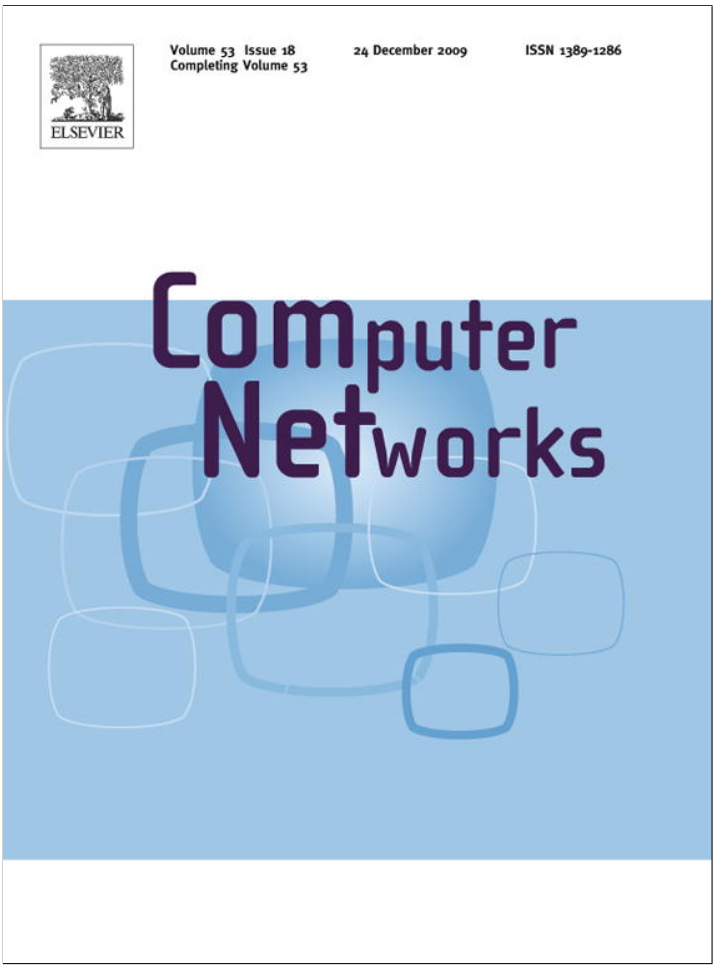


Computer Networks 53 (2009) 3089–3101

Contents lists available at ScienceDirect

Computer Networks

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

A self-adaptive probabilistic packet filtering scheme against entropy attacks in network coding

Yixin Jiang [a,b], Yanfei Fan [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, PR China

ARTICLE INFO

Article history:
Received 6 January 2009
Received in revised form 27 April 2009
Accepted 3 August 2009
Available online 8 August 2009
Responsible Editor: J. Misic

Keywords:
Network coding
Entropy attacks
Packet filtering
Probabilistic
Linear-dependency

ABSTRACT

In this paper, based on a novel self-adaptive probabilistic subset linear-dependency detection (S-PSLD) algorithm, we propose an efficient packet filtering scheme against entropy attacks in network coding. The scheme verifies the received packets probabilistically instead of exactly, and thus it can rapidly filter out the resultant packets from entropy attacks. Moreover, to minimize the packet detection cost at forwarder while keeping the false positive rate at an expected low level, a self-adaptive algorithm is introduced such that each forwarder can dynamically tune the system security parameters according to the available bandwidth or the number of the received packets in buffer. Theoretical analysis and performance evaluation are given to demonstrate the validity and efficiency of the proposed scheme.

1. Introduction

Network coding is a promising information dissemination approach for many practical network applications, such as traditional multicast or broadcast [1], peer-to-peer content distribution [4–7], and wireless access [2,3]. Network coding was first introduced in [8] as an alternative to the traditional packet-forwarding networks, and it has been shown that random linear coding can achieve the optimal throughput for multicast [1,9] and even unicast transmissions [10]. In network coding, intermediate nodes encode the packets en-route, thus each outgoing packet can be a linear combination of incoming packets received from the uplink nodes. The packet mixing operation enables that network coding can offer several potential advantages, such as network resource optimization, computation efficiency for packet scheduling, and robustness to dynamic network topology.

However, due to the packet encoding at intermediate nodes and multi-hop transmission, network coding based applications are susceptible to potential malicious attacks or resource abuse. Two primary types of attacks, *entropy attacks* and *pollution attacks* [5], are particularly relevant to network coding. The *pollution attacks* can be easily launched by injecting polluted information or modifying messages, which is fatal to the whole network coding system.

Unlike pollution attacks, *entropy attacks* can be regarded as a special replay attack, where an adversary may arbitrarily forge non-innovative packets that are trivial linear combinations of the "stale" packets that are collected or intercepted at an earlier time. Although replay attacks frequently occur in traditional packet-forwarding networks without network coding, the entropy attacks are more serious in network coding. If "stale" packets are mixed by a forwarder, outgoing packets may be non-innovative. Such packets still preserve the linear algebraic constraints on the original packets and the appended encoding

* Corresponding author.
E-mail addresses: yixin@bbcr.uwaterloo.ca (Y. Jiang), yfan@bbcr.uwaterloo.ca (Y. Fan), xshen@bbcr.uwaterloo.ca (X. (Sherman) Shen), chlin@ tsinghua.edu.cn (C. Lin).

1389-1286/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.comnet.2009.08.002

vectors. However, these packets with no new coding information will significantly decrease the information entropy of the system, and thus reduce the decoding opportunities at sinks and the overall throughput rate.

Therefore, to efficiently thwart entropy attacks, the non-innovative packets should be detected and filtered out as early as possible, since they may rapidly spread to all downstream nodes due to the subsequent packet encoding procedure. The existing the *cryptography-based* schemes [12,13] or *non-cryptography based* schemes [14,15] can only detect polluted packets, and almost all these schemes are susceptible to entropy attacks, where an adversary can easily create non-innovative packets with the "stale" packets. A naive method against entropy attacks is to directly detect linear-dependency of all the received packets [5]. Nevertheless, the approach is computationally inefficient, as a result of the time-consuming arithmetic operations on a large finite field $\mathbb{Z}_q$, where q is a 256 bit or larger primer, and the considerably large number of symbols in each packet vector, e.g., $m + n = 384$ in content distribution networks [5,13]. The delay caused by testing the linear dependency of a bulk of packets may radically decrease transmission throughput rate and further impair the system scalability. Moreover, though the slightly-redundant non-innovative packets can improve transmission reliability in some unreliable communication scenarios, such entropy attacks can abuse system resources and thus are prone to evolve into a special type of DoS attacks. All such deficiencies motivate us to explore an efficient entropy-attack-tolerant scheme for network coding.

In this paper, based on a novel self-adaptive probabilistic subset linear-dependency detection (S-PSLD) algorithm, we propose an efficient packet filtering scheme for network coding against entropy attacks. To the best of our knowledge, this is the first efficient probabilistic solution to address the entropy attack issue in network coding. The proposed scheme offers the following significant features: (1) *Efficiency*: The proposed scheme supports fast packet filtering. By testing the received packets probabilistically instead of exactly, a node can rapidly determine the linear-dependency of multiple packets in batch such that the total computation overhead can be dramatically reduced. Thus, the performance bottleneck is effectively eliminated due to the lightweight computational overhead at forwarders; (2) *Self-adaptability*: To minimize the packet detection cost at forwarder while keeping the false positive rate at an expected low level, each forwarder can dynamically tune the system security parameters according to the available bandwidth or the size of the received packets in buffer. Theoretical analysis and simulation evaluation also demonstrate the validity and efficiency of the optimized policies.

The remainder of the paper is organized as follows. In Section 2, the related works are surveyed. In Section 3, problems related to the proposed scheme are described, including the network coding and threat model, respectively. In Section 4, the proposed scheme against entropy attacks is introduced in detail. In Section 5, the performance metrics are presented, followed by the conclusions in Section 6.

2. Related works

For securing network coding, some efficient schemes have been proposed, which can be primarily divided into two categories from the viewpoint of cryptography.

