Peer-to-Peer Netw. Appl. (2014) 7:572–589
DOI 10.1007/s12083-013-0239-x

An efficient ECC-based mechanism for securing network coding-based P2P content distribution

Heng He · Ruixuan Li · Zhiyong Xu · Weijun Xiao

Received: 30 December 2011 / Accepted: 30 September 2013 / Published online: 10 November 2013
© Springer Science+Business Media New York 2013

Abstract Network coding has been demonstrated to be able to improve the performance of P2P content distribution. However, it is vulnerable to pollution attacks where malicious peers can flood the network with corrupted blocks easily, leading to substantial performance degradation. Moreover, existing corruption detection schemes for network coding are not well suited to P2P systems. Effective scheme to detect the corruption and identify the attacker is required to thwart such attacks. In this paper, we propose an efficient ECC-based mechanism for securing network coding-based P2P content distribution, namely ESNC, which includes an efficient network coding signature scheme and an identity-based malicious peer identification scheme. The two schemes cooperate to thwart pollution attacks on network coding effectively in P2P networks, not only detecting corrupted blocks on-the-fly efficiently, but also precisely identifying all the malicious peers quickly. ESNC is mainly based on elliptic curve cryptography (ECC) and can provide high level of security. It incurs significantly less computation and communication overheads than other comparable state-of-the-art schemes for P2P systems. ESNC can work with arbitrary topologies, as it is the case in P2P networks. Security analysis demonstrates that ESNC can resist hash collision attacks, signature forgery attacks, and collusion attacks with arbitrary number of colluding malicious peers. Simulation results show that ESNC effectively limits the corruption spread and identifies all the malicious peers in a short time under different practical settings.

Keywords Peer-to-peer · Network coding · Content distribution · Pollution attack · Attacker identification · Elliptic curve cryptography

H. He · R. Li (✉)
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, People's Republic of China
e-mail: rxli@hust.edu.cn

H. He
e-mail: henghe@smail.hust.edu.cn

Z. Xu
Department of Mathematics and Computer Science, Suffolk University, Boston, TX, USA
e-mail: zxu@mcs.suffolk.edu

W. Xiao
Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA, USA
e-mail: wxiao@vcu.edu

1 Introduction

Peer-to-peer (P2P) overlay networks have proven to be a very successful distributed architecture for content distribution. However, P2P content distribution systems often suffer from low efficiency which decreases their overall performance [1]. Such inefficiency often arises from the facts that there is relatively no central scheduler that decides how content should propagate through the network; peers make decisions only with local information in a large distributed environment; and peers are inherently dynamic. These facts may cause some content blocks to become rare or even unavailable when the original peers (referred to as seeds) are no longer available, which will affect the availability of the content.

Network coding was initially proposed to maximize the network throughput and robustness to link failures [2] in information dissemination, and has since received extensive research attention. It is a promising approach in many practical network applications, such as traditional multicast or broadcast networks [3], wireless sensor networks [4, 5], and P2P networks [1, 6–11]. The essence of network coding is that information to be transmitted in a communication session can be encoded rather than simply forwarded by the participating

peers. Recent studies [1, 10, 12] have demonstrated that network coding can bring great benefit for P2P content distribution, since it can improve the resilience to peer churn, decrease bandwidth costs, and achieve high efficient distribution. Network coding distributes encoded content blocks rather than original content blocks, and all encoded blocks are equivalent and useful to any peer, thus the need for selecting blocks and locating rare blocks is eliminated.

However, these main advantages of network coding are applicable in a P2P network only consisting of trustworthy peers, which is not the case in many realistic scenarios. In large scale P2P distributed systems, there exists the possibility that some peers are malicious and may alter and corrupt the encoded content blocks. These malicious peers inject corrupted blocks which do not carry valid information in the network to jam the download. The receiving peer will obtain corrupted blocks, and, if the receiving peer uses these blocks to produce new encoded blocks, it will inject more corrupted blocks in the network involuntarily. As a result, a single corrupted block can rapidly pollute the network. A receiving peer may discover after downloading the entire file that it was receiving corrupted blocks from malicious peers and cannot decode the file. Since each peer produces unique encoded blocks which cannot be signed by the server individually, commonly used methods for protecting the integrity of each block by using digital signatures or hashes do not work with network coding.

The network-coding pollution attack in P2P network can be defined as that a malicious peer can inject corrupted blocks, which do not carry valid information, to its downstream peers, and further contaminate the entire downstream, preventing proper decoding. Network coding-based content distribution is vulnerable to such kind of attack. The attack can cause substantial performance degradation of the distribution network, wasting bandwidth in distributing corrupted content, and increasing the time to download the entire file. Therefore, it is critical to design mechanisms to thwart such attacks in P2P networks.

Corruption detection schemes for network coding have been well studied, in which peers verify encoded blocks on-the-fly to prevent downloading corrupted blocks. Homomorphic hashing functions have first been introduced by Krohn et al. [13] to allow intermediate peers to verify blocks on-the-fly that are encoded at the source using rateless codes. However, homomorphic hash functions are computationally expensive and if each peer verifies all incoming blocks before using them, the computation overheads may be very large. Gkantsidis et al. [6] proposed a cooperative security scheme based on homomorphic hashing technology for network coding in P2P networks. The scheme reduces the computational time of the homomorphic hashes by only requiring peers to verify blocks probabilistically and makes peers cooperate to protect themselves against malicious peers by alerting affected peers when a corrupted block is discovered. However, the scheme cannot identify and remove malicious peers, which means attackers can continuously generate corrupted blocks that will cause more computation and communication overheads. Several other schemes [14–22] have also been proposed to thwart pollution attacks on network coding, but they are generally rather expensive or not applicable to P2P systems. These schemes either incur high computation overheads [15–17], cause relatively high extra transmission overheads [6, 15], endure weak scalability [18, 20, 21], work only on fixed, known topologies [19], suffer from collusion attacks [14, 18, 22], or provide low level of security [14, 18, 19] compared to homomorphic hashing. Compared to the research on corruption detection schemes, identifying attackers has received much less attention. However, attacker identification is a more desirable and efficient scheme because it addresses the pollution attack at its root, not just detecting corrupted blocks. Therefore, it prevents network resources, such as bandwidth and CPU time from being continually wasted on dealing with corrupted blocks from malicious peers.

These aforementioned deficiencies and the new research direction motivate us to explore more efficient schemes. In this paper, we propose an efficient ECC-based mechanism for Securing Network Coding-based P2P content distribution, namely ESNC. ESNC does not require any special or fixed topologies, and can support systems with any size. It is an integrated mechanism because it consists of both an efficient network coding signature scheme and an identity-based malicious peer identification scheme. The two schemes cooperate to thwart pollution attacks on network coding effectively in P2P systems. More specifically, we present a network coding signature scheme, using which every peer can verify encoded blocks on-the-fly efficiently. The scheme is designed on elliptic curve cryptography (ECC) [23] and can achieve both high security and performance. To further optimize computational performance, we employ batch verification approach in the scheme, by which a peer can quickly verify multiple encoded blocks in batch such that the total verification costs can be dramatically reduced. Furthermore, we present a malicious peer identification scheme, which can precisely pinpoint all the malicious peers and eliminate them from the network quickly. To achieve rapid identification of malicious peers, in our scheme, we introduce a fast block verification approach based on the property of the orthogonal space of vectors, and present a lightweight block signature approach which holds every peer accountable for the blocks it sends out. We also evaluate ESNC through detailed analysis and simulations, and results demonstrate the high security and performance of the proposed schemes.

ESNC is a unique and complete mechanism for securing network coding-based P2P content distribution, which simultaneously possesses the following notable properties: (1)

Corruption detection: each peer can check the validity of encoded blocks on-the-fly efficiently. (2) Attacker identification: all the malicious peers in the network can be identified quickly. (3) High level of security: ESNC is mainly based on ECC and its security is based on elliptic curve discrete logarithm problem (ECDLP) [23]. It can resist hash collision attacks, signature forgery attacks, and collusion attacks with arbitrary number of colluding malicious peers. (4) Low overheads: ESNC incurs low computation and communication overheads. For the proposed network coding signature scheme, since it is based on ECC and supports batch verification, the computation overheads are lower than other comparable state-of-the-art schemes [6, 15–17] for P2P systems. To distribute a file, the signature of the file is only about 0.29 % the total file size. For the malicious peer identification scheme, since some fast verification approaches are used, the computation overheads are trivial, compared with those of the network coding signature scheme. Besides, the signature, serving as evidence, is only 48 B for a block. (5) Topology independent: ESNC does not require any special or fixed topologies, which makes it suitable for P2P systems.

The rest of the paper is organized as follows. Section 2 gives some preliminaries related to the proposed research. Section 3 describes the network coding signature scheme. Section 4 describes the identity-based malicious peer identification scheme. Section 5 analyzes the hash collision and signature forgery attacks, and discusses some security problems of ESNC. Section 6 gives the evaluation results of ESNC in terms of computation overheads, communication overheads and efficiency respectively. Section 7 presents the related work. Section 8 concludes the paper.

2 Preliminaries

In this section, we briefly present practical P2P content distribution with network coding, followed by introduction to orthogonal space and elliptic curve cryptography, which is the foundation of ESNC. At last, we present the network model, the attack model and some assumptions of ESNC.

2.1 Content distribution with network coding

We consider a mesh-structured P2P content distribution network that involves a potentially large number of overlay peers and a source server that distributes contents (e.g. software updates, critical patches, videos, and other large files) to the peers. Assume that a file originally exists on the source. To distribute file, the source first divides the file into multiple generations, each of which consists of m content blocks. For clarity, we focus on a single generation. We denote the original blocks of the generation as b_1', …, b_m'. Each block b_i' is subdivided into r codewords $b_i'=(b_{1i},\ldots,b_{ri})\in F_q^r$, $1\le i\le m$.

To perform encoding operations of network coding, the source then augments each block b_i' with m symbols, with 1 at position i and 0 elsewhere, to form the augmented block b_i. Hence, the generation can be considered as a $(r + m) \times m$ matrix G of elements in the finite field F_q, where prime q is the field size and $m << r$.

$$G=(b_1,b_2,\ldots,b_m)=\begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,m} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r,1} & b_{r,2} & \cdots & b_{r,m} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

With network coding, both the source and the peers perform encoding operations. Whenever a peer or the source needs to forward an encoded block of the generation to another peer, it produces a linear combination of all the blocks of the generation it currently stores. The operation of the system is best described in Fig. 1.

