(e'), \sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_v(e'), \sum_{e'} \beta_{e'}(e)\widetilde{\boldsymbol{Y}}_v(e')\}$.

Let ε be the number of generations associated to all the packets entering a node v for a given time window. Without loss of generality, assume that each generation-based aggregate signature include $|in(v)|$ packet-based signatures embedded in the packets entering the node v. So the computation cost to verify such ε generation-based signature primarily consists of $m + n + \varepsilon|in(v)|$ point multiplication operations, one MapToPoint hash operation, and two pairing operations. Compared with the sequential verification in Eq. (12), an attractive result is that the number of time-consuming pairing operation is reduced to two from $2\varepsilon|in(v)|$, and the number of point multiplication operations is reduced to $m + n + \varepsilon|in(v)|$ from $\varepsilon|in(v)|(m + n)$. Thus, the verification delay for a node to verify a large number of received massages can be dramatically reduced, which can apparently reduce the packet loss ratio due to the bottleneck of signature verification.

3.3. M-BAT: multi-level binary authentication tree

If the aggregate signatures pass verification, all the input packets are accepted. Otherwise, one or more packets should be polluted, and therefore, further verification should be carried out. Here, we introduce a modified version of Binary Authentication Tree (BAT) in [29], called M-BAT (Multi-level BAT) to find the malicious packets, which can efficiently address the robustness issues of aggregate signatures.

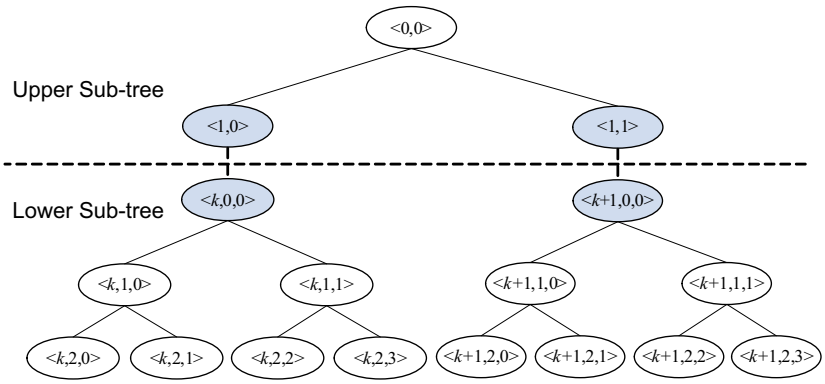

Fig. 2. Multi-level binary authentication tree (M-BAT).

Our approach is based on the data structure in Fig. 2. The M-BAT primarily consists of two level sub-trees. The upper-level tree, called *generation-based sub-tree*, is used to aggregate the generation-based signatures associated to same information source, whereas the lower-level tree, called *packet-based sub-tree*, is used to aggregate the packet-based signature associated to same generation.

Without loss of generality, for a given interval, the generations of the incoming packets at a forwarder are denoted as $\{k, k+1, \ldots, k+b-1\}$, where $b = 2^g$, and the number of edges e' entering a forwarder is equal to $a = 2^h$. For each edge e emanated from this forwarder, the packet $\tilde{Y}_r(e)$ carried on e, tagged with generation r $(k \leqslant r < k+b)$, can be computed as a linear combination of the packet $\widetilde{\boldsymbol{Y}}_r(e'_i)$ on edges e', namely, $\widetilde{\boldsymbol{Y}}_r(e) = \sum_{i=0}^{k-1} \beta_{e'_i}(e)\widetilde{\boldsymbol{Y}}_r(e'_i)(k \leqslant r < k+b)$. Then, the lower sub-tree and the upper sub-tree of an M-BAT can be constructed as follows, respectively. For the lower sub-tree,

(1) The leaf nodes $\langle r, h, v\rangle (v = 0, 2, \ldots, a-1, k \leqslant r < k+b)$ are associated with the signatures $\alpha^r_v = \beta_{e'_v}(e)\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e'_v))$, respectively;
(2) Inner nodes $\langle r, 0, 0\rangle (k \leqslant r < k+b)$, as the root of lower sub-tree and the leaf node of upper sub-tree, are respectively associated to the signatures of $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))(k \leqslant r < k+b)$ in Eq. (15), which are calculated as $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) = \sum_{e'} \beta_{e'}(e) H_S(\widetilde{\boldsymbol{Y}}_r(e'))$. The inner sub-root $\langle r, 0, 0\rangle$ is associated with an aggregate signature $\alpha^r_{\langle 0,0\rangle} = \sum_{i=0}^{a-1} \alpha^r_i$.
(3) The other node $\langle r, l, v\rangle (0 < l < h)$ is associated with an aggregate signature $\alpha^r_{\langle l,v\rangle}$ for the leaf nodes of a sub-tree rooted at $\langle r, l, v\rangle$, where $\alpha^r_{\langle l,v\rangle} = \sum_{i=k_1}^{k_2} \alpha^r_i = \sum_{i=k_1}^{k_2} \beta_{e'_i}(e)\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e'_i)), k_1 = 2^{h-l} \cdot v$, and $k_2 = 2^{h-l} \cdot (v+1) - 1$. The authenticity of the signature $\alpha^r_{\langle l,v\rangle}$ can be verified by checking if

$$\hat{e}(\alpha^r_{\langle l,v\rangle}, P) = \hat{e}\left(\sum_{i=k_1}^{k_2} \beta_{e'_i}(e) H_S(\widetilde{\boldsymbol{Y}}_r(e'_i)), P\right) = \hat{e}\left(H(PID), \sum_{i=1}^{m+n} \tilde{y}^r_i(e) PK_i\right), \quad (19)$$

where each symbol $\tilde{y}^r_i(e)(1 \leqslant i \leqslant m+n)$ is the element in vector $\sum_{i=k_1}^{k_2} \beta_{e'_i}(e)\widetilde{\boldsymbol{Y}}_r(e'_i)$. Eq. (19) can be proofed similarly as that of Eq. (13).