The *cryptography-based schemes* primarily contain secure random checksum scheme [5], homomorphic hash scheme [5,22,24], homomorphic signature scheme [12,13], homomorphic privacy scheme [11]. In [22], Krohn et al. present a practical scheme for secure 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 a linear combination of original file blocks. Li et al. [24] propose a security scheme for batch content distribution, which is based on an invertible trapdoor homomorphic hash function. Compared to the Krohn et al.' scheme, the computational and the bandwidth overhead are significantly reduced. Gkantsidis et al. [5] extend Krohn et al.'s approach and present a homomorphic hashing scheme for secure peer-to-peer content distribution. The scheme considerably reduces the verifying cost while efficiently preventing the propagation of malicious blocks. However, the scheme needs an extra secure channel for the source to distribute its hash values to all nodes. Based on Weil pairing [18], Charles et al. [12] design a different homomorphic signature scheme, which requires no secure channel. However, the scheme relies on the time-consuming pairing operations over elliptic curves, and cannot support batch verification [19,20]. Therefore, the packet verification cost is very heavy in the scheme. Yu et al. [13] propose an efficient RSA-based signature scheme, which inherits the basic framework of Charles et al.'s scheme and the homomorphic hash function of Gkantsidis et al. scheme. In Yu et al.'s scheme, the forwarders can achieve efficient verification at the expense of increased transmission overhead, due to the large size of a RSA signature. Recently, based on homomorphic privacy, Fan et al. [11] present an efficient privacy-preserving scheme for Network Coding. With homomorphic encryptions on Global Encoding Vectors, the scheme offers two significant privacy-preserving features, packet content confidentiality and flow un-traceability, to efficiently thwart the traffic analysis attacks.

The *non-cryptography-based schemes* primarily offer an end-to-end security, where the intermediate nodes do not involve in detecting or filtering out the fake packets. In [14,23], Ho et al. propose an information-theoretically secure scheme to detect Byzantine attacks without using cryptographic functions. In the scheme, a computation-efficient hash value is embedded into each packet. The sinks can use the hashes to detect packet integrity with high probability. Jaggi et al. [15] introduce a distributed polynomial-time rate-optimal network code to thwart Byzantine adversaries. By embedding extra parity information into the source packets, 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 against adversaries with different attacking

capabilities. However, similar to Ho's scheme, Jaggi's scheme can only detect Byzantine attacks at sinks, instead of forwarders. Since it cannot drop the bogus packets enroute, the scheme is unsuitable for resource-limited networks. Wang et al. [21] introduce a broadcast-mode transformation for network coding, which can efficiently impede the influence of potential adversaries. In the scheme, with a sufficient diversity of internally-disjointed 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 to stop adversaries, especially in such applications that cost-prohibitive approaches may be infeasible.

The existing security schemes for network coding emphasize defense from pollution attacks, while entropy attacks have received little attentions. Gkantsidis et al. [5] first introduced the concept of entropy attacks. However, they mainly attempted to address pollution attacks for network coding based content distribution networks. As for entropy attacks, they only used a directly linear-dependency test method to detect entropy attacks. Such intuitive approach may be computationally inefficient, considering that: (1) The symbols in each packet vector are defined on a large finite field $\mathbb{Z}_q$, such as 256 bits in [5]; (2) The size of each packet vector may be considerably large ($m+n=384$ in [5]). These time-consuming arithmetic operations may significantly decrease transmission throughput rate and further impairs the system scalability. Moreover, the cooperation security [5] among the nodes incurs extra communication overhead, which is even unfeasible in communication-intermittent application scenarios.

3. Problem statement

In this section, we first present the description of the preliminary network coding model, followed by the threat model.

3.1. Network coding model

An overview of network coding and possible applications is given in [16]. The principle in network coding is to allow intermediate nodes to encode the incoming packets. Fig. 3 depicts a typical node with three incoming links and one outgoing link. Packets arrive sequentially through each link and enter into a buffer. Once there is a transmission opportunity on an outgoing link, an outgoing packet is formed by taking a random linear combination of incoming packets, as shown in Fig. 1. The transmission opportunity, for example, can occur in wireless links when the MAC gets access to channel.

In our scheme, there is an acyclic network (V,E,c), which consists of a set of nodes V and a set of directed links (or edges) E with unit capacity edges, i.e., $c(e)=1$ for all $e\in E$. Let $\Gamma^+(v)$ be the edges emanated from a node v, $\Gamma^-(v)$ be the edges entering into a node v, and $y(e)\in\mathbb{Z}_q$ be the symbol carried on edge e, where q is a primer. Moreover, there are a single source $s\in V$ and a set of sinks $T\subseteq V$, and the source s attempts to send continuous data blocks to the sinks $T\subseteq V$. Each block can be divided into n packets $\{\boldsymbol{B}_1,\ldots,\boldsymbol{B}_n\}$, where $n=MinCut(s,T)$ is the multicast capacity. Similar to the setting in [5,12], each original packet $\boldsymbol{B}_i$ $(i=1,\ldots,n)$ is further divided into m symbols, that is, $\boldsymbol{B}_i=[b_{i,1},b_{i,2},\ldots,b_{i,m}]$, where $b_{i,j}\in\mathbb{Z}_q$ $(1\leqslant i\leqslant n, 1\leqslant j\leqslant m)$ denote the symbols.

In a real network, symbols sequentially carried on edges can be grouped into packets. To facilitate the decoding at the sinks, each packet carried on edge $e\in\Gamma^+(v)$ is appended with its corresponding global encoding vector $\boldsymbol{G}(e)$. For this, each source packet $\widetilde{\boldsymbol{B}}_i$ is augmented by prefixing the ith original packet vector $\boldsymbol{B}_i$ with the ith unit vector $\boldsymbol{U}_i$.

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

Correspondingly, as shown in Fig. 1, the packet $\widetilde{\boldsymbol{Y}}(e)$ on edge $e\in\Gamma^+(v)$ can be computed as a linear combination of the packet $\widetilde{\boldsymbol{Y}}(e')$ on edges $e'\in\Gamma^-(v)$ or, alternatively, as a linear combination of the packets $\{\widetilde{\boldsymbol{B}}_1,\ldots,\widetilde{\boldsymbol{B}}_n\}$ by induction, namely,

$$\begin{aligned}\widetilde{\boldsymbol{Y}}(e) &= \sum_{e'\in\Gamma^-(v)}\beta_{e'}(e)\widetilde{\boldsymbol{Y}}(e') = \sum_{i=1}^{n} g_i(e)\cdot\widetilde{\boldsymbol{B}}_i\\ &= \left[\sum_{i=1}^{n} g_i(e)\cdot b_{i,1},\ldots,\sum_{i=1}^{n} g_i(e)\cdot b_{i,m},g_1(e),\ldots,g_n(e)\right]\\ &= [y_1(e),\ldots,y_m(e),g_1(e),\ldots,g_n(e)],\end{aligned}\tag{2}$$

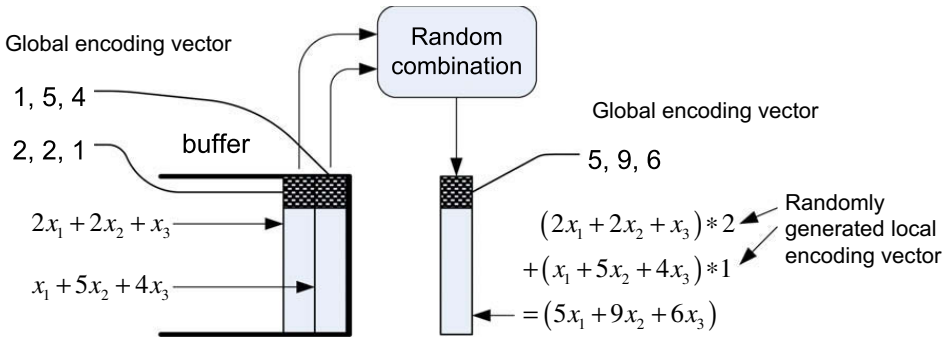


Fig. 1. Encoding incoming packets and forming an outgoing packet.

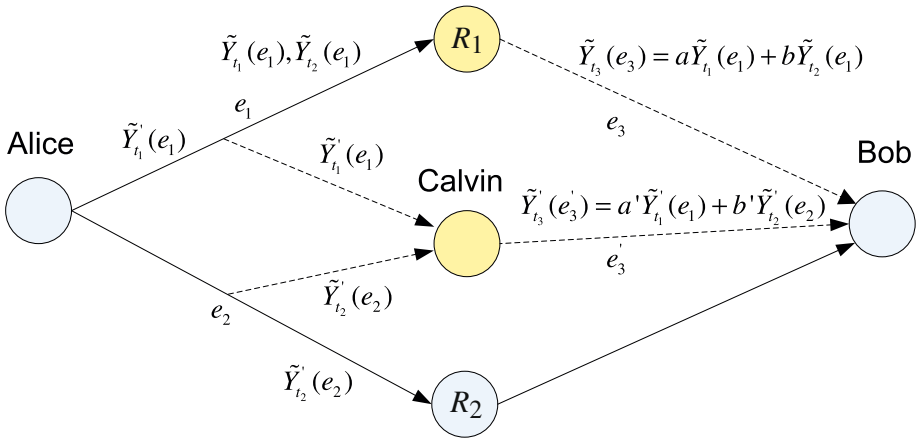


Fig. 2. Entropy attack model: injecting non-innovative packets by inside attackers (malicious forwarder) or outside attackers (Calvin).

where the coefficient $\beta_{e'}(e)$ is a random number. The coefficients, $\boldsymbol{G}(e) = [g_1(e), \ldots, g_n(e)]$, form a *global encoding vector* (GEV) on edge $e \in \Gamma^+(v)$, which is computed recursively as $\boldsymbol{G}(e) = \sum_{e' \in \Gamma^-(v)} \beta_{e'}(e)\boldsymbol{G}(e')$ with a *local encoding vector* (LEV) $\beta(e) = [\beta_{e' \in \Gamma^-(v)}(e)]$ on edge $e \in \Gamma^+(v)$. Once a sink $t \in T$ receives n packet s $\widetilde{\boldsymbol{Y}}(e_k)$ $(1 \leqslant k \leqslant n)$, it can recover the source packets with high probability [17].