When the source attempts to send an encoded block e_n to peer A, it first picks m random coefficients $(c_{1n}, \ldots, c_{mn})$ from F_q. Then the source creates $e_n = (e_{1n}, \ldots, e_{rn}, c_{1n}, \ldots, c_{mn})$ by linearly combining the blocks b_1, …, b_m with $(c_{1n}, \ldots, c_{mn})$ as

$$e_n=\sum_{i=1}^{m}c_{in}b_i=\left(\sum_{i=1}^{m}c_{in}b_{1,i},\cdots,\sum_{i=1}^{m}c_{in}b_{r,i},c_{1n},\ldots,c_{mn}\right), \quad (1)$$

where $(c_{1n}, \ldots, c_{mn})$ is referred to as the global coefficient vector. In other words, e_n is obtained by multiplying its coefficient vector with matrix G in F_q.

Assume that peer A has received n $(n \le m)$ encoded blocks e_1, …, e_n of the generation, either from the source or from

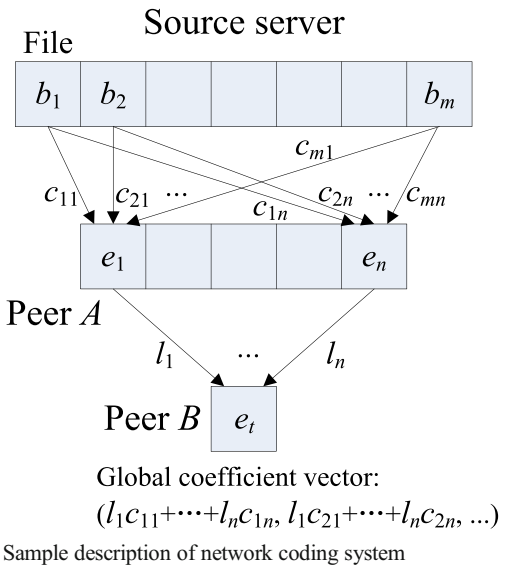


Fig. 1 Sample description of network coding system

other peers. When A needs to send an encoded block $e_t = (e_{1t}, \ldots, e_{rt}, c_{1t}, \ldots, c_{mt})$ to its downstream peer B, it first picks n random local coefficients $(l_1, \ldots, l_n)$ from F_q, then produces e_t as

$$e_t = \sum_{j=1}^{n} l_j e_j. \tag{2}$$

According to Equations (1) and (2), the global coefficient vector $(c_{1t}, \ldots, c_{mt})$ of e_t(s.t., $e_t = \sum_{i=1}^{m} c_{it} b_i$) is computed by $c_{it} = \sum_{j=1}^{n} l_j c_{ij}$, where $(c_{1j}, \ldots, c_{mj})$ is the global coefficient vector of encoded block e_j, $1 \le j \le n$. Therefore,

$$e_t = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} l_j c_{ij} \right) b_i. \tag{3}$$

A peer can reconstruct the original blocks of the generation as soon as receiving m encoded blocks of the generation for which the associated global coefficient vectors are linearly independent to each other. The decoding process is similar to solving a system of linear equations.

Consider the case that encoded block e_t is a corrupted one, then we have

$$e_t \neq \sum_{i=1}^{m} c_{it} b_i. \tag{4}$$

2.2 Orthogonal space

The orthogonal space of matrix G, denoted as V, is the set of all the vectors v for which $vG = 0$. According to the property of the orthogonal space, we have

$$\text{rank}(G) + \dim(V) = m + r,$$

where rank(G) denotes the rank of G, and dim(V) denotes the dimension of V. Since rank(G) = m, dim(V) = r. Therefore, V is spanned by the basis vectors $(v_1, \ldots, v_r)$ which can be found from G using Gaussian elimination.

Thus, each encoded block is orthogonal to any combination of the basis vectors $(v_1, \ldots, v_r)$ of V. That is, for encoded block e_t, we have

$$e_t v^T = 0.$$

2.3 Elliptic curve cryptography

Elliptic curve cryptography (ECC) is a type of public key cryptography (PKC) based on elliptic curve theory [23]. Compared with other PKC systems such as RSA and DSA, ECC uses significantly smaller parameters, but provides equivalent level of security. Some benefits of having smaller key sizes include faster computations, and reductions in processing power, storage space and bandwidth.

The operation of ECC-based schemes involves arithmetic operations on an elliptic curve E over a finite field F_p (denoted as $E(F_p)$) determined by some elliptic curve domain parameters $T = (p, o_1, o_2, P, q)$, which are comprised of a prime power p specifying the field size of F_p, two field elements o_1 and o_2 that specify the equation of the elliptic curve $E(F_p)$, a point $P = (x_P, y_P)$ on $E(F_p)$, and a prime q which is the order of P. Then the cyclic subgroup of $E(F_p)$ generated by P is $<P> = \{O, P, 2P, 3P, \ldots, (q-1)P\}$, where point O denotes the point at infinity.

With the elliptic curve domain parameters, an entity A's private key is defined as a random integer d, $1 \le d \le q-1$, and its public key is defined as a point Q on $E(F_p)$, $Q = dP$. Note that the security of all the ECC-based schemes is based on the apparent intractability of the following elliptic curve discrete logarithm problem (ECDLP):

Given an elliptic curve $E(F_p)$, a point $P \in E(F_p)$ of order q, and a point $Q \in <P>$, determine the integer g, $0 \le g \le q-1$, such that $Q = gP$.

Note that the central operation of ECC-based schemes is point multiplication (e.g. $Q = gP$), which dominates the execution time of most schemes.

2.4 Network model

ESNC can work with arbitrary topologies. Without loss of generality, we consider a mesh-structured P2P content distribution network connecting a source server and a potentially large number of overlay peers, with no regular topology, as shown in Fig. 2.

We assume that a file originally exists on the source server. When the server wishes to publish the file, it encodes the file and distributes encoded blocks to multiple peers simultaneously. Peers download blocks from the server or other peers, and then distribute new encoded blocks which are produced as linear combinations of encoded blocks they hold. In Fig. 2, peer B can upload blocks to peers D, E, F, and download blocks from peers A, C. This subset of peers that B can contact with are called as the neighbors of B and B knows the identities of its neighbors, more specifically, peers A, C

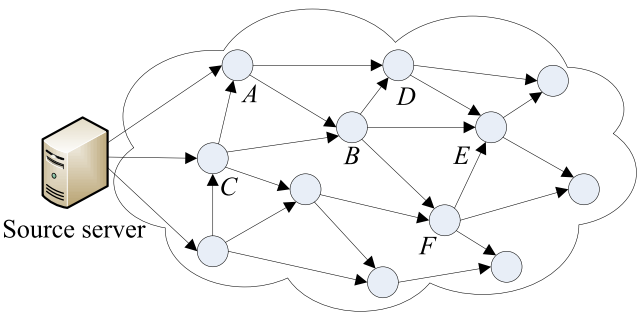


Fig. 2 Network model of mesh-structured P2P network

are the upstream neighbors of B and peers D, E, F are its downstream neighbors.

2.5 Attack model and assumptions

We assume that an arbitrary subset of the total peers is malicious. These malicious peers try to disrupt the distribution of the file by launching network-coding pollution attacks, so that legitimate peers cannot decode the file and the network performance will be reduced significantly. A malicious peer could send any corrupted blocks to any of its downstream neighbors. To disturb security schemes, it can also eavesdrop, modify or drop any messages passing through it, and even try to disparage innocent peers. In addition, multiple malicious peers can collude to launch attacks. P2P networks can suffer from a kind of attack called Sybil attack. In this attack, a malicious user can control multiple malicious peers, each one associated to an identity. This way, if there is not a limitation on the number of identities he can obtain, he could control a big part of peers, which can enhance the power of pollution attacks. In this paper, we focus on the pollution attack in network coding-based P2P content distribution, and design schemes to detect corrupted blocks on-the-fly and precisely identify malicious peers. How to thwart Sybil attacks exceeds the scope of this paper, which has been extensively studied (e.g. [24]).

We assume the source server is trusted, and publicly known so that each peer can contact it directly. To implement ESNC, we make the source server perform some security operations as a central authority like the related schemes [6, 9]. To maintain good system scalability and security, we make the operations performed by the server as few as possible. Although we use such a server, ESNC can be extended to a fully distributed mechanism, for example, the signature of the file can be generated by the peer itself who want to publish the file, instead of the server; when cooperating with reputation systems [25–28] in P2P networks, ESNC can delegate the malicious peer identification to the peers with high reputation. In this paper, we focus on the design and implementation of ESNC. The extension work is out of the scope of this paper and we leave it as our future work.

3 An efficient network coding signature scheme

When using network coding for P2P content distribution, the system is vulnerable to pollution attacks. However, the commonly used methods for protecting the integrity of content blocks by using digital signatures or hashes do not work with network coding, since each peer produces unique encoded blocks which cannot be signed by the server. In this section, we propose an efficient network coding signature scheme to detect corrupted blocks on-the-fly effectively. It can provide high level of security as the homomorphic hashing and homomorphic signatures proposed in [6, 15–17], which are considered as some of the best solutions to prevent pollution attacks on network coding. Moreover, compared to these schemes, our scheme incurs less computation and communication overheads. Thus our scheme is more applicable to P2P systems.

3.1 Basic scheme

The proposed signature scheme is based on ECC. The basic scheme mainly consists of three parts: setup, sign, and verification.

Setup: The source server sets up the ECC domain parameters and the signature keys.

(1) Generate ECC parameters $T = (p, o_1, o_2, P, q)$, which are described in Section 2.3. Then generate a random integer d from the interval $[1, q-1]$ as its private key, and compute its public key as $Q = dP$.
(2) Generate r random points $K = \{K_1, K_2, \ldots, K_r\} \in <P>$ as its signature keys.
(3) Publicize the security parameters $\{T, Q, K\}$ to all peers. In the following, H denotes a cryptographic hash function such that MD5 or SHA-1. For clarity, we focus on a single generation of the file in the following sign and verification algorithms.

Sign: The source computes the signature of the generation as Algorithm 1. The signature S can be denoted as $(\sigma_1, \ldots, \sigma_m, x, z)$. In S, $(\sigma_1, \ldots, \sigma_m)$ is the hash of this generation, which is simply the vector of the hash of each original block in this generation, and it will be used to verify encoded blocks on-the-fly; (x, z) is the signature of $(\sigma_1, \ldots, \sigma_m)$. The signatures of other generations are computed in the same way. Signatures of all the generations constitute the signature of the file and the source distributes them to its downstream peers before propagating encoded content blocks.

Algorithm 1 Signature computation of the generation

INPUT: ECC parameters T, private key d, m original blocks $b_i'(i = 1, \ldots, m)$ of the generation, file identifier id_F, generation identifier id_G, signature keys K.
OUTPUT: Signature of the generation $S = (\sigma_1, \ldots, \sigma_m, x, z)$.

