This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Padding for Orthogonality: Efficient Subspace Authentication for Network Coding

Peng Zhang*, Yixin Jiang*, Chuang Lin*, Hongyi Yao†, Albert Wasef‡ and Xuemin (Sherman) Shen‡
*Tsinghua National Laboratory for Information Science and Technology,
Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
†Dept. of Electrical Engineering and Computer Science, California Institute of Technology, USA
‡Dept. of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada

***Abstract*—Network coding provides a promising alternative to traditional store-and-forward transmission paradigm. However, due to its information-mixing nature, network coding is notoriously susceptible to *pollution attacks*: a single polluted packet can end up corrupting bunches of good ones. Existing authentication mechanisms either incur high computation/bandwidth overheads, or cannot resist the *tag pollution* proposed recently. This paper presents a novel idea termed *"padding for orthogonality"* for network coding authentication. Inspired by it, we design a public-key based signature scheme and a symmetric-key based MAC scheme, which can both effectively contain pollution attacks at forwarders. In particular, we combine them to propose a unified scheme termed *MacSig*, the first hybrid-key cryptographic approach to network coding authentication. It can thwart both normal pollution and tag pollution attacks in an efficient way. Simulative results show that our MacSig scheme has a low bandwidth overhead, and a verification process 2-4 times faster than typical signature-based solutions in some circumstances.**

I. Introduction

Network coding provides a new data transmission paradigm, in which intermediate nodes are allowed to code/mix packets rather than just forward them. This information-mixing based technique is proven capable of achieving maximized throughput [1], enhanced robustness [2], and lower energy consumption [3] for communication networks. Specially, random network coding [4] is verified to have all the above features, and can be efficiently deployed in a distributed way. Due to its nice features, random network coding has already found applications in content distribution networks [5], P2P streaming networks [6], wireless mesh networks [7], etc.

However, the information-mixing nature of network coding also renders it more susceptible to *pollution attacks* than traditional store-and-forward paradigm. Consider a scenario in which a commercial data center is distributing a file to a set of costumers via a network coded P2P network. An adversary pretends as a normal customer, by downloading and contributing packets of the file. In this process, it generates corrupted packets and contributes them to its peers. After being coded with other packets, a single corrupted packet can result in tens or even hundreds of polluted ones. This may cause legitimate users unable to download the file properly.

Existing schemes include information-theoretic schemes [8], [9], and cryptography-based schemes [10]–[18]. For information-theoretic schemes, they can only passively tolerate pollution at sinks, but not actively prevent them. On the other hand, cryptography-based schemes enable forwarders to verify the integrity of their received packets, so that corrupted packets can be discarded before polluting good ones. This paper only considers the cryptography-based schemes, which can be further grouped into two classes. The first class includes schemes [10]–[14] that are built on public-key based techniques, such as homomorphic hash, homomorphic signature, etc. These schemes are provably secure under the hardness assumptions of well-known cryptographic problems, but will incur high computation overhead at forwarders. The second class are schemes [15]–[18] which involve symmetric-key encryptions that are computationally efficient. The main disadvantages of schemes in this class include that they incur a larger bandwidth overhead and must carefully manage the keys.

This paper approaches the problem of network coding authentication using a novel idea called *"padding for orthogonality"*: the source pads each packet with an extra symbol, so that the subspace spanned by these padded packets is orthogonal to a specific vector; forwarders check the integrity of a received packet by verifying whether it maps this vector to zero. Based on this idea, we propose a public-key based scheme and prove its security under the hardness assumption of discrete logarithm problem. In addition, we also propose a symmetric-key based scheme that is secure against a coalition of c adversaries. Most importantly, we carefully combine them to propose *MacSig* — the first hybrid-key based approach to network coding authentication.

Our MacSig scheme offers the following primary features: (1) **Security against Pollution.** It can effectively not only thwart normal pollution attacks, but also resist *tag pollution* presented in [17]. (2) **Bandwidth Efficiency.** It requires a smaller number of tags for each packet compared with [16] (which uses the key distribution scheme given in [19]). (3) **Computation Efficiency.** It needs a moderate/small number of symmetric-key/public-key cryptographic operations. Simulations show that its verification process is 2-4 times faster than typical signature-based schemes [12]–[14] in some circumstances.