3.2. Threat model

In the proposed scheme, the source is always trusted, and the forwarders may be not trusted. Under these assumptions, network coding is generally vulnerable to entropy attacks or pollution attacks. Here we focus on the entropy attacks.

An inside attacker (malicious node) or an outside attacker (Calvin) may forge non-innovative packets that are linear combinations of the "stale" encoded packets which are intercepted or collected by this node. The non-innovative packets injected into the network can rapidly diffuse the whole downstream nodes. Such attacks abuse system resource, which may significantly decrease information entropy of the links, and further reduce the opportunity of packet decoding at sinks and the throughput rate of the system.

As shown in Fig. 2, Alice transmits data to Bob. An inside attacker (malicious node) or an outside attacker (Calvin) attempts to thwart the information flow from Alice to Bob, or at least minimize it. Formally, we describe the entropy attacks as follows.

(1) *Inside attacker*: Consider that Alice has sent two packets $\widetilde{\boldsymbol{Y}}_{t_1}(e_1)$ and $\widetilde{\boldsymbol{Y}}_{t_2}(e_1)$ to a downstream node R_1 at time t_1 and t_2, respectively. Then the malicious node R_1 can arbitrarily forge non-innovative packets at the later time t_3 which is the linear combinations of the two "stale" packets, $\widetilde{\boldsymbol{Y}}_{t_1}(e_1)$ and $\widetilde{\boldsymbol{Y}}_{t_2}(e_1)$ as follows:

$$\widetilde{\boldsymbol{Y}}_{t_3}(e_3) = a\widetilde{\boldsymbol{Y}}_{t_1}(e_1) + b\widetilde{\boldsymbol{Y}}_{t_2}(e_1), \tag{3}$$

where a and b are arbitrary random integers.

(2) *Outsider attacker*: Similar to the inside node, once a outside attacker (Calvin) successfully intercepts two "stale" packets sent from Alice to node R_1 and R_2, $\widetilde{\boldsymbol{Y}}'_{t_1}(e_1)$ and $\widetilde{Y}'_{t_2}(e_2)$, then it can use the two packets to launch a replay attack at the later time t_3 by forging non-innovative packets as $\widetilde{\boldsymbol{Y}}'_{t_3}(e'_3) = a'\widetilde{\boldsymbol{Y}}'_{t_1}(e_1) + b'\widetilde{\boldsymbol{Y}}'_{t_2}(e_2)$, where a' and b' are arbitrary random integers.

The forged packets generated by entropy attacks satisfy the linear algebraic constraints on the original packets and the appended global encoding vectors, and thus they are linearly dependent on the "stale" encoded packets. This intuitive observation inspires us to explore a computationally-efficient entropy-attack-tolerant scheme by testing the linear dependency of the received packets.

In addition, as shown in Fig. 2, an adversary can also generate polluted packets, where the packets do not preserve the linear algebraic constraints on the original packets and the encoding vectors. A packet $\widetilde{\boldsymbol{Y}}(e)$ is polluted if the vector $\widetilde{\boldsymbol{Y}}(e)$ is not equal to the product of the original augmented packet $[\widetilde{\boldsymbol{B}}_1, \ldots, \widetilde{\boldsymbol{B}}_n]$ and the global encoding vector $\boldsymbol{G}(e) = [g_1(e), \ldots, g_n(e)]$, i.e.,

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

As the schemes in [5,12,13,24], such pollution attacks can be thwarted via digital signatures.

4. An efficient packet filtering scheme against entropy attacks

In this section, with a novel Self-adaptive Probabilistic Subset Linear-dependency Detection (S-PSLD) algorithm, we propose an efficient packet filtering scheme against entropy attacks in network coding.

4.1. The proposed probabilistic packet filtering scheme

Unlike pollution attacks [5], the packets forged by entropy attacks seem to be "normal" packets, since it still holds the linear algebraic constraints on the packets and the global encoding vectors. However, an evident feature

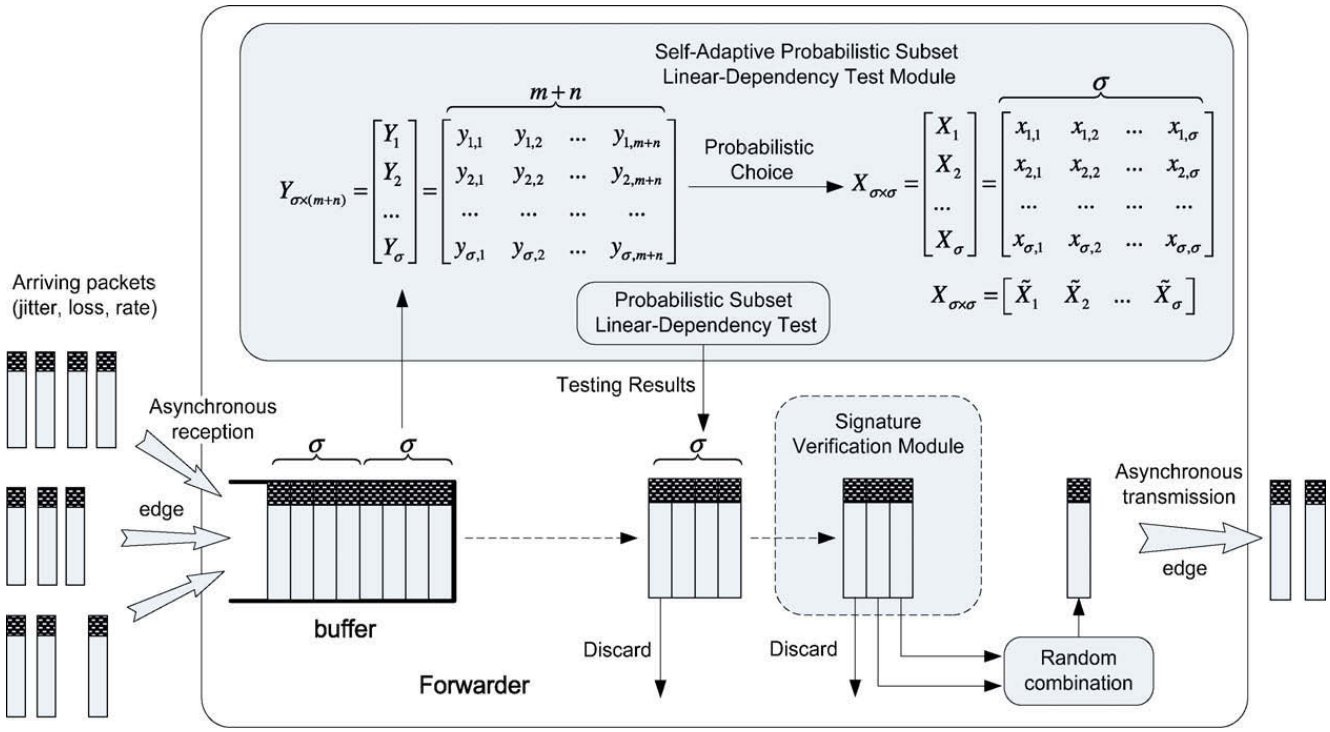


Fig. 3. Basic probabilistic packet filtering architecture with the S-PSLD module.

associated to such packets is that they may linearly depend on some "stale" encoded packets. A naive algorithm against entropy attacks is to determine the linear dependency of the received packets with Gaussian Elimination. However, this algorithm is computationally inefficient, considering the time-consuming arithmetic operations on a large finite field $\mathbb{Z}_q$ and the considerably large number of symbols in each packet (e.g. $m+n=384$ in [5,13]). To reduce the computation cost when directly testing the linear dependency of the packets, we introduce a novel *Self-adaptive Probabilistic Subset Linear-dependency Detection* (S-PSLD) algorithm, as shown in Algorithm 1. By verifying all packet vectors probabilistically instead of exactly, this algorithm can rapidly detect and filter out the resultant non-innovative packets of entropy attacks. Fig. 3 shows the basic probabilistic packet filtering architecture with the S-PSLD module.