(1) Compute the hash value for each block $b_i' = (b_{1i}, \ldots, b_{ri})$ as a point $\sigma_i = (x_i, y_i) = \sum_{j=1}^{r} b_{ji} K_j$, for $i = 1, \ldots, m$.
(2) Compute $h = H(\sigma_1, \ldots, \sigma_m, id_F, id_G)$.
(3) Select a random integer s from $[1, q-1]$.
(4) Compute $sP = (x', y')$ and $x = x' \bmod q$. If $x = 0$ then go to step 3.

(5) Compute $z = s - d - hx \pmod q$. If $z = 0$ then go to step 3.
(6) Return $S = (\sigma_1, \ldots, \sigma_m, x, z)$.

Verification: When a peer B joins the system, it first downloads the signature of the file from its neighbors. Since signatures of different generations are generated separately, B can download them simultaneously from multiple neighbors quickly. To verify the signature of a generation, B performs Algorithm 2. If B receives invalid signatures from a neighbor, B disconnects from it and re-downloads these signatures from other neighbors. Our evaluation results in Section 6.2 show that the signature of the file is quite small, so it can be propagated quickly to all the peers. After downloading the signature of a generation, B can start downloading encoded blocks of this generation immediately.

Algorithm 2 Signature verification of the generation

INPUT: ECC parameters T, public key Q, signature of the generation $S = (\sigma_1, \ldots, \sigma_m, x, z)$, file identifier id_F, generation identifier id_G.
OUTPUT: Acceptance or rejection of the signature.

(1) Verify that x and z are integers in the interval $[1, q-1]$. If any verification fails, then return ("Reject the signature").
(2) Compute $h = H(\sigma_1, \ldots, \sigma_m, id_F, id_G)$.
(3) Compute $w = z + hx \pmod q$.
(4) Compute $R = wP + Q = (x', y')$ and $u = x' \bmod q$.
(5) If $u = x$ then return ("Accept the signature"); else return ("Reject the signature").

Verification: Then peer B uses the signature of the file, which consists of signatures of all the generations, to verify encoded blocks on-the-fly. The signatures of different generations are used to verify encoded blocks of different generations correspondingly. For an encoded block of a generation, B uses the signature of this generation to verify the block as Algorithm 3.

Algorithm 3 Encoded block verification

INPUT: ECC parameters T, signature of the generation $S = (\sigma_1, \ldots, \sigma_m, x, z)$, signature keys K, an encoded block $e_t = (e_{1t}, \ldots, e_{rt}, c_{1t}, \ldots, c_{mt})$ of the generation.
OUTPUT: Acceptance or rejection of the block.

(1) Compute the hash value of the block as $\theta_t = \sum_{j=1}^{r} e_{jt} K_j$.
(2) Compute the hash value of the block as $\omega_t = \sum_{i=1}^{m} c_{it} \sigma_i$.
(3) If $\theta_t = \omega_t$, e_t is valid, then return ("Accept the block"); else return ("Reject the block").

In this algorithm, the computation of the hash value of an encoded block requires multiple point multiplication operations, which may be expensive for some participating peers. If peers verify every encoded block, the system performance may decrease significantly. To reduce the computation overheads, we employ batch verification approach. More specifically, each peer verifies blocks probabilistically and blocks that have not been verified are kept in an insecure window; every time a peer receives an encoded block, it verifies all the encoded blocks of the same generation in the insecure window using batch verification with some probability. Batch verification is based on Algorithm 3 and will be described in Section 3.3.

Note that batching block verification has the risk of letting some corrupted blocks propagate since blocks are transmitted without being verified. Moreover, with network coding, corrupted blocks are quickly re-combined with other valid blocks at each peer. Therefore, a large portion of peers may be corrupted and the system performance will decrease significantly. To prevent corrupted blocks from infecting large portions of the network, we incorporate into our scheme a cooperative protection approach proposed in [6] where well-behaved peers cooperate to protect themselves against malicious peers by informing affected peers when corrupted blocks are discovered. We will describe the approach in Section 3.4.

3.2 Proof

Proof that signature verification works If a signature $S = (\sigma_1, \ldots, \sigma_m, x, z)$ is indeed generated by the source, then $z \equiv s - d - hx \pmod q$. Rearranging gives

$$s \equiv z + d + hx \equiv w + d \pmod q.$$

Thus $R = wP + Q = (w + d)P = sP$ and so $u = x$ as required. □

Proof that encoded block verification works If the encoded block $e_t = (e_{1t}, \ldots, e_{rt}, c_{1t}, \ldots, c_{mt})$ is valid (not corrupted), according to Equations (1),(3),(4), then

$$\theta_t = \sum_{j=1}^{r} e_{jt} K_j = \sum_{j=1}^{r} \left(\sum_{i=1}^{m} c_{it} b_{ji} \right) K_j$$
$$= \sum_{i=1}^{m} c_{it} \left(\sum_{j=1}^{r} b_{ji} K_j \right) = \sum_{i=1}^{m} c_{it} \sigma_i = \omega_t. \square$$

Note that encoded block verification possesses homomorphic property, as the hash value of a linear combination of some input blocks can be constructed by a combination of the hashes of the input blocks.

3.3 Batch verification

Because of the homomorphic property possessed by block verification, we can verify bocks of the same generation using batch verification, which can verify multiple blocks of the generation synchronously instead of sequentially. The batch verification can accelerate the verification process significantly and thus optimize the system performance.

Assume that we have a batch of n ($n \leq m$) encoded blocks e_j ($j = 1, \ldots, n$) of the same generation to verify all together. The batch verification works as follows.

(1) Build a batched encoded block e_b as $e_b = \sum_{j=1}^{n} l_j e_j$, where the coefficients $(l_1, l_2, \ldots, l_n)$ are randomly chosen from F_q.
(2) Go to step (1) and (2) of Algorithm 3 to compute the hash values of e_b as θ_b and ω_b.
(3) If $\theta_b = \omega_b$, then all the n encoded blocks are valid; otherwise, one or more encoded blocks of them are corrupted.

When peers detect corrupted blocks using batch verification, we further use a binary verification method to efficiently identify every corrupted block. The binary verification method is based on the data structure, namely binary verification tree, which is shown in Fig. 3. We construct it as follows.

(1) The leaf nodes (f, j) ($j = 1, 2, \ldots, n$, $n = 2^{f-1}$) are associated with the n encoded blocks e_j respectively, where f denotes the height of the tree.
(2) Root node (1, 1) is associated to the batched encoded block e_b.
(3) The other node (i, j) ($1 < i < f$) is associated with the batched encoded block $e_{(i,j)}$ for the leaf nodes of a sub-tree rooted at (i, j), where $e_{(i,j)} = \sum_{k=k_1}^{k_2} l_k e_k$, $k_1 = 2^{f-i} \cdot (j - 1) + 1$, $k_2 = 2^{f-i} \cdot j$. The validity of $e_{(i,j)}$ can be verified by checking if $\theta_{(i,j)} = \omega_{(i,j)}$.

The binary verification method is to identify all the corrupted blocks in the binary verification tree, which is a process that recursively verifies the sub-tree dictated by the current verification status of batched encoded blocks. Consider the first step to verify e_b at root node (1, 1). If e_b is valid, all the blocks in the leaf nodes are valid. Otherwise, it further verifies the batched encoded block $e_{(2,1)}$ at the left-child node (2, 1) or $e_{(2,2)}$ at the right-child node (2, 2) in the same way, respectively. This binary verifying process will be iteratively carried out in up-to-bottom way until all corrupted blocks are discovered.

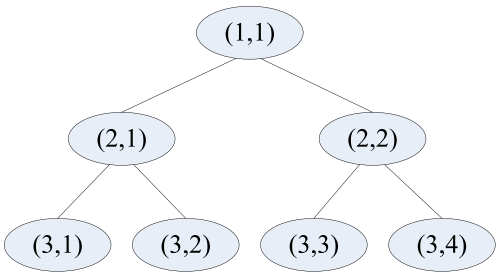


Fig. 3 Sample of the binary verification tree with height $f=3$

3.4 Cooperative protection

As mentioned above, a peer keeps the blocks that have not been verified in an insecure window. Then every peer maintains a table which contains the identities of 1) those peers that downloaded blocks encoded with insecure window blocks, and 2) those peers that delivered blocks inside the insecure window. As shown in Fig. 2, when peer B detects corrupted blocks using batch verification, it sends alert messages to all its neighbors A, C, D, E, F. On receiving the message, A checks if B is in its table. If true, A continues to send alert messages to all its neighbors except B; otherwise, it discards the message. Any other peer that receives the alert message from a neighbor checks if the neighbor is in its table and then acts like A. As discussed in [6], the table can prevent peers that have not been infected from processing the alert message. Alert messages are propagated from one peer to another until all infected peers are informed. A peer starts verifying all the blocks in the insecure window using batch verification right after sending alert messages. To prevent corrupted blocks from propagating, the blocks included in the verifying process will not be used to encode new blocks until they are verified as secure.

3.5 Scheme workflow

We present the complete workflow of the proposed network coding signature scheme through a detailed example, as Fig. 4 shows. In this example, peer A connects with the server, and

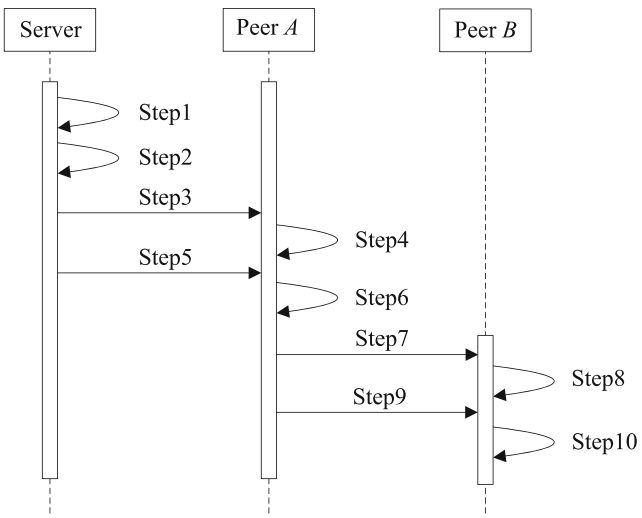


Fig. 4 An example of the work flow of network coding signature scheme

peer B is A's downstream neighbor, as shown in Fig. 2. B joins the system after A has downloaded some encoded blocks. The steps are as follows:

Step 1: Set up the security parameters and publicize them to all peers;
Step 2: Compute the signature of the file using Algorithm 1;
Step 3: Distribute the signature to its downstream peers;
Step 4: Verify the signatures of generations using Algorithm 2;
Step 5: Distribute encoded blocks to its downstream peers;
Step 6: Verify encoded blocks on-the-fly using batch verification with some probability, which is based on Algorithm 3;
Step 7: Send the signatures of generations;
Step 8: Verify the signatures using Algorithm 2 and if any signatures are invalid, disconnect from A and re-download them from other neighbors;
Step 9: Send encoded blocks if B sends a request;
Step 10: Verify encoded blocks on-the-fly using batch verification with some probability, which is based on Algorithm 3; when detect corrupted blocks, send alert messages to all its neighbors.