On the other hand, the upper sub-tree is used to perform *generation-based binary verification*, it is constructed as follows:

(1) The leaf node $\langle g, v\rangle (v = r-k, k \leqslant r < k+b)$ is a counterpart of the root node $\langle r, 0, 0\rangle$ of a lower sub-tree. It is associated with a packet-based aggregate signature $\beta_v = \boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$, which is tagged with generation r $(k \leqslant r < k+b)$;
(2) The root $\langle 0, 0\rangle$ is associated with an aggregate signature $\beta_{\langle 0,0\rangle} = \sum_{i=0}^{k-1} \beta_i$. Each inner node $\langle l, v\rangle$ $(l \leqslant g-1)$ is associated with an aggregate signature $\beta_{\langle l,v\rangle} = \sum_{i=k_1}^{k_2} \beta_i$ in the leaf nodes of the sub-tree rooted at $\langle l, v\rangle$, where $\beta_{\langle l,v\rangle} = \sum_{i=k_1}^{k_2} \beta_i = \sum_{i=k_1}^{k_2} \boldsymbol{H}_S \times (\widetilde{\boldsymbol{Y}}_{i+k}(e)), k_1 = 2^{g-l} \cdot v$, and $k_2 = 2^{g-l} \cdot (v+1) - 1$. The authenticity of the aggregate signature $\beta_{\langle l,v\rangle}$ can be verified by checking if

$$\hat{e}(\beta_{\langle l,v\rangle}, P) = \hat{e}\left(\sum_{i=k_1}^{k_2} \boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_{i+k}(e)), P\right) = \hat{e}\left(H(PID), \sum_{i=1}^{m+n} \tilde{y}^r_i(e) PK_i\right), \quad (20)$$

where $\tilde{y}^r_i(e)$ is the element in vector $\sum_{i=k_1}^{k_2} \widetilde{\boldsymbol{Y}}_{i+k}(e)$.

Similar to binary-searching algorithm, searching a BAT is a process that recursively verifies the sub-tree dictated by the current authentication status of aggregate signatures. Consider the first step to verify the aggregate signature at root node $\beta_{\langle 0,0\rangle}$. If the aggregate signature at $\beta_{\langle 0,0\rangle}$ is genuine, all the signatures in the leaf-nodes are authentic. Otherwise, it further verifies the aggregate signatures of the left-child node $\beta_{\langle 1,0\rangle}$ or right nodes $\beta_{\langle 1,1\rangle}$ in the same way, respectively. This binary checking process will be iteratively carried out in *Up-to-Bottom* way until all bogus packets are found.

The performance evaluation of M-BAT is not trivial, which relies on the number of bogus signatures. According to the theoretical analysis for identity-based binary batch verification in [29], the number of time-consuming pairing

operations to check k signatures with r bogus ones is approximately equal to $2(r+1)\log(k/r)+4k+2$ on average.

4. Security analysis

In this section, we will respectively analyze the hash collision, signature forging, and pair-wise byzantine attacks, which is generally related to the batch verification. We assume that the source is always trusted, and the forwarders may not be trusted.

4.1. Hash collision

To thwart the signature scheme, an adversary may either generate a hash collision for the signature $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ or may forge a signature which can pass the verification.

Firstly, we show that even if an adversary is with the knowledge of SK_i $(1 \leqslant i \leqslant m+n)$, generating a hash collision message is still as hard as computing discrete logarithm problem (DLP).

Proposition 1. *For $m+n$ distinct points SK_i $(1 \leqslant i \leqslant m+n)$ on an elliptic curve $\mathbb{E}/\mathbb{F}_q$ contained in a cyclic subgroup of prime order q, given a message $\widetilde{\boldsymbol{Y}}(e) = (\tilde{y}_1(e), \ldots, \tilde{y}_{m+n}(e)) \in \mathrm{F}_q^{m+n}$ with its signature $\boldsymbol{H}_S(\widetilde{Y}(e)) = \sum_{k=1}^{m+n}\{\tilde{y}_k(e)SK_k\}$, generating a hash-collision message $\widetilde{\boldsymbol{Y}}'(e) = (\tilde{y}'_1(e), \ldots, \tilde{y}'_{m+n}(e)) \in \mathbb{F}_q^{m+n}$ from $\widetilde{\boldsymbol{Y}}(e)$ is equivalent to solve a hard DLP problem, where $\widetilde{\boldsymbol{Y}}(e) \neq \widetilde{\boldsymbol{Y}}'(e)$ and $H_S(\widetilde{\boldsymbol{Y}}(e)) = \boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}'(e))$.*

Proof. First, consider the case that $m+n=2$. Let SK_1 and SK_2 be two distinct points of order q on $\mathbb{E}/\mathbb{F}_q$. Given a valid message $\widetilde{\boldsymbol{Y}}(e) = \{\tilde{y}_1(e), \tilde{y}_2(e)\}$ with its signature $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}(e)) = \tilde{y}_1(e)SK_1 + \tilde{y}_2(e)SK_2$, an adversary attempts to generate a hash-collision message $\widetilde{\boldsymbol{Y}}'(e) = \{\tilde{y}'_1(e), \tilde{y}'_2(e)\} (\widetilde{\boldsymbol{Y}}(e) \neq \widetilde{\boldsymbol{Y}}'(e))$ such that $\tilde{y}_1(e)SK_1 + \tilde{y}_2(e)SK_2 = \tilde{y}'_1(e)SK_1 + \tilde{y}'_2(e)SK_2$. This means $(\tilde{y}_1(e) - \tilde{y}'_1(e))SK_1 + (\tilde{y}_2(e) - \tilde{y}'_2(e))SK_2 = \mathbf{O}$. Suppose that $\tilde{y}_1(e) = \tilde{y}'_1(e)$, then $(\tilde{y}_2(e) - \tilde{y}'_2(e))SK_2 = \mathbf{O}$. Since SK_2 is a point of order q, we have $(\tilde{y}_2(e) - \tilde{y}'_2(e)) \equiv 0 \bmod q$, that is, $\tilde{y}_2(e) = \tilde{y}'_2(e)$ in $\mathbb{F}_q$. This contradicts the assumption that $\widetilde{\boldsymbol{Y}}(e) \neq \widetilde{\boldsymbol{Y}}'(e)$ in $\mathbb{F}_q^2$. Furthermore, if we fix $\tilde{y}'_1(e)$ and define $x = \tilde{y}'_2(e)$, the problem becomes to determine x over group $\mathbb{E}/\mathbb{F}_q$ such that $xSK_2 = ((\tilde{y}_1(e) - \tilde{y}'_1(e))SK_1 + \tilde{y}_2(e)SK_2)$. Evidently, this is a hard DLP problem.