The rest of this paper is organized as follows. Section II gives a formal statement of the problem to be studied. Section III presents our basic idea, and introduces two authentication schemes based on it. Section IV proposes a hybrid-key authentication scheme, whose performance is evaluated in Section

V. Section VI discusses how our schemes can be adapted to function in a more general case. Section VII surveys some related work, and Section VIII concludes.

II. Problem Statement

A. Network Model

We consider a typical multicast scenario, in which a source S needs to deliver a series of packets $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \ldots, \underline{\boldsymbol{x}}_m$ to multiple receivers $\{R_i\}$. Each packet $\underline{\boldsymbol{x}}_i$ is represented as a vector $(\underline{x}_{i,1}, \underline{x}_{i,2}, \ldots, \underline{x}_{i,n})$ of finite field $\mathbb{F}_p^n$, where p is a prime.

For each $\underline{\boldsymbol{x}}_i$, the source S generates an augmented packet $\boldsymbol{x}_i$ by prefixing $\underline{\boldsymbol{x}}_i$ with the i^{th} unit vector of dimension m:

$$\boldsymbol{x}_i = (\overbrace{\underbrace{0, \cdots, 0}_{i-1}, 1, 0, \ldots, 0}^{m}, \underline{x}_{i,1}, \underline{x}_{i,2}, \cdots, \underline{x}_{i,n}) \quad (1)$$

Let V denote the subspace spanned by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$, and term $\boldsymbol{x}_i$ as the i^{th} basis vector of V. Then S sends vectors in V and the network is responsible for replicating of V at each receiver R_i, who can derive $\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_m$ by computing the m basis vectors of V via Guassian eliminations.

Specifically, for random network coding the source sends linear combinations of packets using randomly selected coefficients. For example, linearly combining packets $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l$ using coefficients $\alpha_1, \alpha_2, \ldots, \alpha_l$ results in

$$\boldsymbol{y} = \sum_i^l \alpha_i \boldsymbol{x}_i = (\sum_i^l \alpha_i x_{i,1}, \ldots, \sum_i^l \alpha_i x_{i,m+n}) \quad (2)$$

The first m symbols of $\boldsymbol{y}$ are termed as its *coding coefficients*. Intermediate nodes linearly combines their received packets for output in a similar way. Then a receiver R_i can recover V exactly after receiving m linearly independent packets. In fact, any m received packets are linearly independent with a high probability given the filed size p is sufficiently large [4].

In this paper, we consider a more realistic setting, in which the data D to be sent consists of more than m packets. Using the technique introduced in [20], S should first break D into multiple *generations*:

$$D = [\underbrace{\underline{\boldsymbol{x}}_1, \ldots, \underline{\boldsymbol{x}}_h}_{G_1}, \ldots, \underbrace{\underline{\boldsymbol{x}}_{(n-1)h+1}, \ldots, \underline{\boldsymbol{x}}_{nh}}_{G_n}, \ldots] \quad (3)$$

Then S sends D as a stream of generations, with network coding only performed among packets belonging to the same generation.

B. Adversary Model

The adversary is aimed at injecting a small number of corrupted packets into the network to cause a large scale of pollution. To achieve this goal, he strives to collect legal packets and forge corrupted ones that can pass the verification of other innocent nodes. Without loss of generality, we assume the source is always trusted, but the relay nodes can be compromised. By compromise, we mean that the adversary can read the memory, monitor the input, and control the output of a compromised node. In this paper, we allow the adversary to compromise a coalition of nodes to launch more effective attacks. Finally, we assume that the adversaries are aware of our authentication scheme, but are bounded in computation power, and can only perform polynomial-time algorithms.

III. Homomorphic Subspace Authentication

In this section, we first introduce the basic idea of *"padding for orthogonality"*, and then propose two different schemes for network coding authentication: the *Homomorphic Subspace Signature (HSS)*, and the *Homomorphic Subspace MAC (HSM)*.