4 An identity-based malicious peer identification scheme

Previous schemes [6, 14–19, 22] only fight with corrupted blocks, but do not address the network-coding pollution attack at its root, which is the identification of malicious peers. Consequently, malicious peers can keep generating corrupted blocks to continuously degrade the network performance. In this section, we propose an identity-based malicious peer identification scheme which can precisely pinpoint all the malicious peers in the network quickly.

4.1 Basic scheme

First, every peer appends its generated encoded block with a time stamp that records the time when the block generated. Then peers augment the table mentioned in Section 3.4 and the augmented table contains 1) the identities of those peers that downloaded blocks encoded with insecure window blocks, together with the time stamps of the encoded blocks, and 2) the identities of those peers that delivered blocks inside the insecure window, with the time stamps of those insecure window blocks. To make a rapid malicious peer identification process, we introduce a fast verification approach for encoded blocks based on the property of the orthogonal space of vectors, which is described in Section 2.2.

As shown in Fig. 2, when peer B detects corrupted blocks in its insecure window using batch verification, it sends the server an identification message, which contains the identifier of the polluted generation id_G, to trigger the process of identifying malicious peers.

On receiving the message, the server generates a random linear combination v_B of the basis vectors $(v_1, \ldots, v_r)$ of the orthogonal space of the generation, and sends v_B to peer B.

On receiving v_B, B switches to verify the blocks of the polluted generation in its insecure window as below: For block e_t, checking whether it satisfies the following condition,

$$e_t v_B^T = 0, \tag{5}$$

If Equation (5) holds, e_t is valid. The correctness of the verification is straightforward according to Section 2.2. Note that Equation (5) is a simple multiplication, so B can rapidly perform the verification. If peer A sent B some corrupted blocks, B randomly chooses one and sends the server a report, which contains this corrupted block, its time stamp and the identity of A (i.e. ID_A).

On receiving the report from B, the server verifies the reported blocks. If B reported a valid block, the server identifies B as malicious. If B reported a corrupted block e_t with ID_A, then the server knows that A sent e_t to B. However, at this point, the server cannot confirm that A is malicious, since it may be innocent and received corrupted blocks from its upstream neighbors, with which A produced e_t. Therefore, if A is truely innocent, then it will discover at least one corrupted block it has received from its upstream neighbors, and correspondingly it will report the upstream neighbors to the server. Note that if A is malicious and it is also a downstream neighbor of other malicious peers, receiving corrupted blocks from these peers, then A can still report these malicious peers. As a result, A can hide itself and try to escape from being identified in this case.

To confirm whether A is malicious, the server sends the time stamp t_t of e_t and a random linear combination v_A of the vectors $(v_1, \ldots, v_r)$ to A. Then A verifies the blocks of the polluted generation in its insecure window using v_A like B. If peer C sent A some corrupted blocks, to prove the innocence of itself, A should pick out the corrupted blocks of which the time stamps are less than t_t and send the server a report, which contains such blocks, their time stamps and the identity of C (i.e. ID_C). If no such blocks exist, A randomly chooses one corrupted block to report.

On receiving the report from A, the server verifies the reported blocks and acts as follows:

(1) If A reported a valid block, the server identifies A as malicious.
(2) If A reported one or more corrupted blocks, the server continues to contact the peers which sent these corrupted blocks. The process is the same as the contact with A.

Note that if A reported more than one corrupted block from the same peer, the server randomly chooses one to send its time stamp to the peer. Assume that A reported corrupted blocks e_1, e_2 of which the time stamps are less than t_t. Then the server generates a linear combination v of the vectors $(v_1, \ldots, v_r)$, which also satisfies the condition:

$$e_1 v^T = 0 \text{ and } e_2 v^T = 0. \quad (6)$$

The server further checks the following equation:

$$e_t v^T = 0. \quad (7)$$

If Equation (7) holds, A is identified as innocent at this point. Otherwise, A is treated as malicious.

The identification process operates in this way until all the malicious peers are discovered in the identification route. After that, the server broadcasts the identification result by sending to peers a message which contains the identities of malicious peers. The peers then verify the blocks in their insecure windows in batch and put these malicious peers into their blacklists, stop any transmission with them.

As shown in Fig. 2, when peer B detects corrupted blocks using batch verification, it sends an identification message to the server and alert messages to all its neighbors. Note that after the server accepts the message from B, it will ignore messages from other peers until this identification process is complete. Since during this identification process, some malicious peers which have received alert messages from their neighbors may deliberately send identification messages to the server, if the server accepts all these messages, a lot of system resources, especially server resources, will be wasted on the identification processes which will not discover any new malicious peers in most cases. Although such design may cause missing malicious peers occasionally in the early period, these peers will still be discovered quickly, as shown in Section 6.3. So it is necessary to make such a trade-off.

4.2 Proof

As Section 2.2, we assume that an encoded block $e_t = l_1 e_1 + l_2 e_2 + \sum_{j=3}^{n} l_j e_j$, where e_t, e_1, e_2 are corrupted, $e_3, \ldots, e_n$ are valid, and $l_1, \ldots, l_n$ are n random local coefficients in F_q. The vector v is a linear combination of the vectors $(v_1, \ldots, v_r)$, which are basis vectors of the orthogonal space of the generation.

According to Section 2.2, v is orthogonal to all the valid encoded blocks $e_3, \ldots, e_n$. Thus,

$$\left(\sum_{j=3}^{n} l_j e_j\right) v^T = 0. \quad (8)$$

From Equations (6) and (8), if we have $e_1 v^T = 0$ and $e_2 v^T = 0$, then $(l_1 e_1 + l_2 e_2 + \sum_{j=3}^{n} l_j e_j) v^T = 0$. As a result, if A is innocent and received corrupted blocks e_1, e_2 from its upstream neighbors, with which A produced the corrupted block e_t, then Equation (7) holds. Otherwise, A is malicious and $e_t \neq l_1 e_1 + l_2 e_2 + \sum_{j=3}^{n} l_j e_j$, then $e_t v^T \neq 0$. □

4.3 Block signature

The feasibility of the above identification scheme relies on the requirement that any peer in the identification route must report the true checking result to the server. And the malicious peer cannot deny its behavior. In order to achieve these requirements, we present a lightweight block signature approach which holds every peer accountable for the blocks it sends out. Every peer signs the blocks before sending them out, and the signature associated with a block can be used as evidence by the reporting peer to demonstrate that the reported peer has really sent this block. Given ECC parameters $T = (p, o_1, o_2, P, q)$, a cryptographic hash function H, which are described in Section 2.3, for peer A, we suppose its private key is d_A and its public key is $Q_A = d_A P$. Then it signs an encoded block as follows:

Algorithm 4 Signature computation of the block

INPUT: ECC parameters T, private key d_A, an encoded block e_t, time stamp t_t of the block, generation identifier id_G.
OUTPUT: Signature of the block $\varphi_t = (x_A, z_A)$.

(1) Compute $h_A = H(e_t, t_t, id_G)$.
(2) Select a random integer s_A from $[1, q-1]$.
(3) Compute $s_A P = (x_A', y_A')$ and $x_A = x_A' \bmod q$. If $x_A = 0$ then go to step 2.
(4) Compute $z_A = s_A - d_A - h_A x_A \pmod q$. If $z_A = 0$ then go to step 2.
(5) Return $\varphi_t = (x_A, z_A)$.

When a peer or the server receives an encoded block produced by A, it verifies the signature appended to the block as follows:

Algorithm 5 Signature verification of the block

INPUT: ECC parameters T, public key Q_A, an encoded block e_t, time stamp t_t of the block, signature of the block $\varphi_t = (x_A, z_A)$, generation identifier id_G.
OUTPUT: Acceptance or rejection of the block.

(1) Verify that x_A and z_A are integers in the interval $[1, q-1]$. If any verification fails, then return ("Reject the block").
(2) Compute $h_A = H(e_t, t_t, id_G)$.

(3) Compute $w_A = z_A + h_A x_A \pmod q$.
(4) Compute $R_A = w_A P + Q_A = (x_A', y_A')$ and $u_A = x_A' \bmod q$.
(5) If $u_A = x_A$, φ_t is valid to e_t, t_t, id_G, then return ("Accept the block"); else return ("Reject the block").

Signature computation and verification of the block are similar to those of the generation in Algorithm 1 and Algorithm 2. Note that both the signature computation and verification of the block only need one point multiplication operation, thus the algorithms are lightweight and the signature can be processed quickly.

When a peer receives an encoded block, it verifies the signature appended to the block immediately, and then verifies encoded blocks using batch verification with some probability, as described in Section 3.1.

4.4 Scheme workflow

We present the complete workflow of the proposed malicious peer identification scheme through a detailed example, as Fig. 5 shows. In this example, peer B is A's downstream neighbor and has received corrupted blocks from A, as shown in Fig. 2. All notations used in Fig. 5 have occurred in Sections 4.1 and 4.3. They have the same meaning correspondingly, so the explanation of the notations is omitted for simplicity in the figure. The steps are as follows:

Step 1: Detect corrupted blocks;
Step 2: Send an identification message, containing id_G;
Step 3: Generate v_B;
Step 4: Send v_B;
Step 5: Verify blocks using Equation (5) and discover e_t;
Step 6: Send a report, containing $\{e_t, \varphi_t, t_t, ID_A\}$;
Step 7: Verify e_t and if e_t is corrupted, verify φ_t. Identify B as malicious if e_t is not corrupted or φ_t is not valid;
Step 8: Send $\{t_t, v_A\}$ if e_t is corrupted and φ_t is valid;
Step 9: Verify blocks using v_A and discover e_1, e_2;
Step 10: Send a report, containing the information of e_1, e_2;
Step 11: Verify e_1, φ_1 and e_2, φ_2 to identify A; use Equations (6), (7) to further identify A;
Step 12: Identification process operates in this way until all the malicious peers are discovered in the route;
Step 13: Broadcast the identification result;
Step 14: Put identified malicious peers into blacklist.