For the case that $m+n>2$, given a message $\widetilde{\boldsymbol{Y}}(e) \in \mathbb{F}_q^{m+n}$ with $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}(e)) = \sum_{k=1}^{m+n}\tilde{y}_k(e)SK_k$, the hash-collision message $\widetilde{\boldsymbol{Y}}'(e) \in \mathbb{F}_q^{m+n}$ should satisfy $\widetilde{\boldsymbol{Y}}'(e) \neq \widetilde{\boldsymbol{Y}}(e)$ and $\sum_{k=1}^{m+n}\tilde{y}_k(e)SK_k = \sum_{k=1}^{m+n}\tilde{y}'_k(e)SK_k$, which also means $\sum_{k=1}^{m+n}(\tilde{y}_k(e) - \tilde{y}'_k(e))SK_k = \mathbf{O}$. Similar to the case when $m+n=2$, it is easy to proof that in order to satisfy $\sum_{k=1}^{m+n}(\tilde{y}_k(e) - \tilde{y}'_k(e))SK_k = \mathbf{O}$, there exist at least two distinct items in $\widetilde{\boldsymbol{Y}}(e)$ and $\widetilde{\boldsymbol{Y}}\prime(e)$. Without loss of generality, let the two items be $\tilde{y}_i(e) \neq \tilde{y}'_i(e)$ and $\tilde{y}_j(e) \neq \tilde{y}'_j(e)$. Consider that $SK_i = s_iH(PID)$ $(1 \leqslant i \leqslant m+n)$, where secret s_i are randomly chosen from $\mathbb{F}_q$, we have

$$(\tilde{y}_i(e) - \tilde{y}'_i(e))SK_i + \sum_{k=1,k\neq i}^{m+n}\{(\tilde{y}_k(e) - \tilde{y}'_k(e))s_2^{-1}s_kSK_j\} = O,$$

where the coefficients $r_k = s_2^{-1}s_k$ are unknown to the random oracle for the hash-collision algorithm. Fixing items $\{\tilde{y}'_k(e) | 1 \leqslant k \leqslant m+n, k \neq i\}$ and define $x = \tilde{y}'_i(e)$, the problem becomes how to determine x over group $\mathbb{E}/\mathbb{F}_q$ such that $xSK_i = (\tilde{y}_i(e)SK_i + \sum_{k=1,k\neq i}^{m+n}((\tilde{y}_k(e) - \tilde{y}'_k(e))s_2^{-1}s_kSK_j))$, which is also a hard DLP problem. □

4.2. Signature forging and random forgery attacks

In this sub-section, we will show that forging a signature is at least as hard as solving the so-called computational Diffie–Hellman problem on the elliptic curve and computing discrete logarithms.

Signature forging: A smart adversary may attempt to derive the identity-aware signature keys $SK_i (1 \leqslant i \leqslant m+n)$ from the transmitted packets, since each signature $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ is a linear equation with $m+n$ unknown keys SK_i, i.e.,

$$\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) = \sum_{j=1}^{m+n}\tilde{y}_j^r(e)SK_j = \sum_{j=1}^{m+n}(\tilde{y}_j^r(e)s_jH(PID)).$$

However, as shown in Eq. (6), since the pseudo identity PID of a source will be altered as $PID = h^k(ID)$ after it sends $\lfloor (m+n-1)/n \rfloor \cdot n$ linearly independent packets, the adversary can only collect at most $\lfloor (m+n-1)/n \rfloor \cdot n$ linear independent packets in term of key $SK_i (1 \leqslant i \leqslant m+n)$. Thus, it cannot derive the identity-aware signature keys SK_i by solving the $\lfloor (m+n-1)/n \rfloor \cdot n$ linear independent equations.

Therefore, as far as a group of signature keys SK_i with a pseudo identity PID are concerned, each signature $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) = \sum_{k=1}^{m+n}\{\tilde{y}_k^r(e)SK_k\}$ can be actually regarded as a "one-time" identity-based signature. Without the private key SK_i $(1 \leqslant i \leqslant m+n)$, it is infeasible to forge a valid signature. Because of the NP-hard computation complexity problem of Diffie-Hellman in $\mathbb{G}$, it is difficult to derive the private keys SK_i $(1 \leqslant i \leqslant m+n)$ by using $\{PID, H(PID), P\}$ and PK_i $(1 \leqslant i \leqslant m+n)$. At the same time, since $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e)) = \sum_{k=1}^{m+n}\{\tilde{y}_k^r(e)SK_k\}$ is a Diophantine equation, with the knowledge of $\boldsymbol{H}_S(\widetilde{\boldsymbol{Y}}_r(e))$ and $\tilde{y}_k^r(e)$ $(1 \leqslant k \leqslant m+n)$, it is still difficult to get the private keys SK_i $(1 \leqslant i \leqslant m+n)$. Therefore, forging the "one-time" signature by attempting to derive SK_i is computationally difficult.