A. Basic Idea Overview

Noted from our network model, although packets undergo rounds of coding processes at forwarders, the linear subspace V spanned by them stays constant. We can check the integrity of a packet $\boldsymbol{w}$ by verifying whether $\boldsymbol{w} \in V$. Based on this observation, we can characterize V using a vector $\boldsymbol{v}$ randomly chosen from its orthogonal subspace, and let forwarders check whether $\boldsymbol{w} \cdot \boldsymbol{v}^T = 0$. This approach catches an essential property of network coding — the invariance of linear subspace, and inspires some appealing schemes [13], [14], [18]. However, it still has two practical problems unsolved: (1) For different generations, the source should calculate different $\boldsymbol{v}$'s, and distribute them prior to transmission, which can cause a high startup latency. (2) These $\boldsymbol{v}$'s should also be authenticated, meaning that an extra secure channel is required.

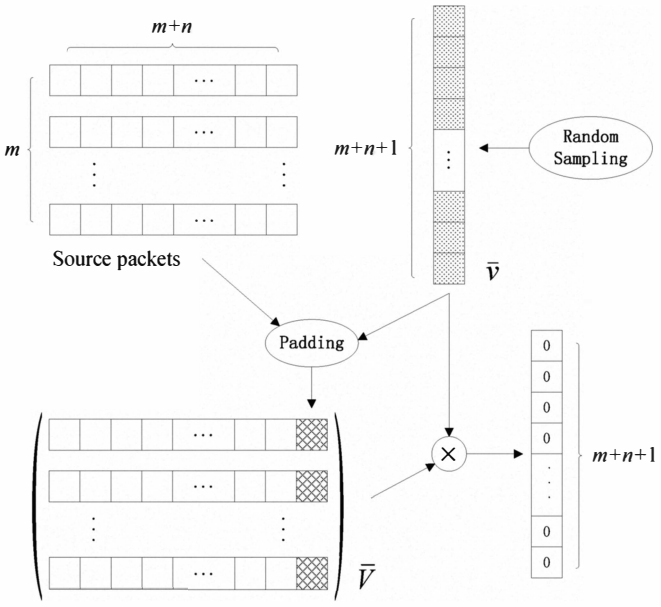

Fig. 1: The idea of "padding for orthogonality"

Now, we present a novel approach called *"padding for orthogonality"* to overcome these two problems. By this approach, the source randomly samples a vector $\bar{\boldsymbol{v}}$ of length $m+n+1$ at the bootstrap stage, and for every generation, the source pads each packet with an extra symbol/tag, so that its

inner product with $\bar{\boldsymbol{v}}$ equals zero, as shown in Fig. 1. Then the subspace $\bar{V}$ spanned by these augmented packets is orthogonal to $\bar{\boldsymbol{v}}$. To verify a packet $\boldsymbol{w}$, a relay node just checks whether $\boldsymbol{w} \cdot \bar{\boldsymbol{v}}^T = 0$. Clearly, using this approach, we don't have to pre-distribute $\bar{\boldsymbol{v}}$'s per generation using a secure channel; we just need m tags which cause no startup latency.

To make this approach function in presence of Byzantine adversaries, who attempt to forge tags for illegal packets, we consider the following two techniques. First, we can let the source keep $\bar{\boldsymbol{v}}$ as a secret key, and based on it generate a public key which can be used by relay nodes to verify the integrity of packets. If it is sufficiently hard to derive $\bar{\boldsymbol{v}}$ based on the public key, then relay nodes cannot successfully forge tags for any illegal packets. For the second solution, we let the source keep a pool $\mathcal{P}$ of $\bar{\boldsymbol{v}}$'s, and pad each packet with multiple tags generated according to these $\bar{\boldsymbol{v}}$'s; each relay node is assigned with a subset of $\mathcal{P}$, and can only verify a packet against part of its tags. If these $\bar{\boldsymbol{v}}$'s are distributed properly, a corrupted packet generated by a malicious node will fail the verifications of other nodes with high probability. The following *Homomorphic Subspace Signature (HSS)* and *Homomorphic Subspace MAC (HSM)* scheme are designed using these two techniques, respectively. For simplicity of introduction, we assume that the transmission consists of only one generation. We will discuss the multiple-generation cases later in Section VI.