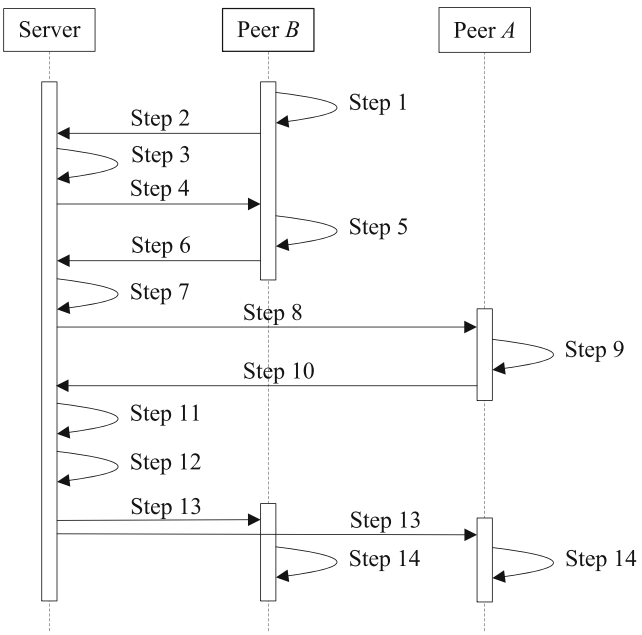


Fig. 5 An example of the work flow of malicious peer identification scheme

5 Security analysis and discussion

In this section, we will analyze the hash collision and signature forgery respectively. Then we will discuss some security problems of ESNC. We preserve the notations of previous sections here.

5.1 Hash collision

To thwart the network coding signature scheme, a malicious peer can produce a hash-collision block which can pass the verification. However, producing such a block is as hard as computing elliptic curve discrete logarithm problem (ECDLP).

Proposition 1 *Given r distinct points $K_1, K_2, \ldots, K_r$ on an elliptic curve $E(F_p)$ contained in a cyclic subgroup of prime order q, an encoded block $e_t=(e_{1t},\ldots,e_{rt},c_{1t},\ldots,c_{mt})\in F_q^{r+m}$ and its hash value $\theta_t=\sum_{j=1}^{r} e_{jt}K_j$, generating a hash-collision block $e_t'=(e_{1t}',\ldots,e_{rt}',c_{1t},\ldots,c_{mt})\in F_q^{r+m}$ from e_t is equivalent to solve a hard ECDLP problem, where $e_t \neq e_t'$ and $\theta_t = \theta_t'$.*

Proof First we treat the case when $r=2$. Let K_1 and K_2 be two distinct points of order q on $E(F_p)$. Given a valid block $e_t = (e_{1t}, e_{2t}, c_{1t}, \ldots, c_{mt})$ with its hash value $\theta_t = e_{1t}K_1 + e_{2t}K_2$, a malicious peer attempts to generate a hash-collision block $e_t' = (e_{1t}', e_{2t}', c_{1t}, \ldots, c_{mt})$ $(e_t \neq e_t')$ such that $e_{1t}K_1 + e_{2t}K_2 = e_{1t}'K_1 + e_{2t}'K_2$. This means $(e_{1t}-e_{1t}')K_1 + (e_{2t}-e_{2t}')K_2 = O$. Suppose that $e_{1t} = e_{1t}'$, then $(e_{2t}-e_{2t}')K_2 = O$. Since K_2 is a point of order q, we have $(e_{2t}-e_{2t}') \equiv 0 \bmod q$, that is, $e_{2t} = e_{2t}'$ in F_q. This contradicts the assumption that $e_t \neq e_t'$ in F_q^2. Thus we have that $K_2 = -(e_{1t} - e_{1t}')(e_{2t} - e_{2t}')^{-1} K_1$, where the inverse is taken modulo q. This is a hard ECDLP problem. So we cannot determine e_t'.

Next, consider the case that $r>2$. Given a valid block $e_t = (e_{1t}, \ldots, e_{rt}, c_{1t}, \ldots, c_{mt})$ with its hash value $\theta_t=\sum_{j=1}^{r} e_{jt}K_j$, a

malicious peer attempts to generate a hash-collision block $e_t' = (e_{1t}', \ldots, e_{rt}', c_{1t}, \ldots, c_{mt})$ $(e_t \neq e_t')$ such that $\sum_{j=1}^{r} e_{jt}K_j = \sum_{j=1}^{r} e_{jt}'K_j$. This means $\sum_{j=1}^{r}(e_{jt}-e_{jt}')K_j = O$. Similar to the case when $r=2$, it is easy to prove that in order to satisfy $\sum_{j=1}^{r}(e_{jt}-e_{jt}')K_j = O$, there exist at least two distinct elements in $(e_{1t}, \ldots, e_{rt})$ and $(e_{1t}', \ldots, e_{rt}')$. Without loss of generality, we assume that $e_{it} \neq e_{it}'$ and $e_{kt} \neq e_{kt}'$. Consider that $K_j = g_jP$ $(1 \leq j \leq r)$, where secret g_j are chosen at random from F_q, we have $(e_{it}-e_{it}')K_i + \sum_{j=1, j\neq i}^{r}(e_{jt}-e_{jt}')g_k^{-1}g_jK_k = O$. Thus $K_i = -(e_{it}-e_{it}')^{-1}\sum_{j=1, j\neq i}^{r}(e_{jt}-e_{jt}')g_k^{-1}g_jK_k$, where the inverse is taken modulo q. This is also a hard ECDLP problem. So we cannot determine e_t'.□

5.2 Signature forgery

A malicious peer may attempt to forge signatures to make peers receive invalid signatures of generations, or disparage innocent peers with corrupted blocks not sent by these peers. A malicious peer M is able to forge the signature of the source or other peers as follows. We assume that Q is the public key of the peer whose signature is forged by M.

First, M selects an arbitrary integer s_M and compute $s_MP = (x_M', y_M')$. According to Section 3.2, $R_M = s_MP$. Then M sets $x_M = x_M' \bmod q$. Next, M must determine z_M to generate the signature which contains (x_M, z_M). According to Algorithm 2 and Algorithm 5, we know if M can find z_M such that $(z_M + h_Mx_M)P = R_M - Q$, then it can forge the signature successfully. However, determining z_M from the equation is obviously a hard ECDLP problem. So the malicious peer cannot launch signature forging attacks.

Note that in Algorithm 1 and Algorithm 4, the random integer s selected from $[1, q-1]$ has the same security requirement as the private key. If a malicious peer M learns s_A that peer A used to generate a signature on a block, then M can recover A's private key since $d_A = s_A - z_A - h_Ax_A \pmod q$. So the random integer s must be kept secret. In addition, s should be generated randomly. This ensures that s never repeats, which is important because otherwise the malicious peer can forge signatures once getting a valid signature. To see this, suppose that peer A uses the same s_A to generate signatures and (x_A, z_A) is one signature. Then the malicious peer M can forge a signature of A as (x_M, z_M), where $x_M = x_A$ and $z_M = z_A + h_Ax_A - h_Mx_M \pmod q$. According to Algorithms 2, 5 and Section 3.2, $R_M = w_MP + Q_A = (z_M + h_Mx_M)P + Q_A = (z_A + h_Ax_A + d_A)P = s_AP$. Thus, the malicious peer can forge the signature of A successfully.

5.3 Discussions

Security level The implementation of ESNC is mainly based on ECC and its security is based on ECDLP. The best algorithm known for solving the underlying hard mathematical problems in ECC (ECDLP) takes fully exponential time. On the other hand, the best algorithms known for solving the underlying hard mathematical problems in RSA and DSA (the integer factorization problem, and the discrete logarithm problem respectively) take sub-exponential time. This means that significantly smaller parameters can be used in ECC than in other systems such as RSA and DSA, but with equivalent level of security. A typical example of the size in bits of the keys used in different systems, with a comparable level of security, is that a 160-bit ECC key is equivalent to RSA and DSA with a modulus of 1024 bits. Thus, ESNC can provide high level of security with small ECC parameters while possessing performance advantages (e.g. $|p| = |q| = 192$ bits).

To accelerate the malicious peer identification process, we introduce a fast verification approach based on the property of the orthogonal space of vectors. The probability that a corrupted block of a generation can be verified as valid using the vector in the orthogonal space of the generation is only $1/q$ (e.g. 2^{-192} when $|q| = 192$ bits). Note that the fast verification approach cannot replace the proposed network coding signature scheme to verify encoded blocks on-the-fly for the reason that the vectors in the orthogonal space of generations cannot be disseminated to peers before the propagation of blocks, or else malicious peers can easily launch attacks which can produce corrupted blocks that can pass the verification, or launch other kinds of attacks, such as collusion attacks and DoS attacks.

DoS attacks Malicious peers may send a lot of blocks to the server in the identification process to waste the resources of the server. To thwart this kind of DoS attacks, in our scheme, the server verifies the block and its signature using fast verification approaches presented in Section 4.1 and 4.3. Moreover, once a block or its signature fails to pass the verification, the peer who sent them to the server is identified as malicious immediately.

Collusion attacks ESNC can deal with arbitrary number of colluding malicious peers. This is because no matter how many or which peers are colluding, they still cannot learn the private key of the source or legitimate peers; thus, they cannot make a legitimate peer accept invalid signatures of generations, or disparage innocent peers with corrupted blocks not sent by them.

We consider the situation where multiple malicious peers collude to hide themselves. Consider a transmission path $C \rightarrow A \rightarrow B$, where A, C are two malicious peers, and B is infected by A or C. When B detects corrupted blocks and reports A to the server, A may not report C even if A has verified that it has received corrupted blocks from C. However, C will still be identified if it sends corrupted blocks to any legitimate downstream peers. If C only sends corrupted blocks to A, then A will be identified as malicious and the pollution flow from C

to A will be stopped at A, without infecting any legitimate peers.

Peer churn Our scheme can effectively identify malicious peers even if they leave right after sending corrupted blocks, because the malicious peer identification process does not require the involvement of malicious peers. With a blacklist maintained by the server and the peers, our system can prevent malicious peers from rejoining the system to launch pollution attacks again. Besides, if a legitimate peer A who is a downstream neighbor of a malicious peer C leaves, then A can not report C to the server. However, as long as any other downstream neighbor of C is alive, C will be identified.

6 Performance evaluation

In this section, we evaluate ESNC in terms of computation overheads, communication overheads and efficiency respectively.

6.1 Computation overheads

We start by evaluating the computation overheads of the proposed network coding signature scheme, comparing it with those of other four comparable schemes with equivalent level of security for encoded block verification [6, 15–17]. Table 1 shows the main computation overheads of the five schemes in terms of the initializing cost, signing cost of an encoded block, combining cost of signatures and verifying cost of an encoded block respectively. We define the computation costs of the primarily cryptographic operations of the five schemes as follows. Let mul. denote the time cost to perform a point multiplication over an elliptic curve, exp. the time cost to perform one modular exponent operation, and par. the time cost of a pairing operation. Note that the paring operation is more time-consuming than the other two operations, and the point multiplication has a lower cost than the modular exponential operation, when the schemes are at the same security level. We neglect all the trivial operations for the sake of clarity.