Random forgery attacks: In [38], Johnson et al. conclude that "*In any additive signature scheme on the lattice $L = (\mathbb{Z}/m\mathbb{Z})^d$, if one can get signature $Sig(x_1), \ldots, Sig(x_d)$, where $x_1, \ldots, x_d$ are a basis for L, then one can succeed at any random forgery.*" Therefore, knowing the signatures on a basis is useless to forge a message, if computing the representation of a given message in that basis is hard.

In previous homomorphic signature schemes for network coding [5,12,13,36], the signature keys keep invariant. However, in our scheme, the *identity-aware signature keys* $SK_i = s_iH(PID)$ $(1 \leqslant i \leqslant m+n)$, as the basis of a $(m+n)$-dimension signature, is dynamically alterable, since the pseudo identity PID will be altered as $PID = h^k(ID)$ after the source sends $\lfloor (m+n-1)/n \rfloor \cdot n$ linearly independent packets, as shown in Eq. (6). Therefore, an

adversary can only collect at most $\lfloor (m+n-1)/n \rfloor \cdot n$ linearly independent signature $\boldsymbol{H}_S(\tilde{\boldsymbol{Y}}_r(e)) = \sum_{k=1}^{m+n} \{\tilde{y}_k^r(e) SK_k\}$ in term of $SK_i (1 \leqslant i \leqslant m+n)$, thus it is difficult to derive the dynamic signature keys SK_i by solving such $\lfloor (m+n-1)/n \rfloor \cdot n$ linearly independent equations. In other words, it is most unlikely for an adversary having signatures on a basis of SK_i $(1 \leqslant i \leqslant m+n)$ to forge a message, since it is computationally hard to derive the linear representation of a given message.

4.3. Pair-wise byzantine attacks

Generally, the batch verification may be exposed to a specific attack, called the pair-wise byzantine attack [5]. For instance, an adversary with any two correct packets M_i $(i = 1, 2)$ is easy to create the following two corrupted packets, $M_1' = M_1 + \varepsilon$ and $M_2' = M_2 - \varepsilon$. When they are verified in batch, the verifier will fail to capture the corrupted packets due to $M_1' + M_2' = M_1 + M_2$.

To address such byzantine attack in RSA-based batch verification, a small exponent test method [23] is introduced by multiplying each message with a random coefficient, respectively. Similarly, the proposed M-BAT algorithm addresses this attack by generating each new encoding vector with random local vectors. Thus, an adversary can only launch a successful pair-wise attack, if it can create two packets that after being multiplied by random coefficients will counteract each other, which is very infeasible. For example, for a successful pair-wise attack, an adversary with two correct messages M_i $(i = 1, 2)$ is required to forge two messages M_1' and M_2' satisfying $w_1 M_1' + w_2 M_2' = w_1 M_1 + w_2 M_2$. However, due to the randomness of coefficients w_1 and w_2, it is very unlikely to generate such two message M_1' and M_2'.

4.4. Discussion

As shown in the security analysis, the signatures on a set of vectors $(v_1, \ldots, v_m)$ can be used to generate a valid signature on any vectors from the vector space $V = \mathbf{span}(v_1, \ldots, v_m)$. Therefore, the proposed schemes can efficiently thwart pollution attacks by signing linear subspaces in the sense that a signature σ on a subspace V authenticates exactly those vectors in V.

Unlike polluted packets, an adversary may also arbitrarily forge non-innovative packets by launching a special replay attack, called entropy attack [5]. Such non-innovative packets preserve the linear algebraic constraints on the original packets and the appended encoding vectors. However, these packets with no new coding information will reduce the decoding opportunities at sinks and the overall throughput rate. How to efficiently filter out such non-innovative packets is another important issue to be explored [36].

5. Performance evaluation

In this section, we evaluate the proposed scheme by simulation, and compare it with Charles et al.'s scheme and Yu et al.'s scheme in terms of computation and communication overheads, respectively.

5.1. Computation overhead

We define the computation cost of the primarily cryptographic operations as follows. Let C_{me} denote the time cost to perform one modular exponent operation, C_{mul} the time cost to perform a point multiplication over an elliptic curve, C_{mtp} the time of a MapToPoint hash operation, and C_{par} the time of a pairing operation. We neglect all the trivial operations such as addition operations for the sake of simplicity.

Table 1 shows the combination of the dominant operations of the three signature schemes in terms of signing or verifying an encoded packet, respectively. The proposed scheme has the optimal computation complexity on average, considering that the point multiplication over an elliptic curve (160-bits) has a lower cost than modular exponential operations (1024-bits) with the same security level. Specifically, when signing a packet, both the proposed scheme and Charles et al.'s scheme need approximately $(m+n)C_{mul}$, while Yu et al.'s scheme requires $(m+n+1)C_{me}$. To verify a packet, the proposed scheme requires $C_{mtp} + 2C_{par} + (m+n)C_{mul}$ and Charles et al.'s scheme needs $(m+n)C_{par} + (m+n+1)C_{mul}$, whereas Yu et al.'s scheme involves $(m+n+2)C_{me}$. Clearly, the number of time-consuming pairing operations in the proposed scheme is remarkably reduced to two from $(m+n)C_{par}$, due to the adoption of the identity-based batch verification.

Note that since Yu et al.'s scheme and Charles et al.'s scheme are not identity-based signature schemes, additional one C_{me} or C_{mul} operations are required to verify the public key's certificate.