(1) As packets arrive at forwarder v through different uplinks $e_i \in \Gamma^-(v)$ asynchronously, they are first put into a shared buffer sequentially.
(2) To detect entropy attacks, the forwarder is required to determine the linear-dependency of the received packet vectors in the buffer. For this, the forwarder first takes σ packets at the head of the packet queue in the buffer, and combine them into a new matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)} = [\widetilde{\boldsymbol{Y}}_1, \widetilde{\boldsymbol{Y}}_2, \ldots, \widetilde{\boldsymbol{Y}}_\sigma]^T$.
(3) To reduce the computation overhead of directly testing linear dependency of matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$, the forwarding node generates a sub-matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma} \subseteq \widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$, where each column vector $\widetilde{\boldsymbol{X}}_i \in \widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ $(1 \leqslant i \leqslant \sigma)$ is picked from matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ in an independent and random way. Then it further performs Gaussian Elimination to test the linear dependency of matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$. According to the testing results of matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$, the corresponding linear-dependent packets in matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ are discarded; the other packets in matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ are accepted and further used for network coding.
(4) The S-PSLD cannot detect the polluted packets. As shown in Fig. 3 with dashed grey rectangle, the Signature Verification Module can filter out the polluted packets by authenticating the digital signature of each packet, such as in [5,11,13,24].
(5) Once the number of linearly-dependent packets on an uplink $e_i \in \Gamma^-(v)$ exceeds a threshold value, the forwarder v may consider that the upstream node associated to the edge $e_i \in \Gamma^-(v)$ is a malicious one. Some defense policies, such as spread alert message or cutting off this link, can be further performed to thwart entropy attacks.

Evidently, the primarily computation overhead is dominated by the computation complexity of performing Gaussian Elimination.

4.2. Self-adaptive optimization policies

The S-PSLD algorithm is designed to circumvent the expensive computations of Gaussian Elimination, when we directly test the linear-dependency for a given large number of vectors. The key optimization policy is to introduce two security parameters, σ and ρ, which denote the dimension of the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ and the pre-defined maximal repeated times of one PSLD test (Algorithm 1, Lines 09–20), respectively. According to the performance metrics in Section 5, the parameter ρ is a very small integer and σ is far less than $m+n$. Therefore, compared with the directly linearly-

Algorithm 1. Probabilistic subset linear-dependency detection

00: **Function** ***Self_adaptive_Probablistic_Subset_Linear_Dependency_Detection***() {
01: *nc* = 0;/* Variable *nc* records the number of the received linearly-dependent vectors */
02: **while** (the input buffer of the forwarder is not empty) {
03: Choosing a parameter value a according to transmission bandwidth and the number of packet vectors in buffer;
04: Taking σ packet vectors $\widetilde{\boldsymbol{Y}}_1, \widetilde{\boldsymbol{Y}}_2, \ldots, \widetilde{\boldsymbol{Y}}_\sigma$ from the input buffer; and combine them into a matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)} = [\widetilde{\boldsymbol{Y}}_1, \widetilde{\boldsymbol{Y}}_2, \ldots, \widetilde{\boldsymbol{Y}}_\sigma]^{\mathrm{T}}$;
05: Choosing a reasonable security parameter $\rho = \lfloor(-\log_2\delta)/\sigma\rfloor$, according to the value σ and the expected false positive rate δ;
06: $k = 0$;
07: *Flag* = *FALSE*;/* Linearly-independent Flag */
08: **while** $((k < p)\&\&(Flag! = TRUE))$ {
09: Creating a vector subset $\widetilde{X}_{\sigma\times\sigma} = [\widetilde{X}_1, \widetilde{X}_2, \ldots, \widetilde{X}_\sigma]$, where each column vector $\widetilde{X}_k (1 \leqslant k \leqslant \sigma)$ is picked from the matrix $\widetilde{Y}_{\sigma\times(m+n)}$ in an independent and uniform way;
10: Using Gaussian Elimination to get the Echelon of the matrix $\widetilde{X}_{\sigma\times\sigma}$;
11: **if** $(\mathrm{rank}(\widetilde{X}_{\sigma\times\sigma}) = \sigma)$ /* linearly-independent */
12: **break;**/* linearly-independent, exit while-cycle */
13: **else**{/* possible linearly-dependent, maybe need perform another test */
14: $k = k + l$;
15: **if** $(k > p)$ {/* reaching the allowable maximal number of linearly-dependent test */
16: $nc = nc + 1$;
17: **if** $(nc > \gamma)$ {/* γ is a relax factor, which is the number of allowable linearly-dependent vectors */
18: Discarding the corresponding linearly-dependent packet vectors in matrix $\widetilde{\boldsymbol{Y}}_{a<m+n)}$;
19: *Flag* = *TRUE;*
20: }
21: }/* end of cycle while */
22: Generating outgoing packets as the random linear combinations of the independent packets in matrix $Y_{\sigma\times(m+n)}$;
23: Flushing all the counterparts in the buffer corresponding to the packets in matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$;
24: }/* end of cycle while */
25: }

dependent test, the S-PSLD algorithm is computation-efficient, and can efficiently tolerate entropy attacks, since the forged non-innovative packets that are the linear combinations of the "stale" packets can be rapidly dropped.

To guarantee the validity and efficiency of the S-PSLD algorithm, the following two critical issues should be clearly addressed.

(1) The false positive rate $f(\rho, \sigma)$: In S-PSLD algorithm, since we test packet vectors probabilistically instead of exactly, this may incur the false alarm. That is, the row vectors of matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ are actually linearly-independent, while the test result of the corresponding row vectors of matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ may be linearly-dependent. Therefore, it is required to derive the *false positive rate* $\delta = f(\rho, \sigma)$ as the function of parameter σ (the dimension of matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$) and ρ (the pre-defined maximal repeated times of individual PSLD test).
(2) Minimizing the computation cost of the S-PSLD algorithm: Given an expected low false positive rate $\delta = f(\rho, \sigma)$, how to determine the security parameter σ and ρ, such that the computation cost of the S-PSLD algorithm can be minimized.

Note that in some unreliable communication scenarios, such as delay/disruption tolerant networks (DTNs) [25,26], a slightly number of redundant non-innovative packets can efficiently improve transmission reliability. To efficiently balance the performance and security, we introduce a relax factor γ (Algorithm 1, Line 17), which enables that the scheme can tolerate the allowable number of linearly-dependent packets. Evidently, the relax factor γ is a link-status aware parameter, which is dependent on the different communication channels. How to determine a reasonable relax factor γ may be another open and complex issue, which exceeds the scope of this paper.

To guarantee the false positive rate $\delta = f(\rho, \sigma)$ at an expected low level, a forwarder is required to dynamically tune its security parameter (σ and ρ) and the relaxing factor γ, such that it could adapt itself to the available transmission bandwidth or the increasing number of the received packets.

5. Performance metrics

In this section, we derive the false positive rate and optimize the computation cost for the S-PSLD algorithm by choosing two reasonable security parameters, σ and ρ.

5.1. False positive rate

Algorithm 1 detects packets probabilistically instead of exactly, so it may incur the false alarm, i.e., the matrix

$\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is linearly-independent, while the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ is linearly dependent. The following Lemma gives the false positive rate for each individual PSLD test (Algorithm 1, Lines 09–20).

Lemma 1. *Let $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ be a matrix such that each row vector of $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is a packet vector picked from the buffer of a node, and $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}\subseteq\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is the subset randomly generated from $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$. For a given security parameter σ (the dimension of the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$), the false positive rate $\lambda=f(\sigma)$ for each individual PSLD test can be computed as $f(\sigma)=2^{-\sigma}$. Accordingly, the true positive rate for one PSLD test is $s(\sigma)=1-f(\sigma)=1-2^{-\sigma}$ (The proof can be seen in the Appendix A).*

The parameter ρ is the pre-defined maximal repeated times of one individual PSLD test, which is determined by the expected false positive rate $\delta=f(\rho,\sigma)$ and the dimension of the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$.

$$\rho=\left\lceil\frac{-\log_2\delta}{\sigma}\right\rceil=\left\lceil\frac{-\log_2 f(\rho,\sigma)}{\sigma}\right\rceil. \tag{5}$$

Eq. (5) can be derived from Lemma 2, which gives the false positive rate $\delta=f(\rho,\sigma)$ as the function of parameter σ and ρ, for the S-PSLD Algorithm (Algorithm 1, Lines 08–21).