Yu et al.'s scheme, Charles et al.'s scheme and Boneh et al.'s scheme are homomorphic signature schemes for network coding. In these schemes, every time the source sends an encoded block, it needs to generate a signature appended to this block. The computation overheads of generating such a signature are described in Table 1 as signing cost. Furthermore, each peer can generate a signature for its produced encoded block as a random linear combination of the signatures of incoming blocks. The computation overheads of generating the signature are described in Table 1 as combining cost. From the table, we can see that these homomorphic signature schemes incur substantial signing overheads when encoded blocks are propagating in the network. Unlike these homomorphic signature schemes for network coding, Gkantsidis et al.'s scheme and our scheme don't incur any signing cost and combining cost for the reason that there is no such signature appended to encoded blocks. In these two schemes, the signature of the file is produced and disseminated to peers before the propagation of encoded blocks. The overheads of producing the signature of one generation are described in the table as initializing cost. Note that blocks of a generation can start propagating as soon as the signature of this generation is disseminated to peers. Compared with Gkantsidis et al.'s scheme, our scheme incurs less computation overheads for the point multiplication over an elliptic curve has a lower cost than modular exponential operations.

Block verification is critical for the system performance. To compare the verifying costs of these schemes with equivalent level of security, we give the benchmarks of the cryptographic operations on Intel Core™ 2 Duo 2.0 GHz Linux machine: mul. = 0.68 ms; par. = 2.50 ms, and exp. = 0.75 ms. The choice of elliptic curve can influence the computation costs of point multiplication and pairing operations. Existing studies [29, 30] have shown that these operations, especially the time-consuming paring operation, can be performed fast on a family of super-singular Koblitz elliptic curves of characteristic 3. Thus, we implement a super-singular Koblitz curve $y^2 = x^3 - x \pm 1$ over $F_{3^{97}}$. The point multiplication and pairing operation are based on this curve. According to Table 1, we plot the verifying costs of an encoded block with different numbers of codewords in these schemes, as shown in Fig. 6. The verifying cost of Charles et al.'s scheme is always the largest. Boneh et al.'s scheme has slightly larger verifying cost than Gkantsidis et al.'s scheme. Yu et al.'s scheme has the same approximate verifying cost as Gkantsidis et al.'s scheme.

Table 1 Comparisons of computation overheads

	Gkantsidis et al. [6]	Yu et al. [15]	Charles et al. [16]	Boneh et al. [17]	ESNC
Initializing cost	mr exp.	N/A	$(m + r)$ mul.	N/A	$(mr + 1)$ mul.
Signing cost	N/A	$(m + r + 1)$ exp.	$(m + r)$ mul.	$(m + r + 1)$ exp.	N/A
Combining cost	N/A	n exp.	n mul.	n exp.	N/A
Verifying cost	$(m + r)$ exp.	$(m + r + 1)$ exp.	$(m + r + 1)$ par. + $(m + r)$ mul.	$(m + r)$ exp. + 2 par.	$(m + r)$ mul.

We preserve the notations of previous sections in the table

It is evident that the basic verifying cost of our scheme (not applying batch verification) is smaller than the other three schemes. Moreover, when applying batch verification in our scheme, the verifying cost can be significantly reduced further. From Section 3.3, we know the averaged verifying cost per block is inverse proportional to the number of blocks in batch verification, which is $(m + r)/n$ mul. In Fig. 6, we show the averaged cost per block for verifying only two blocks in batch verification of our scheme. Hence, the proposed network coding signature scheme can efficiently verify blocks on-the-fly with low computation overheads. Figure 6 shows the comparison of the verifying costs of an encoded block in five schemes. As the number of blocks increases, the total verifying costs will incur much more computation overheads in the schemes with larger verifying cost per block. Besides, in our scheme, for the signature of a generation, only one point multiplication is required to verify the signature, so we omit it in Table 1.

To make malicious peer identification process rapid, we introduce a fast verification approach based on the property of the orthogonal space of vectors, and present a lightweight block signature approach where the signature appended to a block serves as evidence to demonstrate that the block is really sent by some peer. For the latter, both the signing cost and verifying cost of a block are only one point multiplication operation. Comparing to the network coding signature schemes analysed above, the computation overheads of the approach is trivial. For the former, the processing speed of the approach is extremely fast. For example, it takes only about 1 μs to verify a block.

6.2 Communication overheads

Homomorphic signature schemes [15–17] require appending a signature to every transmitted block, which not only incur large computation overheads, but also incur significant communication overheads to the system. For example, Yu et al.'s scheme can be considered as a RSA-based signature, and the size of a signature is 128 B per block with RSA modulus of 1024 bits.

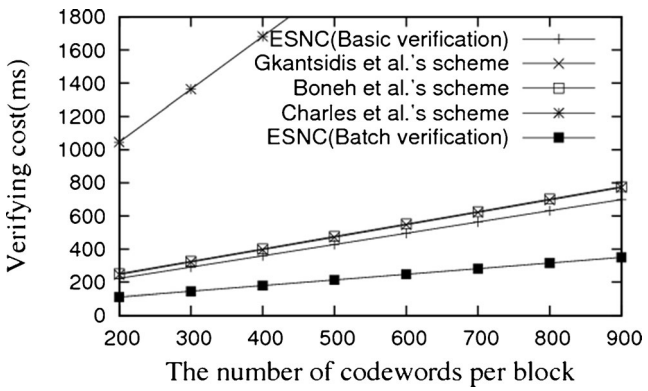


Fig. 6 Verifying cost as a function of the number of codewords per encoded block (the generation size is set to 128 blocks)

For the proposed network coding signature scheme and Gkantsidis et al.'s scheme, peers only need to download signature of the file once and no signature is appended to transmitted blocks. In our scheme, the signature mainly consists of points on elliptic curve $E(F_p)$ and the size of a point is $2\log_2 p$ bits, or 48 B when $|p| = 192$ bits, according to ECC. The point is only 0.0029 times the size of a block with size 16 KB. Thus, the signature of the file is only about 0.29 % the total file size. In Gkantsidis et al.'s scheme, the signature of the file is much larger. The signature consists of the same number of elements while the size of an element is 128 B. Thus, their signature is 2.67 times the size of the signature in our scheme.

In our scheme, some extra communication overheads are needed for the proposed lightweight block signature approach. However, the overheads are small. The signature associated with a block is only 48 B when $|p| = 192$ bits.

6.3 Efficiency

In this section, we present simulations to evaluate the efficiency of ESNC. For the simulation methodology, we refer to the related work [1, 6], in which the simulation model can capture many important properties of network coding-based P2P content propagation, and we conduct simulations using PeerSim [31], which is a widely used P2P simulation framework under various settings and scenarios. Our purpose is not to construct a perfectly realistic simulation, but to demonstrate the efficiency of ESNC in some specific scenarios, which, we believe, are of practical importance. In the simulations, we use the percentage or the number of corrupted peers as a metric to measure the efficiency of security schemes (the efficiency is inversely proportional to the percentage or the number of corrupted peers).

We start by generating the overlay topology of the network. We construct a random mesh-structured topology with one source server, where peers randomly select their neighbors. The topology is a directed random graph. The out-degree for each peer is set to 5; but other values give similar results. The network size is fixed to 500–2000 peers and will be specified in different simulation scenarios. We randomly choose a fraction of all peers to be malicious, and each of them injects corrupted blocks to all its downstream neighbors. The percentage of malicious peers varies between 5 %–20 % and the percentage of corrupted blocks injected by malicious peers varies between 10 %–100 % in simulations. The rest of the peers cooperate to distribute a file that has a large number of blocks using network coding. Note that when network coding is used, encoded blocks of a generation are equivalent and useful to any peer, thus block select policy such as rarest first policy in traditional P2P content distribution network is not needed; moreover, file blocks satisfy uniform distribution in

the network and optimal network transmission can be achieved. Peers rely on local information and then download the encoded blocks of required generations. The generation size of the file is set to 128 blocks. The simulation is round-based, where in each round a peer can upload encoded blocks to its neighbors at random. We assume that the upload capacity of each peer is 5 blocks per round. Every time a legitimate peer receives a block, it verifies the insecure window blocks using batch verification with some probability and, if one or more of them are corrupted, then the peer alerts other infected peers and triggers a malicious peer identification process. We will also vary the probability of verifying insecure window blocks from 1 %–10 % in different simulation scenarios. All the results are averaged over 50 runs.

Impact of the probability of verifying blocks In Fig. 7 we show how the corruption varies in a network of 1,000 peers where 10 % of them are malicious and each malicious peer injects corrupted blocks at a rate of 50 %, with different probabilities of verifying blocks for each legitimate peer. When we increase the verifying probability from 1 % to 5 %, the percentage of corrupted peers decreases significantly, and while when we further increase the verifying probability from 5 % to 10 %, there is a small decrease in the percentage of corrupted peers, hence, larger verifying probability does not justify the extra computational effort in this case. Note that even when the verifying probability is 1 %, the mean percentage of corrupted peers per round in the corruption period is still small, about 8.2 %. In all the three cases, the corruption can be cleared quickly, which means all the malicious peers can be identified by our scheme quickly. Thus, ESNC can achieve high efficiency with small verifying probabilities.

Impact of the rate of injecting corrupted blocks In Fig. 8 we show how the corruption varies in a network of 1,000 peers where 10 % of them are malicious and each legitimate peer verifies blocks with 5 % probability, with different rates of injecting corrupted blocks for each malicious peer. Obviously, when we increase the attack rate from 10 % to 100 %, the percentage of corrupted peers also increases. To make more

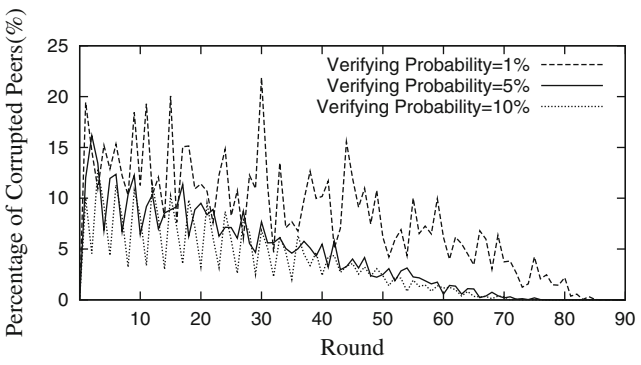


Fig. 7 Corruption variation with different verifying probabilities

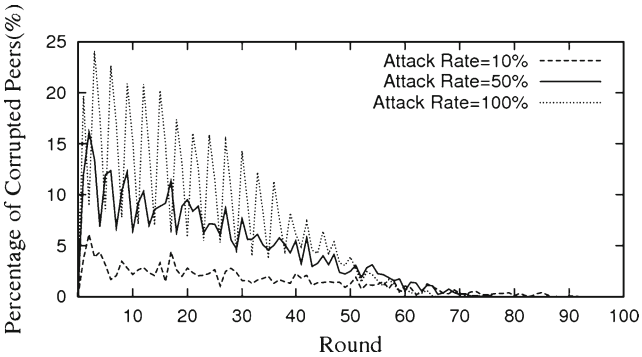


Fig. 8 Corruption variation with different attack rates

corruption, malicious peers may inject corrupted blocks at a high rate. Note that even when the attack rate is 100 %, the mean percentage of corrupted peers per round in the corruption period is still small, about 8.1 %; moreover, in this case, the corruption is cleared the most quickly and it takes only 66 rounds to identify all the malicious peers. To make the identification time longer, malicious peers may inject corrupted blocks at a low rate. However, even when the attack rate is 10 %, it just takes 92 rounds to identify all the malicious peers and the mean percentage of corrupted peers per round is quite small, about 1.5 %. Thus, ESNC can achieve high efficiency regardless of the attack rate.