Table 2 shows the comparisons of computation complexity in term of different optimized verification policies. Both the basic verification and pre-computation verification have the similar performance. The packet-based batch verification for authenticating $|in(v)|$ signatures and the generation-based verification for authenticating $\varepsilon|in(v)|$ signatures can significantly reduce the verification cost in term of the normalized verification cost per packet, with the result of $\{(m+n+|in(v)|)C_{mul} + 2C_{par}\}/|in(v)|$ and $\{(m+n+\varepsilon|in(v)|)C_{mul} + 2C_{par}\}/(\varepsilon|in(v)|)$, respectively.

Table 1
Comparisons of computation overhead.

Scheme	Signing	Verifying
Yu et al.'s scheme	$(m+n+1)C_{me}$	$(m+n+2)C_{me}$
Charles et al.'s scheme	$(m+n)C_{mul}$	$(m+n)C_{par} + (m+n+1)C_{mul}$
The proposed basic scheme	$(m+n)C_{mul}$	$C_{mtp} + 2C_{par} + (m+n)C_{mul}$

Table 2
Computation comparisons of optimized policies.

Optimized policies	Verification cost	Normalized cost per packet
Basic verification	$2C_{par} + (m+n)C_{mul}$	$2C_{par} + (m+n)C_{mul}$
PRE verification	$C_{par} + (m+n)C_{me}$	$C_{par} + (m+n)C_{me}$
PB verification	$(m+n+\lvert in(v)\rvert)C_{mul} + 2C_{par}$	$\frac{(m+n+\lvert in(v)\rvert)C_{mul}+2C_{par}}{\lvert in(v)\rvert}$
GB verification	$(m+n+\varepsilon\lvert in(v)\rvert)C_{mul} + 2C_{par}$	$\frac{(m+n+\varepsilon\lvert in(v)\rvert)C_{mul}+2C_{par}}{\varepsilon\lvert in(v)\rvert}$

Note: (1) PRE: Pre-computation verification with one signature; (2) PB: Packet-Based verification with $|in(v)|$signatures; (3) GB: Generation-Based verification with $\varepsilon|in(v)|$ signatures.

Note that the number of MapToPoint hash operations $H(PID)$ is only one, so it is ignored in Table 2.

Packet verification is the primary workload of nodes (forwarders or sinks). Efficient verification approaches can eliminate the performance bottleneck and be helpful to achieve the optimal rate when a source sends packets. To compare the verification cost of the three schemes, we first give the benchmarks of the primitive cryptographic operations on Intel Core™ 2 Duo 1.83 GHz Linux machine: $C_{mul} = 0.75$ ms,$C_{mtp} = 1.18$ ms,$C_{par} = 2.75$ ms, and $C_{me} = 0.83$ ms. We also implement a super-singular curve of embedded degree $k = 6$ over $\mathbb{F}_{3^{97}}$ with C program. The choice of the elliptic curve can certainly influence the overall computation cost of the proposed scheme. For example, Barreto et al. [30] reduce the cost of generating a BLS signature [21] on a super-singular curve of embedded degree $k = 6$ over $\mathbb{F}_{3^{97}}$, whereas the BLS scheme uses a super-singular curve $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^l}$ where l is a positive exponent.

Fig. 3 shows the relationship between the verification cost and the number of symbols per packet $(m = 2n)$. The verification cost of different schemes approximately increases linearly along with the growth of the number of symbols per packet. The verification cost of Charles et al.'s scheme is always the largest. The verification cost of the Zhu et al.'s scheme is close to that of the basic verification scheme, while the verification cost of the two optimized methods (the packet-based or generation-based batch verification) is much faster than the other two schemes. Evidently, the packet-based or generation-based batch verification can significantly reduce the verification delay in term of normalized verification cost. In Fig. 3, we only show the verification cost for $\varepsilon = 2$ and $|in(v)| = 2$. As shown in Table 2, *the normalized verification cost is approximately inverse proportional to value*$|in(v)|$ *or* ε. Hence, the proposed scheme effectively eliminates the computation workload at each forwarder, and can achieve the lower packet loss ratio when the network traffic load increases, due to the identity-based batch verification.

In adverse scenario with bogus packets, the batch verification is disabled. We can adopt the M-BAT algorithm to address the robustness issue. The performance of BAT has been discussed in [29].

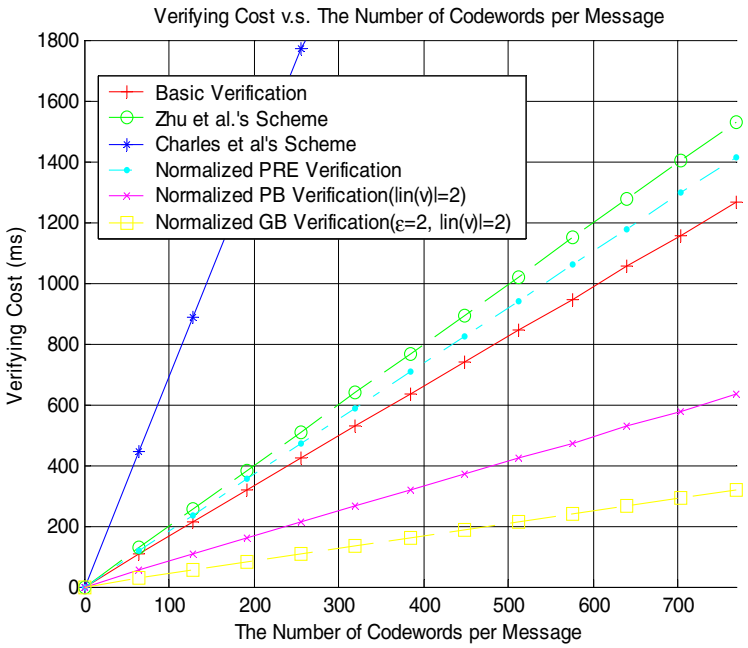

Fig. 3. Verification cost vs. number of codes per encoded message $(m = 2n)$.

Table 3
Comparisons of transmission overhead per packet.