Lemma 2. *For the given parameter σ and ρ, the false positive rate $\delta=f(\rho,\sigma)$ in the S-PSLD is $f(\rho,\sigma)=2^{-\rho\sigma}$, and the true positive rate is $s(\rho,\sigma)=1-2^{-\rho\sigma}$.*

In Algorithm 1, each subset matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}\subseteq\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is randomly generated from matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$, and each individual PSLD test is mutually independent. Hence, according to Lemma 1, the true positive rate $s(\rho,\sigma)$ can be calculated as

$$\begin{aligned}s(\rho,\sigma)&=s(\rho)+f(\rho)s(\rho)+\cdots+f(\rho)s^{\sigma-1}(\rho)\\&=(1-2^{-\rho})+(1-2^{-\rho})(2^{-\rho})+\cdots+(1-2^{-\rho})(2^{-\rho})^{\sigma-1}\\&=1-2^{-\rho\sigma}\end{aligned} \tag{6}$$

Lemma 2 shows a nice property of the S-PSLD algorithm, in which the false positive rate $\delta=f(\rho,\sigma)$ only relies on the security parameter σ and ρ, and irrelevant to m and n, i.e., $f(\rho,\sigma)=2^{-\rho\sigma}$. For practical application, it is enough to choose $\rho\sigma\geqslant 12$, since $s(\rho,\sigma)=1-2^{-12}=99.98\%$. Therefore, the S-PSLD is computationally efficient.

5.2. Computation overhead and comparisons

The primary computation cost $C_{\mathrm{PSLD}}(\rho,\sigma)$ in Algorithm 1 is dominated by the Gaussian Elimination operation. Here, we adopt a modified Gaussian Elimination algorithm to avoid the time-consuming modular multiplicative inverse operation over a finite field $\mathbb{Z}_q$, where q is a 256-bit long prime in [5,13]. For example, in the first round elimination, the ith row vector $\boldsymbol{X}_i\in\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ is eliminated by the first row vector $\boldsymbol{X}_1\in\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ as $\boldsymbol{X}_i'=x_{1,1}\boldsymbol{X}_i-x_{i,1}\boldsymbol{X}_1$, where $x_{1,1}\in\boldsymbol{X}_1$ and $x_{i,1}\in\boldsymbol{X}_i$ are the first element of packet vector $\boldsymbol{X}_1$ and $\boldsymbol{X}_i$, respectively. Let $C_{\mathrm{Guass}}(\widetilde{\boldsymbol{X}}_{\sigma\times\sigma})$ denote the computation cost to perform one modified Gaussian Elimination operation for matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$, which is

$$\begin{aligned}C_{\mathrm{Guass}}(\widetilde{\boldsymbol{X}}_{\sigma\times\sigma})&=\sum_{k=1}^{\sigma}2k(k-1)\cdot C_{\mathrm{Mul}}=2\left(\sum_{k=1}^{\sigma}k^2-\sum_{k=1}^{\sigma}k\right)C_{\mathrm{Mul}}\\&=\frac{2\sigma(\sigma^2-1)}{3}C_{\mathrm{Mul}},\end{aligned} \tag{7}$$

where C_{Mul} is the computation cost to perform one modular multiplication operation over the finite filed $\mathbb{Z}_q$. We neglect all other trivial operations such as modular addition operations.

Evidently, $C_{\mathrm{PSLD}}(\rho,\sigma)$ is determined by the two parameters, σ and ρ. To derive the expectation of $C_{\mathrm{PSLD}}(\rho,\sigma)$, we first give the state transitions of the S-PSLD algorithm with a Markov chain, as shown in Fig. 4. We assume that occurrences of the false positive rate for each individual PSLD test are random and mutually independent. Assume that p_s denote the probability that the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is linearly independent. Let state S_k $(0\leqslant k\leqslant\rho-1)$ denote the individual PSLD test for the kth randomly-generated matrix $\widetilde{\boldsymbol{X}}^{(k)}_{\sigma\times\sigma}\subseteq\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$, and state S_{-1} be termination state. The state transition is triggered according to the probabilistic test result. From Lemma 1, the false positive rate for each individual PSLD test is $\lambda=f(\sigma)=2^{-\sigma}$.

According to the state transitions in Fig. 4, the expected value of $C_{\mathrm{PSLD}}(\rho,\sigma)$ is calculated as

$$\begin{aligned}E[C_{\mathrm{PSLD}}(\rho,\sigma)]&=E\left[C_{\mathrm{Guass}}\left(\widetilde{\boldsymbol{X}}^{(0)}_{\sigma\times\sigma}\right)+\sum_{k=1}^{\rho-1}I(\widetilde{\boldsymbol{Z}}_k)\cdot C_{\mathrm{Guass}}\left(\widetilde{\boldsymbol{X}}^{(k)}_{\sigma\times\sigma}\right)\right]\\&=C_{\mathrm{Guass}}\left(\widetilde{\boldsymbol{X}}^{(0)}_{\sigma\times\sigma}\right)+\sum_{k=1}^{\rho-1}E\left[I\left(\widetilde{\boldsymbol{Z}}_k\right)\cdot C_{\mathrm{Guass}}\left(\widetilde{\boldsymbol{X}}^{(k)}_{\sigma\times\sigma}\right)\right],\end{aligned} \tag{8}$$

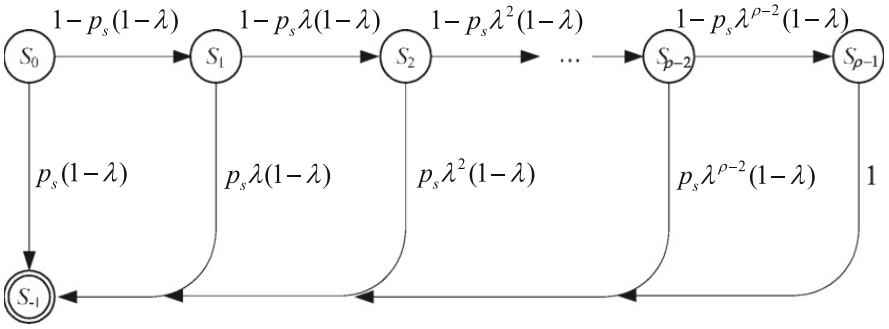


Fig. 4. The state transitions of the S-PSLD algorithm.

where the variable $\widetilde{Z}_k$ lim denotes an event such that an individual PSLD test for matrix $\widetilde{X}^{(k)}_{\sigma\times\sigma}$ is trigged, and accordingly, $I(\widetilde{Z}_k)$ is an indicator random variable function, which is defined as

$$I(\widetilde{Z}_k)=\begin{cases}1, & \text{if event } \widetilde{Z}_k \text{ occurs;}\\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

Let $P\left(\widetilde{Z}^+_k\right)$ and $P\left(\widetilde{Z}^-_k\right)$ denote the linearly-independent and linearly-dependent probability of the individual PSLD test for the matrix $\widetilde{X}^{(k)}_{\sigma\times\sigma}$ (Algorithm 1, Line 08), respectively. From the state transitions in Fig. 4, for $1\leqslant k\leqslant \rho-1$, we have

$$\begin{aligned}E(I(\widetilde{Z}_k)) &= 1\cdot P\left(\widetilde{Z}^+_k|\widetilde{Z}^+_{k-1}\right)\cdot P\left(\widetilde{Z}^+_{k-1}\right)\\ &\quad +0\cdot P\left(\widetilde{Z}^-_k|\widetilde{Z}^+_{k-1}\right)\cdot P\left(\widetilde{Z}^+_{k-1}\right)\\ &= P\left(\widetilde{Z}^+_k|\widetilde{Z}^+_{k-1}\right)\cdot P\left(\widetilde{Z}^+_{k-1}|\widetilde{Z}^+_{k-2}\right)\cdots P\left(\widetilde{Z}^+_2|\widetilde{Z}^+_1\right)\cdot P\left(\widetilde{Z}^+_0\right)\\ &= (1-p_s(1-\lambda))(1-p_s\lambda(1-\lambda))\cdots(1-p_s\lambda^{k-1}(1-\lambda)).\end{aligned} \tag{10}$$

Considering that the computation overhead $C_{Guass}(\widetilde{X}^{(k)}_{\sigma\times\sigma})$ for different matrix $\widetilde{X}^{(k)}_{\sigma\times\sigma}$ $(1\leqslant k\leqslant\rho)$ is the same, according to Eqs. (10) and (8) can be reduced to

$$\begin{aligned}E[C_{\mathrm{PSLD}}(\rho,\sigma)] &= C_{\mathrm{Guass}}(\widetilde{X}_{\sigma\times\sigma})\left(1+\sum_{k=1}^{\rho-1}E(I(\widetilde{Z}_k))\right)\\ &= C_{\mathrm{Guass}}(\widetilde{X}_{\sigma\times\sigma})\left\{1+\sum_{k=1}^{\rho-1}\left(\prod_{i=1}^{k}(1-p_s\lambda^{i-1}(1-\lambda))\right)\right\}.\end{aligned} \tag{11}$$

Therefore, according to Eq. (7) and the Lemma 1 $(\lambda=2^{-\sigma})$, the expected computation cost of testing each individual packet, $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$, can be normalized with σ as

$$\begin{aligned}&\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)\\ &=\frac{E[C_{\mathrm{PSLD}}(\rho,\sigma)]}{\sigma}=C_{\mathrm{Mul}}\frac{2(\sigma^2-1)}{3}\\ &\quad\times\left\{1+\sum_{k=1}^{\rho-1}\left(\prod_{i=1}^{k}\left(1-p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\right)\right\}.\end{aligned} \tag{12}$$

Especially, consider two extreme entropy attack scenarios as follows.