Comparison with Gkantsidis et al.'s scheme We implement the cooperative security scheme proposed by Gkantsidis et al. [6]. In their scheme, homomorphic hashing is used, where peers cooperate to protect themselves by sending alert messages. Gkantsidis et al. show by simulations that their scheme guarantees higher efficiency than other probabilistic schemes that use homomorphic hashing. Figure 9 shows the corruption variation comparison between ESNC and their scheme. We consider a network of 1,000 peers where 10 % of them are malicious and each malicious peer injects corrupted blocks at a rate of 50 %. Each legitimate peer verifies blocks with 5 % probability. In the case of cooperative security, since their scheme can not identify malicious peers, the corruption is

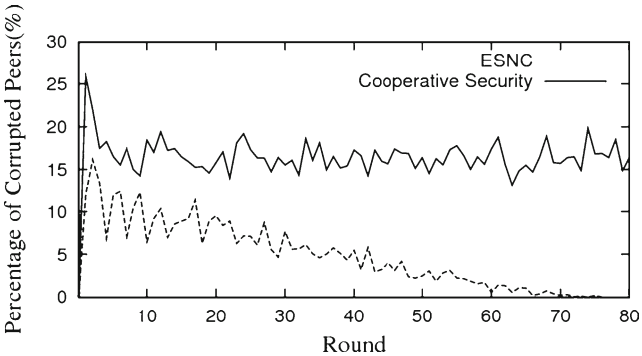


Fig. 9 Corruption variation comparison between ESNC and cooperative security

continual in the entire process of content distribution, hence, malicious peers can continuously degrade the network performance by injecting corrupted blocks. Compared with their scheme, the percentage of corrupted peers in ESNC is much smaller, thus, ESNC can achieve much higher efficiency.

Impact of the percentage of malicious peers In Fig. 10 we show how the corruption varies in a network of 1,000 peers where each malicious peer injects corrupted blocks at a rate of 50 % and each legitimate peer verifies blocks with 5 % probability, with different percentages of malicious peers. Obviously, as we increase the percentage of malicious peers from 5 % to 20 %, the percentage of corrupted peers also increases. Note that even when the percentage of malicious peers is up to 20 %, the mean percentage of corrupted peers per round is about only 9.7 %, until the corruption is cleared. The rounds required for identifying all the malicious peers also increase as malicious peers are increased. To evaluate the time for identifying all the malicious peers, we conduct simulation study as described in the next paragraph.

Evaluation of the time for identification In Fig. 11 we show the number of valid blocks that have been downloaded by peers at the time when all the malicious peers are identified. We consider a network of 1,000 peers where 10 % of them are malicious and each malicious peer injects corrupted blocks at a rate of 50 %. Each legitimate peer verifies blocks with 5 % probability. In the figure, the number of the downloaded blocks is sorted in an ascending order. Since the generation size of the file is 128 blocks, most peers have only downloaded 2 generations at this time. Generally, a file consists of many generations. For example, if the size of a file is 256 MB and the size of a block is 16 KB, then the file consists of 16384 blocks and 128 generations. Thus, for distributing a file, all the malicious peers can be identified and the corruption can be cleared quickly when most peers have just completed downloading 2 generations. In our simulations, when the block size and the generation size are smaller (e.g. in P2P systems distributing media streaming content), the identification process will be faster.

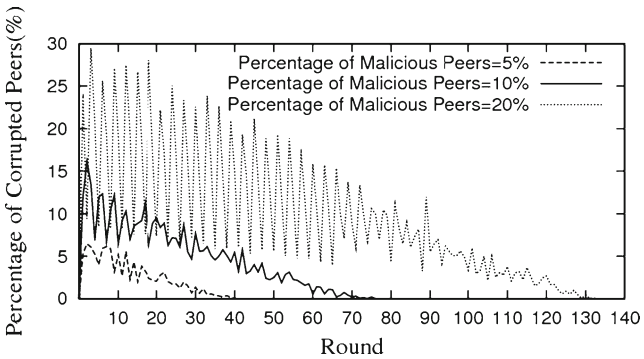


Fig. 10 Corruption variation with different percentages of malicious peers

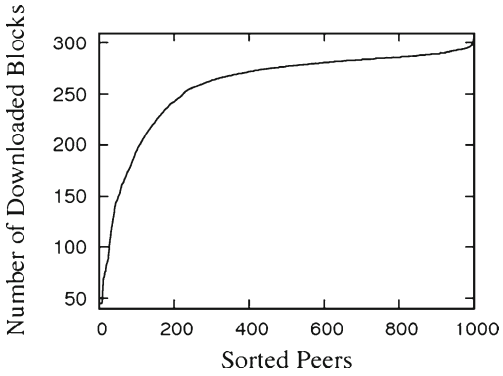


Fig. 11 Number of blocks downloaded by peers at the time when all the malicious peers are identified

Impact of the network size In Fig. 12 we show how the corruption varies in the network with different network sizes. The number of malicious peers is fixed to 100 and each malicious peer injects corrupted blocks at a rate of 50 %. Each legitimate peer verifies blocks with 5 % probability. Figure 12 shows that the tails of the three curves are close, which means that with different network sizes, the time required for identifying all the malicious peers is close and, hence, the identification time mainly depends on the number of malicious peers, rather than network size. When we decrease the network size from 2000 to 500, there is a small decrease in the number of corrupted peers. The mean number of corrupted peers per round is 55.4, 46.3 and 34.3 in the network of size 2000, 1000 and 500 respectively. Because as the network size is decreased, the overlap in the corrupted regions will increase, and the total number of corrupted peers will decrease. This is an important result. This means that, if we fix the network size and increase the number of malicious peers, the overlap in the corrupted regions will also increase and while the total number of corrupted peers increases, the efficiency of malicious peers decreases.

7 Related work

Besides the relevant work conducted by Krohn et al. [13] and Gkantsidis et al. [6], some schemes utilizing homomorphic

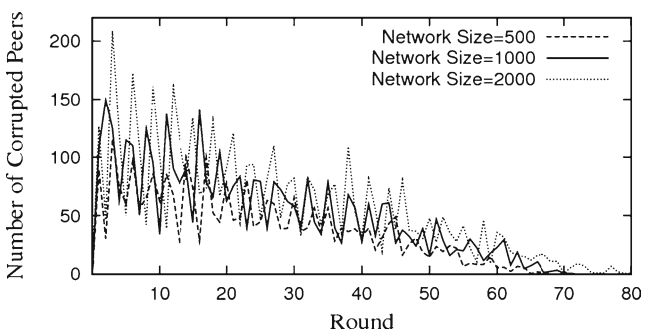


Fig. 12 Corruption variation with different network sizes

signatures are also proposed in [15–17]. These schemes, however, incur more computation or communication overheads, which result in high delays when encoded blocks are propagating in the network. Thus, they are too expensive for P2P systems.

Zhao et al. [32] implemented a middleware that uses NVIDIA GPUs to accelerate homomorphic hashing, and they have achieved much higher hashing throughput. The main issue of this scheme is that the middleware is only applicable for NVIDIA GPUs. Because there are a lot of different GPU vendors, this scheme is not suitable for many realistic P2P systems which may contain peers that have various GPUs, or don't have any GPU.

Agrawal and Boneh [18] proposed a homomorphic MAC scheme for linear network coding. The scheme allows peers to efficiently verify encoded blocks on-the-fly using MAC tags. However, it provides much lower level of security for encoded block verification than the schemes based on homomorphic hashing and homomorphic signatures. Moreover, it relies on the assumption that the source and all the legitimate peers share secret keys, which is not the case in P2P systems. In other words, the scheme will not be able to defeat network-coding pollution attacks in P2P systems. The authors in [18] further extended their scheme to pre-distribute different keys to system peers based on the cover free family concept so that peers in P2P systems could verify blocks. However, this scheme is only c-collusion resistant for some pre-determined c. It is also susceptible to tag-pollution attacks, where malicious peers tamper with subsets of tags of the blocks they receive, as presented in [19]. For P2P systems, collusion attacks are common. Thus, the scheme is not suitable for P2P systems.

Li et al. [19] proposed RIPPLE, also a homomorphic MAC scheme utilizing time asymmetry (inspired by TESLA [33]) to achieve collusion and tag-pollution resistance. This scheme, however, only works on fixed directed acyclic graph networks: it needs to know the longest path from each peer to the source in order to pre-compute the MAC tags for the peers. Thus, the scheme can not be used in P2P systems with general random network topologies.

Zhang et al. [22] introduced both a homomorphic MAC scheme and a homomorphic signature scheme, based on which they further proposed a hybrid scheme. This hybrid scheme has lower computation overheads than the schemes based on homomorphic signatures; however, the overheads are still significantly higher than those of the other two schemes based on homomorphic MAC. This scheme is not susceptible to tag-pollution attacks but only c-collusion resistant.

In [14], Kehdi and Li proposed a light-weight scheme based on the null-space property of network coding. There are still some drawbacks of this scheme. First, the scheme provides low level of security, and is vulnerable to collusion attacks, where multiple malicious peers can collude to infer the null keys employed for the verification of network coding and let legitimate peers accept corrupted blocks. Next, the null keys distribution phase may be attacked; to protect the null keys, homomorphic hashing must be used, which will cause extra overheads.

Some schemes [20, 21] aim at correcting errors at decoders. However, these schemes are applicable only when less than a threshold number of corrupted blocks injected into the network or specified number of network links can be eavesdropped and corrupted.

All the above schemes focus on the corruption detection or error correction for network coding. Research work on identifying attackers is much lesser. However, attacker identification is a more efficient scheme to thwart network-coding pollution attacks.

Jafarisiavoshani et al. [34] leverages the subspace properties of randomized network coding to locate pollution attackers. In a general network topology having a single attacker, the scheme can only locate the attacker with an uncertainty of two peers; when there are multiple attackers, the uncertainty is within a set of peers including the attackers and their parents and children. Thus, the scheme can not locate attackers precisely.