Yu et al.'s scheme	Charles et al.'s scheme	The proposed scheme
(128 + 675) bytes	(22 + 125) bytes	(44 + 22) bytes

5.2. Communication overhead

Communication overhead contains a signature and a certificate appended to the original packet, while the packet itself is not considered. Table 3 shows the comparison of the three schemes in terms of communication overhead. Yu et al.'s scheme can be considered as a RSA-based signature, and the size of its signature is 128 bytes. The signature of the proposed signature scheme and Charles et al.'s scheme is similar to that of a BLS-based aggregate signature scheme. Since the size of signature in BLS scheme [21] is equal to that of the ECDSA signature of IEEE1069.2 [31], the size of a signature in the proposed scheme and Charles et al.'s scheme is equal to that of the ECDSA, or 22 bytes.

In addition, we should take the certificate into consideration, which incurs extra communication overhead. In Charles et al.'s scheme or Yu et al.'s scheme, a certificate must be transmitted along with the signature. If we adopt the certificate in IEEE 1609.2 Standard [31], which has 125 bytes in length, the total transmission overhead of Charles et al.'s scheme is 22 + 125 bytes, as shown in Table 3. Yu et al.'s scheme also has to incorporate a certificate in the packet, which is 675 bytes long in the case of using RSA certificate according to X.509-v3 Standard [32]. The total transmission overhead of Yu et al.'s scheme is 128 + 675 bytes. In contrast, the proposed scheme does not need any certificate due to the adoption of identity-based cryptography; instead, only a 44 bytes short-length identity is sent, i.e., $|PID| = |ID_1| + |ID_2| = 44$ bytes. Thus, the total transmission cost of the proposed scheme is 44 + 22 bytes.

6. Related works

Security issue in network coding has attracted increasing attentions recently. To secure network coding against pollution attacks, several efficient solutions have been appeared, which can be primarily divided into two categories from the view of cryptography.

In [14], Ho et al. propose a *non-cryptography-based scheme* on how distributed randomized network coding can be extended to detect Byzantine modification attacks without the use of cryptographic functions. For the scheme, a computation-efficient hash value is embedded into each packet. The sinks can use the hashes to detect integrity of the corresponding packets and with high probability, when there are Byzantine attacks. However, the sinks cannot recover the source packets correctly, even though the polluted packets have been detected. In [15], Jaggi et al. introduce an information-theoretically secure network coding, which can efficiently tolerate the presence of Byzantine adversaries. The basic idea is to embed extra parity information into the source packets so that the sinks can use such information to recover the source packets when suffering Byzantine attacks. To achieve optimal rate, Jaggi et al. present several polynomial-time algorithms, which can efficiently target adversaries with different attacking capabilities even without any knowledge of the topology. However, similar to Ho's scheme, Jaggi's scheme can only allow the sinks, instead of the forwarders, to detect Byzantine attacks. Since it cannot drop the junk packets en-route, the scheme is unsuitable for resource-constrained networks. Following Jaggi et al. scheme, Wang et al. [33] introduce a broadcast-mode transformation for network coding, which efficiently impedes the influence of potential adversaries by limiting them to a single transmission opportunity per generation. With a sufficient diversity of internally-disjoint paths from source to sink(s), the multicast capacity may not be greatly affected by this transformation. In addition, combined with error-control coding, this approach may be effective in dealing with adversaries, particularly in such application scenarios, where cost-prohibitive approaches may be infeasible.

Cryptography-based schemes primarily include homomorphic hash scheme [5,34,36], homomorphic signature scheme [12,13], and secure random checksum scheme [5]. All of these techniques try to detect a polluted packet before it gets mixed into the buffer of forwarders. In [34], Krohn et al. present a practical security scheme for peer-to-peer content distribution by using homomorphic hashing function, which enables a downloader to efficiently perform on-the-fly verification of erasure-encoded blocks, where each block is linear combination of original file blocks. Gkantsidis et al. [5] extend Krohn et al.'s approach and present a homomorphic hashing scheme for securing peer-to-peer file distribution via network coding against pollution attacks. The scheme remarkably reduces the cost of verifying blocks on-the-fly while efficiently preventing the propagation of malicious blocks. Due to the homomorphic hash function, the hash value of each linear encoded block can be efficiently calculated as the homomorphic hash combination of the original file blocks. However, Gkantsidis et al.'s scheme needs an extra secure channel for the source to distribute its hashes to all nodes before sending the source blocks.

Charles et al. [12] design a different homomorphic signature scheme based on Weil pairing. The signature is calculated based on augmented packet which covers both the packet and its encoding vector. Hence, the scheme requires no secure channel. However, this scheme relies on time-consuming pairing computations over elliptic curves, and cannot efficiently support batch verification [24]. Experimental results show that Charles et al.'s scheme is much slower than Gkantsidis et al.'s scheme in terms of packet verification [13]. Following [5,12], Yu et al. [13] propose an efficient signature-based scheme, which inherits the basic framework in [12] and the homomorphic signature in [5]. It provides comparable computation-efficiency with Gkantsidis et al.'s scheme, while offering similar security of Charles et al.'s scheme.

Zhao et al. [39] also present a signature scheme for network coding by authenticating the spanned vector subspace $V = \mathbf{span}(v_1, \ldots, v_m)$. With the public signature information, a node can verify if $w \in V$ for any received packet w. One of the significant drawbacks in this scheme

is that the size of both the signature information and the public keys is at least the square root of the file size. Moreover, the scheme is not efficient for distributing multiple files with the same public key, which significantly impairs the system scalability. Finally, to pre-compute the public signature information, the source is required to buffer the entire file in advance, which enables the scheme not suitable for transmitting on-the-fly streaming data.

Recently, Fan et al. [40] present an efficient privacy-preserving scheme against traffic analysis attacks in Network Coding. To thwarting the entropy attacks, Jiang et al. [36] propose a self-adaptive probabilistic subset linear-dependency test algorithm for securing network coding, which can fast filter out the resultant packets from entropy attacks and thus efficiently improve the data availability.