B. The Homomorphic Subspace Signature

The Model. A *Homomorphic Subspace Signature (HSS)* is defined as a tuple of four probabilistic polynomial-time (PPT) algorithms (**Setup**,**Sign**,**Combine**,**Verify**):

- **Setup.** Input: 1^k, the security parameter, and N, the length of vectors to be signed. Output: a prime number q, a secret key K_s, and a public key K_p.
- **Sign.** Input: a vector $\boldsymbol{x} \in \mathbb{F}_q^N$, and the secret key K_s. Output: a vector $\bar{\boldsymbol{x}} = (\boldsymbol{x}, \sigma)$, where $\sigma \in \mathbb{F}_q$ is termed as the signature of $\boldsymbol{x}$.
- **Combine.** Input: l vectors $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l$, where $\bar{\boldsymbol{x}}_i = (\boldsymbol{x}_i \in \mathbb{F}_q^N, \sigma_i \in \mathbb{F}_q)$, and l coefficients $\alpha_1, \ldots, \alpha_l$, where $\alpha_i \in \mathbb{F}_q$. Output: a vector $\bar{\boldsymbol{x}} = (\sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i, \sigma \in \mathbb{F}_q)$.
- **Verify.** Input: a vector $\bar{\boldsymbol{x}} = (\boldsymbol{x} \in \mathbb{F}_q^N, \sigma \in \mathbb{F}_q)$, and the public key K_p. Output: either 1 (accept), or 0 (reject).

An HSS is said to be *correct* if the following two conditions are satisfied:

(1) $\textbf{Verify}(\textbf{Sign}(\boldsymbol{x}, K_s), K_p) = 1$ and

(2) $\textbf{Verify}(\bar{\boldsymbol{x}}_i, K_p) = 1$ for $i = 1, \ldots, l \Rightarrow$

$\textbf{Verify}(\textbf{Combine}(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l; \alpha_1, \ldots, \alpha_l), K_p) = 1$

An HSS is said to be *secure* if for any PPT adversary A, the probability that A wins the security game **HSS-GAME** defined below is negligible in the security parameter k:

- **Setup.** The adversary A specifies parameters 1^k and N. The challenger C runs $\textbf{Setup}(1^k, N)$ to generate q, K_s and K_p, of which it sends q and K_p to A.
- **Query.** A adaptively submits vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ to C, who runs **Sign** for these vectors and sends the corresponding $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_m$ to A.
- **Forge.** A generates a vector $\bar{\boldsymbol{y}} = (\boldsymbol{y} \in \mathbb{F}_q^N, \sigma \in \mathbb{F}_q)$ with $\boldsymbol{y} \notin span(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$. If $\textbf{Verify}(\bar{\boldsymbol{y}}, K_p) = 1$, then A wins; otherwise A loses.

Remarks. When applying the above HSS scheme to the network coding model given in Section II-A, we can just let $N = m + n$ and $q = p$.

The Construction. Based on the above model, we give our construction of HSS.

- **Setup.** Given 1^k and N, perform the following steps: (1) choose a prime number $q > 2^k$; (2) find a multiplicative cyclic group $\mathbb{G}$ of order q, and select a generator g for $\mathbb{G}$; (3) set $\boldsymbol{\beta} \xleftarrow{R} \mathbb{F}_q^N \mathbb{F}_q^*$, and calculate $\boldsymbol{h} = (g^{\beta_1}, \ldots, g^{\beta_{N+1}})$. Output q, $K_s = \boldsymbol{\beta}$, and $K_p = \boldsymbol{h}$.
- **Sign.** Given $\boldsymbol{x} \in \mathbb{F}_q^N$ and $\boldsymbol{\beta}$, calculate the signature $\sigma = -(\sum_{i=1}^{N} \beta_i x_i)/\beta_{N+1}$. Output $\bar{\boldsymbol{x}} = (\boldsymbol{x}, \sigma)$.
- **Combine.** Given $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l$, where $\bar{\boldsymbol{x}}_i \in \mathbb{F}_q^{N+1}$, and $\alpha_1, \ldots, \alpha_l$, where $\alpha_i \in \mathbb{F}_q$. Output $\bar{\boldsymbol{x}} = \sum_{i=1}^{l} \alpha_i \bar{\boldsymbol{x}}_i$.
- **Verify.** Given $\bar{\boldsymbol{x}} \in \mathbb{F}_q^{N+1}$ and $\boldsymbol{h}$, calculate $\delta = \boldsymbol{h}^{\bar{\boldsymbol{x}}} \triangleq \prod_{i=1}^{N+1} h_i^{\bar{x}_i}$. Output 1 if $\delta = 1$, or 0 otherwise.