(1) *Heavy entropy attacks* $(p_s=0)$: Under this assumption, all the matrixes $\widetilde{Y}_{\sigma\times(m+n)}$ are linear dependent, we have

$$\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)=C_{\mathrm{Mul}}\frac{(\sigma^2-1)}{3}\rho. \tag{13}$$

(2) *No entropy attacks* $(p_s=1)$: Under this assumption, all the matrixes $\widetilde{Y}_{\sigma\times(m+n)}$ are linear independent, thus,

$$\begin{aligned}\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma) &= C_{\mathrm{Mul}}\frac{2(\sigma^2-1)}{3}\\ &\quad\times\left\{1+\sum_{k=1}^{\rho-1}\left(\prod_{i=1}^{k}\left(1-\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\right)\right\}.\end{aligned} \tag{14}$$

From Lemma 2, to assure the false positive rate within an accepted low level, it is enough to choose system parameter as $12\leqslant\rho\sigma\leqslant30$, since under the assumption of $12\leqslant\rho\sigma\leqslant30$ we have $2^{-30}\leqslant f(\rho,\sigma)\leqslant2^{-12}$ or $99.9756\%\leqslant s(\rho,\sigma)\leqslant99.9999999\%$. In addition, it is reasonable to assume that the dimension of each randomly-generated matrix $\widetilde{X}_{\sigma\times\sigma}\subseteq\widetilde{Y}_{\sigma\times(m+n)}$ satisfies $\sigma\geqslant3$.

Therefore, the optimization objective for the normalized computation cost $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ can be described as: for a given false positive rate $\delta=f(\rho,\sigma)$, it is required to determine the parameter σ and ρ, such that $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ is minimized, i.e.,

$$\begin{aligned}\text{Minimize}: &\quad \widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)=C_{\mathrm{Mul}}\frac{2(\sigma^2-1)}{3}\\ &\quad\times\left\{1+\sum_{k=1}^{\rho-1}\left(\prod_{i=1}^{k}\left(1-p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\right)\right\},\\ \text{Subject to}: &\quad \sigma\geqslant3,\quad 2^{-24}\leqslant f(\rho,\sigma)=2^{-\rho\sigma}\leqslant2^{-12},\\ &\quad 0\leqslant p_s\leqslant1.\end{aligned} \tag{15}$$

To directly optimize Eq. (15) is not trivial. However, considering the constraint conditions on parameter σ and ρ, they can only be assigned in a very small scope. Thus we can evaluate the normalized computation overhead $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ in an enumerative way, by assigning all possible values for σ and ρ, as illustrated in the subsequent section. For example, when $\rho\sigma=12$ and $\sigma\geqslant3$, the parameter pair (ρ,σ) can be assigned as (4,3), (3,4), (2,6), (1,12), respectively.

Moreover, we can further derive the boundaries for Eq. (12). Firstly, the item $\prod_{i=1}^{k}\left(1-p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)$ in Eq. (12) satisfies the following inequality, due to $0\leqslant p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\leqslant1$.

$$\begin{aligned}&1-\sum_{i=1}^{k}\left(p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\\ &\leqslant\prod_{i=1}^{k}\left(1-p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\leqslant1.\end{aligned} \tag{16}$$

Then, we can obtain the following upper and lower boundaries for $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ in Eq. (12):

$$\begin{aligned}&C_{\mathrm{Mul}}\frac{2(\sigma^2-1)}{3}\left(\rho+p_s\left(1+\frac{1}{2^\sigma}-\rho\right)\right)\leqslant\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)\\ &\leqslant C_{\mathrm{Mul}}\frac{2(\sigma^2-1)}{3}\rho.\end{aligned} \tag{17}$$

Evidently, the upper boundary of $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ is consistent with the extreme case of heavy entropy attacks $(p_s=0)$.

5.3. Numeric simulations

The implementation of the modified Gaussian Elimination algorithm without modular inverse operation is built on the top of the OpenSSL-0.9.8g software library and tested on a PIII 1.0G/256 MB Linux machine. The benchmark of the modular multiplication operations over

a finite field $\mathbb{Z}_q$ is $C_{\text{Mul}} = 0.005$ ms, where q is a 256-bit long prime. Fig. 5 shows the normalized computation cost $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ as the function of p_s under the assumption of $12 \leqslant \rho\sigma \leqslant 30$ and $\sigma \geqslant 3$. We consider the following four cases: (a) $\rho\sigma = 12$; (b) $\rho\sigma = 18$; (c) $\rho\sigma = 24$; (d) $\rho\sigma = 30$. Fig. 6 depicts $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ as the function of the parameter σ (the dimension of the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$), where p_s varies from 0.0 to 1.0, under the constraints of $\rho\sigma = 12$, $\rho\sigma = 18$, $\rho\sigma = 24$, and $\rho\sigma = 30$, respectively. From Figs. 5 and 6, we have the following observations:

(1) The normalized computation overhead $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ is slightly increased with the decrease of the parameter p_s, as shown in Fig. 5. Fig. 6 also demonstrates this property. For each case ($\rho\sigma = 12$, $\rho\sigma = 18$, $\rho\sigma = 24$, or $\rho\sigma = 30$), the curve with $p_s = p_1$ is on the top of the one with $p_s = p_2$, where $p_1 < p_2$. However, the vertical distance between different curves with different p_s is very small. This feature shows that the algorithm can efficiently tolerate heavy entropy attacks, since $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ is not sensitive to the increase or decrease of parameter p_s.

(2) The smaller the false positive rate $\delta = f(\rho,\sigma)$ is, the higher the normalized computation cost $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ is. Fig. 6 also clearly shows this property. The curve slope in case of $\rho\sigma = 30$ is the steepest in all four cases, while the curve slope in case of $\rho\sigma = 12$ is the gentlest one. Generally, $12 \leqslant \rho\sigma \leqslant 18$ is a reasonable choice for the S-PSLD algorithm to make an efficient trade-off between the system security and algorithm performance.

(3) The smaller the parameter σ is, the lower the normalized computation cost $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ is. Based on the above observations, it can be seen that the desir-