7. Conclusions

In this paper, we have proposed an efficient security scheme for network coding against pollution attacks and random forgery attacks. Two identity-based verification techniques, packet-based and generation-based batch verification, enable a node to verify multiple received signatures synchronously and require neither separate certificates nor extra secure channels. In addition, our scheme provides source authentication via the forward one-way hash identity-chain. We have demonstrated that the proposed scheme can achieve high efficiency and security in packet signature and filtering, and meet the important and emerging requirements for securing network coding. Our future works include efficient privacy-preservation signature scheme design in Xor-based wireless network coding.

Acknowledgements

This work is financially supported by the Bell University Laboratories (BUL), and this research has been supported in part by the NSFC under Contracts No. 60970101 and No. 60872055.

References

[1] Y. Zhu, B. Li, J. Guo, Multicast with network coding in application-layer overlay networks, IEEE Journal on Selected Areas in Communications, January 2004.
[2] D. Petrovic, K. Ramchandran, J. Rabaey, Overcoming untuned radios in wireless networks with network coding, IEEE Transactions on Information Theory 52 (6) (2006) 2649–2657.
[3] S. Katti, H. Rahul, D. Katabi, W.H.M. M'edard, J. Crowcroft, Xors in the air: practical wireless network coding, in: Proceedings of ACM SIGCOMM, 2006.
[4] C. Gkantsidis, P. Rodriguez, Network coding for large scale file distribution, in: Proceedings of IEEE INFOCOM, 2005.
[5] C. Gkantsidis, P. Rodriguez, Cooperative security for network coding file distribution, in: Proceedings of IEEE INFOCOM, 2006.
[6] S. Deb, C. Choute, M. Medard, R. Koetter, How good is random linear coding based distributed networked storage? in: Proceedings of NetCod 2005.
[7] K. Jain, L. Lovasz, P.A. Chou, Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding, in: Proceedings of ACM Symposium on Principles of Distributed Computing, 2005.
[8] R. Ahlswede, N. Cai, S. Li, R. Yeung, Network information flow, IEEE Transactions on Information Theory 46 (4) (2000) 1204–1216.
[9] S. Li, R. Yeung, N. Cai, Linear network coding, IEEE Transactions on Information Theory 49 (2) (2003) 371–381.
[10] Z. Li, B. Li, Network coding: the case of multiple unicast sessions, in: Proceedings of 42th annual allerton conference on communication, control, and computing, 2004.
[11] D.S. Lun, M. M'edard, R. Koetter, Network coding for efficient wireless unicast, in: Proceedings of 2006 International Zurich Seminar on Communications (IZS'06), Zurich, Switzerland, 2006.
[12] D. Charles, K. Jian, K. Lauter, Signature for Network Coding, Technique Report MSR-TR-2005-159, Microsoft, 2005.
[13] Z. Yu, Y. Wei, B. Ramkumar, Y. Guan, An efficient signature-based scheme for securing network coding against pollution attacks, in: Proceedings of IEEE INFOCOM, 2008.
[14] T. Ho, B. Leong, R. Koetter, M. M'eard, M. Effros, D. Karger, Byzantine modification detection in multicast networks using randomized network coding, in: Proceedings of 2004 IEEE International Symposium on Information Theory (ISIT), 2004.
[15] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. M'eard, Resilient network coding in the presence of byzantine adversaries, in: Proceedings of IEEE INFOCOM, 2007.
[16] A. Chou, Y. Wu, Network coding for the internet and wireless networks, MSR-TR-2007-70, Microsoft Research, 2007.
[17] T. Ho, M. Médard, R. Koetter, D.R. Karger, M. Effros, J. Shi, B. Leong, A random linear network coding approach to multicast, IEEE Transactions on Information Theory 52 (October) (2006).
[18] P. Sanders, S. Egner, L. Tolhuizen, Polynomial time algorithms for network information flow, in: Proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures, 2003.
[19] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, Proceedings of Crypto, LNCS 2139 (2001) 213–229.
[20] A. Miyaji, M. Nakabayashi, S. Takano, New explicit conditions of elliptic curve traces for FR-reduction, IEICE Transactions on Fundamentals 5 (2001) 1234–1243.
[21] D. Boneh, B. Lynn, H. Shacham, Short Signatures from the Weil pairing, Journal of Cryptology 17 (4) (2004) 297–319.
[22] A. Fiat, Batch RSA, Proceedings of CRYPTO, LNCS 435 (1989) 175–185.
[23] J. Pastuszak, D. Michatek, J. Pieprzyk, J. Seberry, Identification of bad signatures in batches, Proceedings of PKC'00, LNCS 3958 (2000) 28–45.
[24] J.C. Cha, J.H. Cheon, An identity-based signature from gap Diffie–Hellman groups, Proceedings of Public Key Cryptography (PKC) (2003) 18–30.
[25] F. Zhang, R. Safavi-Naini, W. Susilo, Efficient verifiably encrypted signature and partially blind signature from bilinear pairings, Proceedings of Indocrypt, LNCS 2904 (2003) 191–204.
[26] H. Yoon, J.H. Cheon, Y. Kim, Batch verification with ID-based signatures, Proceedings of Information Security and Cryptology (2004) 233–248.
[27] J. Camenisch, S. Hohenberger, M. Pedersen, Batch verification of short signatures, Proceedings of EUROCRYPT, LNCS 4514 (2007) 246–263.
[28] J. Camenisch, A. Lysyanskaya, Signature schemes and anonymous credentials from bilinear maps, Proceedings of CRYPTO, LNCS 3152 (2004) 56–72.
[29] Y. Jiang, M. Shi, X. Shen, C. Lin, BAT: a robust signature scheme for vehicular communications using binary authentication tree, IEEE Transactions on Wireless Communications 8 (4) (2009) 1974–1983.
[30] P. Barreto, H. Kim, B. Lynn, M. Scott, Efficient algorithms for pairing-based cryptosystems, Proceedings of CRYPTO'02, LNCS 2442 (2002) 354–368.
[31] IEEE Standard 1609.2, IEEE Trial-Use Standard for Wireless Access in Vehicular Environments, Security Services for Applications and Management Messages, July, 2006.
[32] R. Housley, W. Polk, W. Ford, D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF RFC 3280, April 2002.
[33] D. Wang, D. Silva, F.R. Kschischang, Constricting the adversary: a broadcast transformation for network, in: Proceedings of 45th Annual Allerton Conference on Communication, Control and Computing, 2007.
[34] M. Krohn, M. Freeman, D. Mazieres, On-the-fly verification of rateless erase codes for efficient content distribution, in: Proceedings of IEEE Symposium on Security and Privacy, 2004.
[35] P. Chou, Y. Wu, K. Jain, Practical network coding, in: Proceedings of Allerton Conference on Communication, Control, and Computing, 2003.
[36] Y. Jiang, Y. Fan, X. Shen, C. Lin, A self-adaptive probabilistic packet filter scheme against entropy attacks in network coding, Computer Networks, Elsevier, 2009.