Theorem 1. Our construction of HSS is correct.

Proof: Let $\bar{\boldsymbol{x}} = \textbf{Sign}(\boldsymbol{x}, \boldsymbol{\beta})$, then it is easy to verify that $\bar{\boldsymbol{x}} \cdot \boldsymbol{\beta}^T = \sum_{i=1}^{N+1} \bar{x}_i \beta_i = 0$, and then $\delta = \boldsymbol{h}^{\bar{\boldsymbol{x}}} = g^{\bar{\boldsymbol{x}} \cdot \boldsymbol{\beta}^T} = 1$. Thus $\textbf{Verify}(\bar{\boldsymbol{x}}, \boldsymbol{h}) = 1$, and Condition (1) holds. Similarly, it is easy to verify that any vector $\bar{\boldsymbol{x}}$ which passes the verification must satisfy $\bar{\boldsymbol{x}} \cdot \boldsymbol{\beta}^T = 0$. Therefore, if we assume $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l$ pass the verification, then $\boldsymbol{\beta}$ is orthogonal to the subspace $\bar{V} = span(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l)$. By definition, $\bar{\boldsymbol{y}} = \textbf{Combine}(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l; \alpha_1, \ldots, \alpha_l) \in \bar{V}$, then $\bar{\boldsymbol{y}} \cdot \boldsymbol{\beta} = 0$, and $\textbf{Verify}(\bar{\boldsymbol{y}}, \boldsymbol{h}) = 1$. Thus Condition (2) also holds. ■

Theorem 2. Our construction of HSS is secure.

Proof: Suppose A wins the security game with some $\bar{\boldsymbol{y}} = (\boldsymbol{y}, \sigma)$. Since $\boldsymbol{y} \notin span(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, it immediately follows that $\bar{\boldsymbol{y}} \notin span(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_m)$. In addition, we have $\boldsymbol{h}^{\bar{\boldsymbol{y}}} = 1$ and $\boldsymbol{h}^{\bar{\boldsymbol{x}}_i} = 1$, $i = 1, \ldots, m$. By employing the techniques given in Section 3.2 of [21], A can also solve the discrete logarithm problem over $\mathbb{G}$ with a probability at least $1 - 1/q$. Note that for any PPT algorithm, the probability that it solves the discrete logarithm problem over a cyclic group of order $q = 2^k$ is negligible in k [22]. Thus, for any PPT adversary A, the probability that it wins **HSS-GAME** is also negligible in k. ■

C. The Homomorphic Subspace MAC

The Model. Similar to HSS, a *Homomorphic Subspace MAC (HSM)* is defined as a tuple of four probabilistic polynomial-time (PPT) algorithms (**Setup**,**MAC**,**Combine**,**Verify**):

- **Setup.** Input: 1^k, the security parameter, and N, the length of vectors to be authenticated. Output: a prime number q, a set K consisting of r MAC keys.