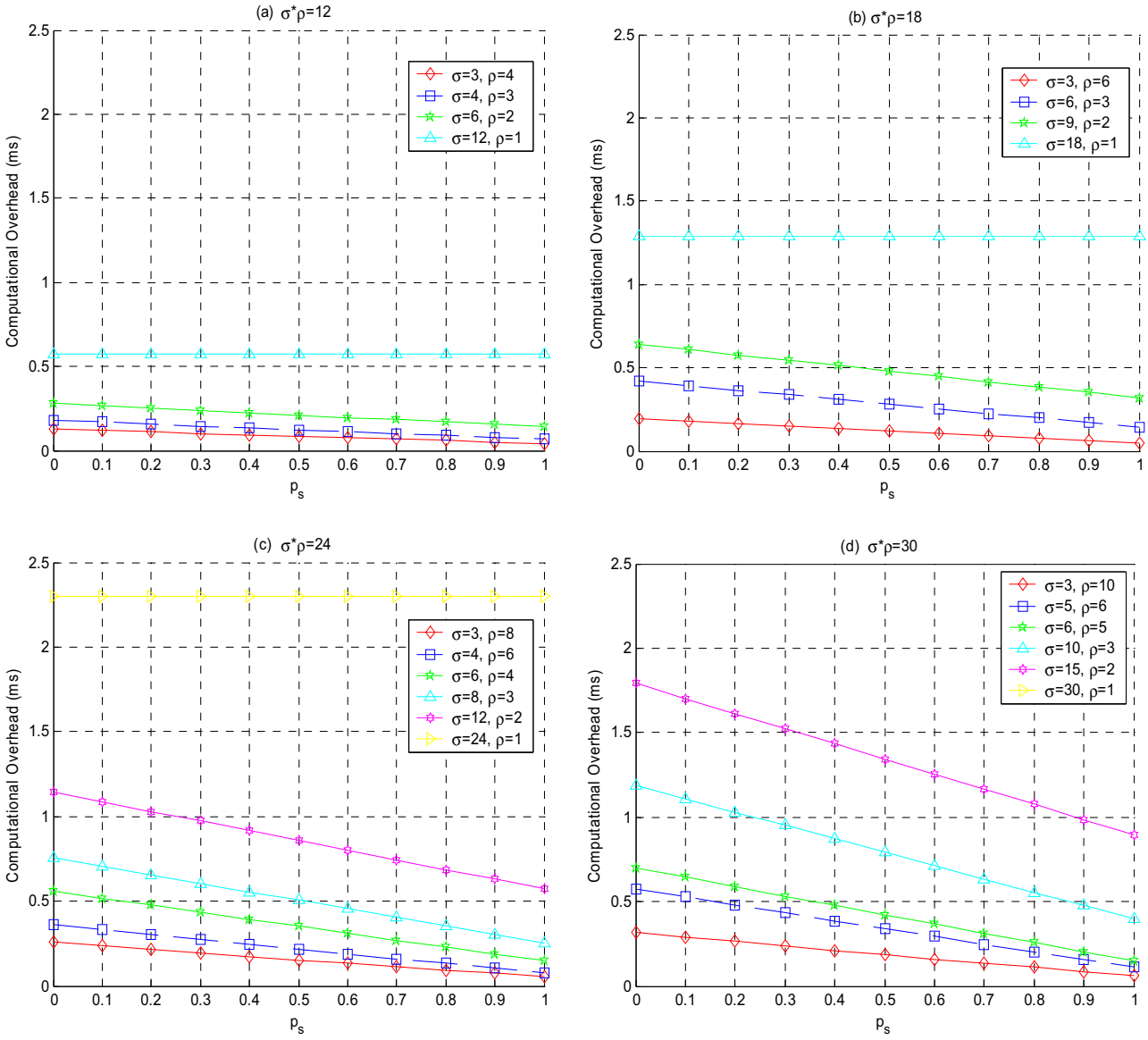


Fig. 5. $\widetilde{C}_{\text{PSLD}}(\rho,\sigma)$ vs. p_s (the probability that the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ is linear independency): (a) $\rho\sigma = 12$; (b) $\rho\sigma = 18$; (c) $\rho\sigma = 24$; (d) $\rho\sigma = 30$.

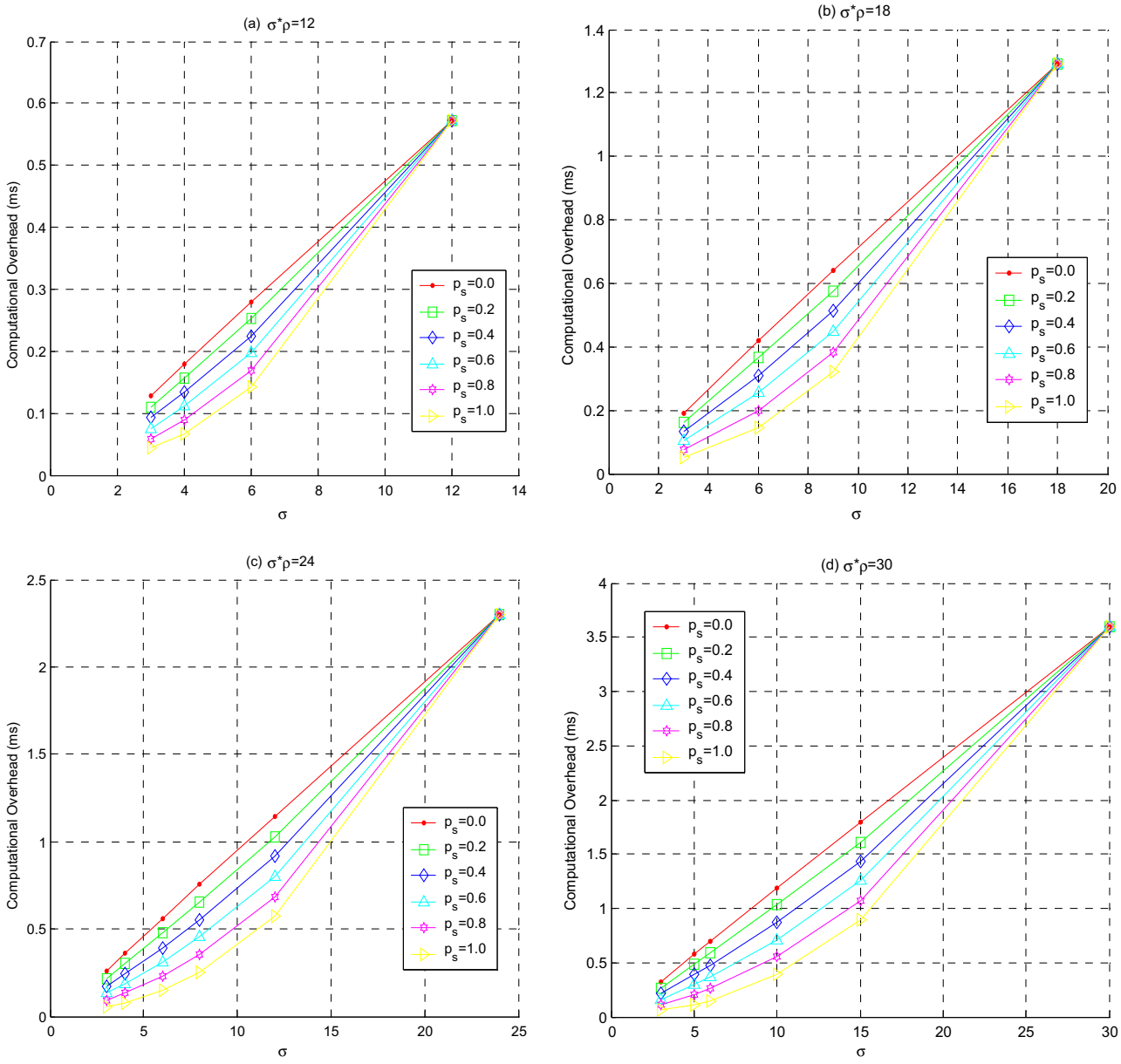


Fig. 6. The normalized computation overhead $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ vs. the system security parameter σ: (a) $\rho\sigma = 12$; (b) $\rho\sigma = 18$; (c) $\rho\sigma = 24$; (d) $\rho\sigma = 30$.

able dimension of the matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma}$ is $3 \leqslant \sigma \leqslant 6$. Thus, for a reasonable false positive rate 2^{-12}–2^{-18} ($12 \leqslant \rho\sigma \leqslant 18$), the pre-defined maximal repeated times in the S-PSLD algorithm can be chosen as $\rho \leqslant 4$.

Therefore, the proposed S-PSLD algorithm is efficient in terms of computation cost, even under serious entropy attacks, due to the low computation complexity.

5.4. Performance comparisons

In this subsection, we show that the average computation complexity of the S-PLSD algorithm is pretty low, compared with the naive Directly Linear-dependent Determination (DLD) algorithm.

Similar to the derivation steps in Eq. (7), the average computation cost of testing each packet, $\widetilde{C}_{DLD}(\sigma)$, can be likewise calculated as

$$\begin{aligned}\widetilde{C}_{\mathrm{DLD}}(\sigma) &= \frac{\sum_{k=1}^{\sigma} 2k(m+n-k)\cdot C_{\mathrm{Mul}}}{\sigma} \\ &= 2\left(\sum_{k=1}^{\sigma}(m+n)k - \sum_{k=1}^{\sigma}k^2\right)C_{\mathrm{Mul}} \\ &= \frac{(\sigma+1)(3(m+n)-(2\sigma+1))}{3}C_{\mathrm{Mul}}. \end{aligned} \tag{18}$$

According to Eq. (12), we further calculate the ratio of $\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)$ to $\widetilde{C}_{\mathrm{DLD}}(\sigma)$ as

Table 1
Computation overhead ratio of the S-PSLD algorithms to the DLD algorithm ($\max(\pi(\widetilde{C}_{\mathrm{DLD}}, \widetilde{C}_{\mathrm{PSLD}}))$).

	$\rho\sigma = 12$ (%)	$\rho\sigma = 18$ (%)	$\rho\sigma = 24$ (%)	$\rho\sigma = 30$ (%)
Parameter $\sigma = 3$	1.4	2.1	2.8	3.5
Parameter $\sigma = 4$	1.6	N/A	3.2	N/A
Parameter $\sigma = 6$	1.8	2.6	3.5	4.4
Parameter $\sigma = 12$	2.0	N/A	3.9	N/A

$$\begin{aligned}\pi(\widetilde{C}_{\mathrm{PSLD}}, \widetilde{C}_{\mathrm{DLD}}) &= \frac{\widetilde{C}_{\mathrm{PSLD}}(\rho,\sigma)}{\widetilde{C}_{\mathrm{DLD}}(\sigma)} \\ &= \frac{2(\sigma-1)\left\{1+\sum_{k=1}^{\rho-1}\left(\prod_{i=1}^{k}\left(1-p_s\left(\frac{1}{2^\sigma}\right)^{i-1}\left(1-\frac{1}{2^\sigma}\right)\right)\right)\right\}}{3(m+n)-(2\sigma+1)}.\end{aligned} \tag{19}$$

Considering the pessimistic case of entropy attacks ($p_s = 0$) as shown in Eq. (13), the ratio $\pi(\widetilde{C}_{\mathrm{PSLD}}, \widetilde{C}_{\mathrm{DLD}})$ can be further reduced to

$$\max\left(\pi(\widetilde{C}_{\mathrm{DLD}}, \widetilde{C}_{\mathrm{PSLD}})\right) = \frac{2(\sigma-1)\rho}{3(m+n)-(2\sigma+1)}. \tag{20}$$

With Eq. (20), it is convenient to compare the performance between the S-PSLD algorithm and DLD algorithm under the extreme case (heavy entropy attacks), where $m+n = 384$ as adopted in scheme [5,13]. Table 1 clearly shows the performance comparisons of the normalized computation complexity in term of different security parameters, σ and ρ. Evidently, even under the heave entropy attacks, the computation cost of the S-PSLS algorithm is far less than that of the DLD algorithm. Eq. (19) further demonstrates that the ratio $\pi(\widetilde{C}_{\mathrm{PSLD}}, \widetilde{C}_{\mathrm{DLD}})$ in the general cases ($0 < p_s \leqslant 1$) should be far less than that in the extreme case, that is, $\pi(\widetilde{C}_{\mathrm{PSLD}}, \widetilde{C}_{\mathrm{DLD}}) \leqslant \max(\pi(\widetilde{C}_{\mathrm{DLD}}, \widetilde{C}_{\mathrm{PSLD}}))$ as shown in Eq. (20).