[37] C. Cocks, An identity based encryption scheme based on quadratic residues, in: Proceedings of the 8th IMA International Conference on Cryptography and Coding, 2001.
[38] R. Johnson, D. Molnar, D. Song, D. Wagner, Homomorphic signature schemes, Proceedings of RSA, Cryptographer's Track. LNCS 2271 (2002).
[39] F. Zhao, T. Kalker, M. Medard, K.J. Han, Signatures for content distribution with network coding, in: Proceedings of IEEE ISIT, 2007.
[40] Y. Fan, Y. Jiang, H. Zhu, X. Shen, An efficient privacy-preserving scheme against traffic analysis attacks in network coding, in: Proceedings of IEEE INFOCOM, 2009.

Yixin Jiang is an associate professor in Tsinghua University. In 2007–2009, he was a Post Doctorial Fellow with University of Waterloo. He received the Ph.D degree (2006) from department of Computer Science, Tsinghua University, China. In 2005, he was a Visiting Scholar with the Department of Computer Sciences, Hong Kong Baptist University. He has served as the Technical Program Committee (TPC) member for network conferences, such as IEEE ICCCN, IEEE GLOBECOM, IEEE ICC, IEEE WCNC, etc. His current research interests include wireless network security, trusted computing and network coding.

Haojin Zhu received his B.Sc. degree (2002) from Wuhan University (China) and his M.Sc. (2005) degree from Shanghai Jiao Tong University (China), both in computer science. He is currently working toward his Ph.D. degree in the electrical and computer engineering at the University of Waterloo, Waterloo, Canada. His current research interests include wireless network security and applied cryptography. He is the recipient of best paper awards of IEEE ICC 2007 – Computer and Communications Security Symposium and Chinacom 2008 – Wireless Communication Symposium.

Minghui Shi received a B.S. degree in 1996 from Shanghai Jiao Tong University, China, and an M.S. degree and a Ph.D. degree in 2002 and 2006, respectively, from the University of Waterloo, Ontario, Canada, all in electrical engineering. He was a NSERC Postdoctoral Fellow at McMaster University, Ontario, Canada between 2007 and 2008. He is currently a visiting scientist with the University of Waterloo. His current research interests include security protocol and architecture design, authentication and key distribution for ad hoc/sensor networks, heterogeneous networks interworking, delay tolerant networks, vehicular networks, etc.

Xuemin (Sherman) Shen (IEEE M'97-SM'02-FE'09) received the B.Sc.(1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. He is a Professor and University Research Chair, and the Associate Chair for Graduate Studies, Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on mobility and resource management in interconnected wireless/wired networks, UWB wireless communications systems, wireless security, and vehicular ad hoc networks and sensor networks. He is a co-author of three books, and has published more than 300 papers and book chapters in wireless communications and networks, control and filtering. He serves as the Technical Program Committee Chair for IEEE Globecom' 07, General Co-Chair for Chinacom'07 and QShine'06, the Founding Chair for IEEE Communications Society Technical Committee on P2P Communications and Networking. He also serves as a Founding Area Editor for IEEE Transactions on Wireless Communications; Editor-in-Chief for Peer-to-Peer Networking and Application; Associate Editor for IEEE Transactions on Vehicular Technology; KICS/IEEE Journal of Communications and Networks, Computer Networks; ACM/Wireless Networks; and Wireless Communications and Mobile Computing (Wiley), etc. He has also served as Guest Editor for IEEE JSAC, IEEE Wireless Communications, and IEEE Communications Magazine. Dr. Shen received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004 and 2008 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 from the Faculty of Engineering, University of Waterloo. Dr. Shen is a registered Professional Engineer of Ontario, Canada.

Chuang Lin is a professor and the head of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from Tsinghua University in 1994. In 1985–1986, he was a Visiting Scholar with the Department of Computer Sciences, Purdue University. In 1989–1990, he was a Visiting Research Fellow with the Department of Management Sciences and Information Systems, University of Texas at Austin. In 1995–1996, he visited the Department of Computer Science, Hong Kong University of Science and Technology. His current research interests include computer networks, performance evaluation, network security, logic reasoning, and Petri net and its applications. He has published more than 200 papers in research journals and IEEE conference proceedings in these areas and has published three books. He is an IEEE senior member and the Chinese Delegate in IFIP TC6. He serves as the General Chair, ACM SIGCOMM Asia workshop 2005; the Associate Editor, IEEE Transactions on Vehicular Technology; and the Area Editor, Journal of Parallel and Distributed Computing.