Wang et al. [9] proposed a malicious peer identification scheme which achieves high computational efficiency. However, the identification requires the server to distribute multiple checksums of all the blocks in the polluted generation to all the peers every time pollution attacks are detected, which incurs significant communication overheads. In their scheme, pollution attacks are detected when some peer fails in the decoding process. In other words, pollution attacks cannot be detected until some peer has completed downloading the entire generation. In network coding-based P2P content distribution, a corrupted block can rapidly pollute the network and, hence, by the time pollution attacks are detected, a lot of network bandwidth has been already wasted on transmitting corruption blocks. In dynamic P2P networks, because of the churn of malicious peers, the bandwidth waste would be more serious. Next, because the success of the identification relies on peers receiving all the checksums, this makes the identification vulnerable to collusion attacks. Multiple malicious peers may collude to prevent some peers from receiving the checksums. Moreover, the scheme cannot identify all the malicious peers for a kind of non-functional malicious peers cannot be identified. A peer is called as a non-functional malicious peer if it exhibits malicious behaviors but replacing it with a legitimate peer will not change the set of infected peers. More explanation can be found in [9].

The content pollution is a widespread problem in P2P networks, and not only exists in network coding-based P2P networks. Reputation systems are long-term solutions which are used to reduce content pollution and are mainly divided into three categories: peer reputation systems such as Eigentrust [25] and Scrubber [26], object reputation systems

such as Credence [27] and hybrid peer and object reputation systems such as Xrep [28]. In peer reputation systems, each peer assigns reputations to each other and malicious peers with low reputation will be identified, and in object reputation systems, peers assign reputations to the objects (files) they download regarding their authenticity and the object reputation is then used to decide whether the object should be downloaded. Hybrid peer and object reputation systems combine the benefits of both strategies. By maintaining reliable reputation information about peers or objects, reputation systems can form the basis of an incentive system and can guide peers in their decision making, thus reducing the content pollution in P2P networks.

In traditional P2P networks, hash authentication codes are used to check the validity of content blocks, thus providing the basis for reputation systems. When using network coding in P2P networks, content pollution is more serious due to the "combination" nature of network coding and traditional hash verification mechanism is no longer useful, so reputation systems cannot be established and run efficiently in this case. ESNC can thwart pollution attacks on network coding. Moreover, it can cooperate with reputation systems to cope with content pollution effectively in network coding-based P2P networks in the long run.

8 Conclusion

In this paper, we propose a novel and efficient ECC-based mechanism for securing network coding-based P2P content distribution, namely ESNC, which provides both an efficient corruption detection scheme and a precise attacker identification scheme for P2P systems to thwart network-coding pollution attacks. ESNC is mainly based on ECC and can provide high level of security at low computation and communication overheads. We also demonstrate that ESNC effectively limits the corruption spread and identifies all the malicious peers quickly under practical settings. Compared to the existing related state-of-the-art schemes, ESNC is especially suitable for P2P content distribution. It can achieve high security and has high overall performance. Despite the rich literature in network coding, we are not aware of any operational network coding-based P2P content distribution system except the commercial P2P media streaming distribution system UUSee [10]. We plan on developing a real system using network coding, deploying ESNC on it, and then reporting our experiences in future work.

Acknowledgments This work is supported by National Natural Science Foundation of China under grants 61173170, 60873225, 61300222 and 61303117, National High Technology Research and Development Program of China under grant 2007AA01Z403, Innovation Fund of Huazhong University of Science and Technology under grants 2013QN120, 2012TS052 and 2012TS053, and Innovation Fund of Wuhan University of Science and Technology under grants 2013xz012.

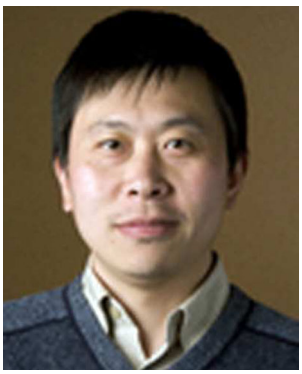
References

1. Gkantsidis C, Rodriguez P (2005) Network coding for large scale content distribution. Proc IEEE INFOCOM 2005:2235–2245
2. Ahlswede R, Cai N, Li SR, Yeung RW (2000) Network information flow. IEEE Trans Inf Theory 46(4):1204–1216
3. Zhu Y, Li B, Guo J (2004) Multicast with network coding in application layer overlay networks. IEEE J Sel Areas Commun 22(1): 107–120
4. Petrovic D, Ramchandran K, Rabaey J (2006) Overcoming untuned radios in wireless networks with network coding. IEEE Trans Inf Theory 52(6):2649–2657
5. Katti S, Rahul H, Hu W, Katabi D, Medard W, Crowcroft J (2008) Xors in the air: practical wireless network coding. IEEE/ACM Trans Networking 16(3):497–510
6. Gkantsidis C, Rodriguez P (2006) Cooperative security for network coding file distribution. Proceedings IEEE INFOCOM 2006:1–13
7. Jain K, Lovasz L, Chou PA (2005) Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. Proc ACM Symp Princ Distrib Comput 2005:51–59
8. Small T, Li B, Liang B (2008) Topology affects the efficiency of network coding in peer-to-peer networks. Proc IEEE ICC 2008: 5591–5597
9. Wang Q, Vu L, Nahrstedt K, Khurana H (2010) Identifying malicious nodes in network-coding-based peer-to-peer streaming networks. Proc IEEE INFOCOM 2010:1–5
10. Liu Z, Wu C, Li B, Zhao S (2010) UUSee: large-scale operational on-demand streaming with random network coding. Proc IEEE INFOCOM 2010:2070–2078
11. Liu F, Shen S, Li B, Li B, Yin H, Li S (2011) Novasky: cinematic-quality VoD in a P2P storage cloud. Proc IEEE INFOCOM 2011: 936–944
12. Li B, Niu D (2011) Random network coding in peer-to-peer networks: from theory to practice. Proc IEEE 99(3):513–523
13. Krohn M, Freedman M, Mazieres D (2004) On-the-fly verification of rateless erasure codes for efficient content distribution. Proc IEEE Symp Secur Priv 2004:226–240
14. Kehdi E, Li B (2009) Null Keys: limiting malicious attacks via null space properties of network coding. Proc IEEE INFOCOM 2009: 1224–1232
15. Yu Z, Wei T, Ramkumar B, Guan Y (2008) An efficient signature-based scheme for securing network coding against pollution attacks. Proc IEEE INFOCOM 2008:1409–1417
16. Charles D, Jain K, Lauter K (2009) Signatures for network coding. Int J Inf Coding Theory 1(1):3–14
17. Boneh D, Freeman D, Katz J, Waters B (2009) Signing a linear subspace: signature schemes for network coding. Proc PKC 2009: 68–87
18. Agrawal S, Boneh D (2009) Homomorphic MACs: MAC-based integrity for network coding. Proc ACNS 2009:292–305
19. Li Y, Yao H, Chen M, Jaggi S, Rosen A (2010) RIPPLE authentication for network coding. Proc IEEE INFOCOM 2010:2258–2266
20. Jaggi S, Langberg M, Katti S, Ho T, Katabi D, Medard M (2008) Resilient network coding in the presence of Byzantine adversaries. IEEE Trans Inf Theory 54(6):2596–2603

21. Koetter R, Kschischang FR (2008) Coding for errors and erasures in random network coding. IEEE Trans Inf Theory 54(8): 3579–3591
22. Zhang P, Jiang Y, Lin C, Yao H, Wasef A, Shen X (2011) Padding for orthogonality: efficient subspace authentication for network coding. Proc IEEE INFOCOM 2011:1026–1034
23. Darrel H, Alfred M, Scott V (2004) Guide to Elliptic Curve Cryptography. Springer, New York
24. Touceda DS, Sierra JM, Soriano M (2012) Decentralized certification scheme for secure admission in on-the-fly peer-to-peer systems. Peer-to-Peer Netw Appl 5(2):105–124
25. Kamvar S, Schlosser M, Molina HG (2003) The Eigentrust algorithm for reputation management in P2P networks. In Proceedings of WWW 2003:640–651
26. Costa C, Soares V, Almeida J, Almeida V (2007) Fighting pollution dissemination in peer-to-peer networks. Proc ACM Symp Appl Comput 2007:1586–1590
27. Walsh K, Sirer EG (2005) Fighting peer-to-peer SPAM and decoys with object reputation. In Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems, 2005:138–143
28. Damiani E, Vimercati S, Paraboschi S, Samarati P, Violante F (2002) A reputation-based approach for choosing reliable resources in peer-to-peer networks. Proc ACM Conf Comput Commun Secur 2002: 207–216
29. Barreto P, Kim H, Lynn B, Scott M (2002) Efficient algorithms for pairing-based cryptosystems. Proc CRYPTO 2002:354–368
30. Boneh D, Lynn B, Shacham H (2004) Short signatures from the Weil pairing. J Cryptol 17(4):297–319
31. Montresor A, Jelasity M (2009). PeerSim: A scalable P2P simulator. In Proceedings of the IEEE International Conference on Peer-to-Peer Computing, IEEE Computer Society, Washington, DC, 2009, 99–100
32. Zhao K, Chu X, Wang M, Jiang Y (2009) Speeding up homomorpic hashing using GPUs. Proc IEEE ICC 2009:856–860
33. Perrig A, Canetti R, Tygar JD, Song D (2002) The TESLA broadcast authentication protocol. RSA Cryptobytes 5(2):2–13
34. Jafarisiavoshani M, Fragouli C, Diggavi S (2008) On locating Byzantine attackers. Proc Netw Coding Work Theory Appl 2008:1–6

Heng He received the B.S. degree from School of Computer Science at Wuhan University of Technology in 2004 and his M.S. degree from School of Computer Science and Technology at Huazhong University of Science and Technology in 2007. Now he is a Ph.D. candidate in the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include P2P computing, cloud computing, network coding and network security.

Ruixuan Li received the B.S., M.S. and Ph.D. in Computer Science from Huazhong University of Science and Technology, China in 1997, 2000 and 2004 respectively. He was a Visiting Researcher in Department of Electrical and Computer Engineering at University of Toronto from 2009 to 2010. He is currently a Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include peer-to-peer computing, distributed data management, and distributed system security. He is a member of IEEE and ACM.



Zhiyong Xu received the received the B.S. and M.S. in Computer Science from Huazhong University of Science and Technology, China in 1994 and 1997 respectively., and Ph.D. degrees in Computer Engineering from University of Cincinnati in 2003. He is currently an Associate Professor in the Department of Mathematics and Computer Science at Suffolk University. His research interests include Peer-to-Peer Computing, High performance I/O and File systems, Multimedia Applications, Parallel and Distributed Computing. He is a member of IEEE.

Weijun Xiao received the B.S. and M.S. in computer science from Huazhong University of Science and Technology, China in 1995 and 1998 respectively, and Ph.D. in electrical engineering from University of Rhode Island, USA. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, Virginia Commonwealth University, USA. His research interests include computer architecture, networked storage system, embedded system, and performance evaluation. He is a member of the IEEE.