- **MAC.** Input: a vector $\boldsymbol{x} \in \mathbb{F}_q^N$, and the key set K. Output: a vector $\bar{\boldsymbol{x}} = (\boldsymbol{x}, t_1, \ldots, t_r)$, where $t_i \in \mathbb{F}_q$ is a MAC of $\boldsymbol{x}$ calculated using the i^{th} MAC key.
- **Combine.** Input: l vectors $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l$, where $\bar{\boldsymbol{x}}_i = (\boldsymbol{x}_i \in \mathbb{F}_q^N, T_i \in \mathbb{F}_q^r)$, and l coefficients $\alpha_1, \ldots, \alpha_l$, where $\alpha_i \in \mathbb{F}_q$. Output: a vector $\bar{\boldsymbol{x}} = (\sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i, T \in \mathbb{F}_q^r)$.
- **Verify.** Input: a vector $\bar{\boldsymbol{x}} = (\boldsymbol{x}_i \in \mathbb{F}_q^N, T \in \mathbb{F}_q^r)$, and a key set $K' \subset K$. Output: either 1 (accept), or 0 (reject).

An HSM is said to be *correct* if the following two conditions are satisfied:

(1) **Verify**(**MAC**$(\boldsymbol{x}, K), K) = 1$ and

(2) **Verify**$(\bar{\boldsymbol{x}}_i, K) = 1$ for $i = 1, \ldots, l \Rightarrow$

Verify(**Combine**$(\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l; \alpha_1, \ldots, \alpha_l), K) = 1$

An HSM is said to be *secure* if for any PPT adversary A, the probability that A wins the security game **HSM-GAME** defined below is no greater than $1/q^d$:

- **Setup.** The adversary A specifies parameters 1^k and N. The challenger C runs **Setup**$(1^k, N)$ to generates q and K. Then it randomly selects two key sets $K' \subset K$ and $K'' \subset K$, with $|K'' \backslash K'| = d$, and sends K' to A.
- **Query.** A adaptively submits vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ to C, who runs **MAC** for these vectors, and sends to A the MACs $T_1, T_2, \ldots, T_m$, where $T_i = \{t_{i,1}, \ldots, t_{i,r}\}$.
- **Forge.** A chooses a vector $\boldsymbol{y} \notin span(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$. Then for each $i = 1, \ldots, r$, it calculates the MAC t_i if $k_i \in K'$, or randomly forges the MAC t_i if $k_i \notin K'$. If **Verify**$((\boldsymbol{y}, t_1, \ldots, t_r), K'') = 1$, then A wins; otherwise A loses.

Remarks. Different from homomorphic subspace signatures, an HSM uses symmetric keys, i.e., MAC keys, for authentication. The advantage is that forwarders can perform the **Verify** procedure much more efficiently. However, an adversary can also easily forge MACs for illegal packets if all MAC keys are publicized. Thus, we require that the source hold a set K of MAC keys, and each relay node be assigned with a random subset of the K. In this way, each forwarder can only forge some MACs correctly for an illegal packet. If the receiver of this illegal packet has some MAC keys that the adversary does not have, then it can successfully detect the forgery. **HSM-GAME** characterizes this security requirement, by simulating a scenario in which a malicious node with key set K' attempts to forge MACs that pass the verification of another node with key set K''.

The Construction. Based on the above model, we give our construction of HSM.

- **Setup.** Given 1^k and N, choose a prime number $q > 2^k$, and set $\boldsymbol{\gamma}_i = (\gamma_{i,1}, \ldots, \gamma_{i,N+1}) \xleftarrow{R} \mathbb{F}_q^N \mathbb{F}_q^*$ for each $i = 1, \ldots, r$. Output $K = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_r)$.
- **MAC.** Given $\boldsymbol{x} \in \mathbb{F}_q^N$ and K, calculate a tag $t_i = -(\sum_{j=1}^{N} \gamma_{i,j} x_j)/\gamma_{i,N+1}$ for each $i = 1, \ldots, r$. Output $\bar{\boldsymbol{x}} = (\boldsymbol{x}, t_1, \ldots, t_r)$.
- **Combine.** Given $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_l$, where $\bar{\boldsymbol{x}}_i \in \mathbb{F}_q^{N+r}$, and $\alpha_1, \ldots, \alpha_l$, where $\alpha_i \in \mathbb{F}_q$, output $\bar{\boldsymbol{x}} = \sum_{i=1}^{l} \alpha_i \bar{\boldsymbol{x}}_i$.
- **Verify.** Given $\bar{\boldsymbol{x}} \in \mathbb{F}_q^{N+r}$ and $K' \subset K$, calculate $\xi_i = \sum_{j=1}^{N} \gamma_{i,j} \bar{x}_j + \gamma_{i,N+1} \bar{x}_{N+i}$, for each $\gamma_i \in K'$. Output 1 if all $\xi_i = 0$, or 0 otherwise.