6. Conclusions

In this paper, based on a novel probabilistic subset linear-dependency test algorithm, we have proposed an efficient self-adaptive packet filtering scheme against entropy attacks in network coding. The algorithm enables that an intermediate node can rapidly detect and filter out the resultant non-innovative packets from the entropy attacks, and at the same time the false positive rate is kept in an expected low level. We have also demonstrated that the proposed algorithm can achieve high efficiency and security in packet filtering even under heavy entropy attacks, and meet the important and emerging requirements for securing network coding. Our future work will focus on designing an efficient strong privacy-preserving scheme for XORing based 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.

Appendix A. Proof of Lemma 1

Proof. As illustrated in Algorithm 1, line 03, the S-PSLD algorithm takes σ received packets $\widetilde{\boldsymbol{Y}}_1, \ldots, \widetilde{\boldsymbol{Y}}_\sigma$ from the buffer of the node v, and constructs a new matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)} = [\widetilde{\boldsymbol{Y}}_1, \ldots, \widetilde{\boldsymbol{Y}}_\sigma]^T$. The random sub-matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma} \subseteq \widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ (Algorithm 1, line 08) is denoted as $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma} = [\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_\sigma]$, where each column vector $\widetilde{\boldsymbol{X}}_k (1 \leqslant k \leqslant \sigma)$ is picked randomly from the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$.

To directly test the linear dependency of the row vectors in the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$, the Gaussian Elimination is performed. Let row vector $\widetilde{\boldsymbol{W}}$ be the bottom row vector in the Echelon form of the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$. Then we can define a vector $\widetilde{\boldsymbol{V}}$ associated with the vector $\widetilde{W}$, where each element $v_i \in \widetilde{\boldsymbol{V}}$ satisfies $v_i = 0$, if $w_i = 0$, where $w_i \in \widetilde{\boldsymbol{W}}$; otherwise $v_i = 1$. Thus, the vector $\widetilde{\boldsymbol{V}}$ can be seen as a series of symbol $\{0,1\}^*$, which has $m+n$ combination forms $\{(0, m+n), (1, m+n-1), \ldots, (m+n, 0)\}$ in term of the number of zero elements in vector $\widetilde{\boldsymbol{V}}$.

Without loss of generality, assume that each zero element is randomly distributed in the space of the row vector $\widetilde{\boldsymbol{V}}$, respectively. For combination form $(k, m+n-k)$, let random variable X be the number of non-zero elements in the vector $\widetilde{\boldsymbol{V}}$, which follows a Binomial distribution

$$P(X=k) = \binom{n}{k}\left(\frac{1}{2}\right)^k\left(1-\frac{1}{2}\right)^{n-k} = \binom{n}{k}\left(\frac{1}{2}\right)^n. \tag{21}$$

If we determine the linear dependency of the matrix $\widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ by testing the subset matrix $\widetilde{\boldsymbol{X}}_{\sigma\times\sigma} \subseteq \widetilde{\boldsymbol{Y}}_{\sigma\times(m+n)}$ using the Gaussian Elimination method, it is possible to generate false positive alarm when $\sigma \leqslant m+n-k$. Thus, the expected false positive rate $\lambda = f(\sigma)$ can be calculated as

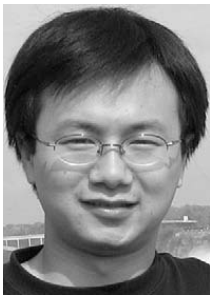
$$\begin{aligned} f(\sigma) &= \sum_{k=0}^{m+n-\sigma}\left\{P(X=k)\left\{\binom{m+n-k}{\sigma}\binom{k}{0}\binom{m+n}{\sigma}^{-1}\right\}\right\} \\ &= \sum_{k=0}^{m+n-\sigma}\left\{\left(\frac{1}{2}\right)^n\binom{m+n}{k}\left\{\binom{m+n-k}{\sigma}\binom{k}{0}\binom{m+n}{\sigma}^{-1}\right\}\right\} \\ &= \left(\frac{1}{2}\right)^{m+n}\sum_{k=0}^{m+n-\sigma}\frac{(m+n-\sigma)!}{k!(m+n-k-\sigma)!} \\ &= \left(\frac{1}{2}\right)^{m+n}\sum_{k=0}^{m+n-\sigma}\binom{m+n-\sigma}{k} = \frac{1}{2^\sigma}. \end{aligned} \tag{22}$$

Therefore, the successful detection rate $s(\sigma)$ can be denoted as $s(\sigma) = 1 - f(\sigma) = 1 - 2^{-\sigma}$. □

References

[1] Y. Zhu, B. Li, J. Guo, Multicast with network coding in application-layer overlay networks, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 1–13.
[2] S. Katti, H. Rahul, D. Katabi, M'edard, J. Crowcroft. Xors in the air: practical wireless network coding, in: Proceedings of ACM SIGCOMM, 2006.
[3] 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.
[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 Transaction 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] 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.
[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 IEEE 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 Information Theory 52 (10) (2006) 4413–4430.
[18] D. Boneh, B. Lynn, H. Shacham, Short signatures from the weil pairing, Journal of Cryptology 17 (4) (2004) 297–319.
[19] J. Camenisch, S. Hohenberger, M. Pedersen, Batch verification of short signatures, in: Proceedings of EUROCRYPT, LNCS, vol. 4514, 2007.
[20] 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.
[21] 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.
[22] 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.
[23] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, D.R. Karger, Byzantine modification detection in multicast networks with random network coding, IEEE Transactions on Information Theory 54 (6) (2008).
[24] Q. Li, D. Chiu, J. Lui, On the practical and security issues of batch content distribution via network coding, in: Proceedings of IEEE ICNP, 2006.
[25] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of ACM SIGCOMM, 2003.
[26] J. Burgess, B. Gallagher, D. Jensen, B. Levine, MaxProp: routing for vehicle-based disruption-tolerant networks, in: Proceedings of IEEE INFOCOM, 2006.

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 and Technology, Tsinghua University, China. In 2005, he was a Visiting Scholar with the Department of Computer Sciences, Hong Kong Baptist University. In 2009, he was a Visiting Scholar with the Department of Computer Science and Engineering, the Chinese University of Hong Kong. He has served as the Technical Program Committee (TPC) member for main network conferences, such as IEEE ICCCN, IEEE GLOBECOM, IEEE ICC, IEEE WCNC, etc. He is a member of IEEE CISTC. His current research interests include network coding, clouding computing, security and privacy in wireless communication and mobile computing. He has received Excellent Backbone Talents Fund Award, Outstanding Doctoral Graduate Award, and Excellent Doctoral Thesis Award of Tsinghua University.



Yanfei Fan received the M.Eng. degree (2005) from Tsinghua University, China, and the B.Eng. degree (2002) from Beijing University of Posts and Telecommunications, China, all in Computer Science. He is currently pursuing his Ph.D. degree in the Department of Electrical and Computer Engineering at University of Waterloo, Canada. His research interests include network coding, security in wireless communication and mobile computing.

Xuemin (Sherman) Shen (IEEE M'97-SM'02-F'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 University Research Chair Professor, 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 networks, wireless network security, wireless body area networks and vehicular ad hoc and sensor networks. He is a co-author of three books, and has published more than 400 papers and book chapters in wireless communications and networks, control and filtering. He is a Distinguished Lecturer of IEEE Communications Society. He serves as the Tutorial Chair for IEEE ICC'08, the Technical Program Committee Chair for IEEE Globecom'07, the 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, IEEE Communications Magazine, and ACM Mobile Networks and Applications, etc. He 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 and 2007 from the Faculty of Engineering, University of Waterloo. He 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.