Theorem 3. Our construction of HSM is correct.

Proof: For $r = 1$, the proof is much similar to that of Theorem 1, and it is easy to extend the proof to cases of $r > 1$. We omit the details here due to the limit of space. ■

Theorem 4. Our construction of HSM is secure.

Proof: For each $i = 1, \ldots, r$, we consider the following three cases: (1) $\boldsymbol{\gamma_i} \in K'$. A can accurately calculate the MAC t_i which evaluates ξ_i to zero. (2) $\boldsymbol{\gamma_i} \notin K'$ and $\boldsymbol{\gamma_i} \notin K''$. Any $t_i \in \mathbb{F}_q$ is valid since it will not be checked. (3) $\boldsymbol{\gamma_i} \notin K'$ but $\boldsymbol{\gamma_i} \in K''$. After the **Query** step, A can get the following group of equations regarding $\boldsymbol{\gamma_i}$:

$$\begin{pmatrix} \boldsymbol{x}_1, & t_{i,1} \\ \boldsymbol{x}_2, & t_{i,2} \\ \vdots & \vdots \\ \boldsymbol{x}_m, & t_{i,m} \end{pmatrix} \cdot \boldsymbol{\gamma}_i^T = \mathbf{0} \tag{4}$$

which has p^{N+1-R} solutions for $\boldsymbol{\gamma}_i$, where R is row rank of the coefficient matrix. Suppose we insert into this group of equations with:

$$(\boldsymbol{y}, t) \cdot \boldsymbol{\gamma}_i^T = 0 \tag{5}$$

where $\boldsymbol{y} \notin span(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, $t \in \mathbb{F}_q$. Then the row rank of the coefficient matrix will be $R+1$, the solution set for $\boldsymbol{\gamma}_i$ will have cardinality q^{N-R}. This means that $q^{N-R}/q^{N+1-R} = 1/q$ of solutions to Eq. (4) can solve Eq. (5). As we assume that the MAC key $\boldsymbol{\gamma}_i$ is sampled randomly from $\mathbb{F}_q^{N+1}$, the probability any t is a valid MAC that evaluates ξ_i to zero is exactly $1/q$. Since there are totally d such i satisfying that $\boldsymbol{\gamma_i} \notin K'$ and $\boldsymbol{\gamma_i} \in K''$, the probability of $\xi_i = 0$ for all these i is then $1/q^d$. ■

D. MAC Key Distribution for HSM

Recall in our construction of HSM, the probability that any packet polluted by a node A can pass the verification of another node B is bounded by $1/q^d$, where d is the number of MAC keys held by B but not by A. This implies that the security level of HSM depends on how MAC keys are distributed among nodes in the network. In this subsection, we first formalize this problem of MAC key distribution, and then introduce our proposed scheme. We assume a strong adversary model, in which a set of compromised nodes can collude to launch pollution attacks.

The Problem. Let Ω denote the set of all nodes except the source S, with $|\Omega| = N$. Let K be the set of all MAC keys held by S. To each node $\omega_i \in \Omega$, S assigns a subset $K(\omega_i) \subset K$ of MAC keys. For a set $A \subset \Omega$, define its keys as $K(A) \triangleq \bigcup_{\omega_i \in A} K(\omega_i)$. We say a key $k \in K$ is *safe* with respect to a node ω_i and a set A, if $k \in K(\omega_i) \backslash K(A)$. We say the key distribution scheme is *c-secure* if for any $\omega_i \in \Omega$ and $A \subset \Omega$ with $|A| \leq c$, there is at least one safe key.

Canetti et al. [19] introduce a probabilistic key distribution mechanism, in which every node $\omega_i \in \Omega$ is assigned with any

key $k \in K$ with an equal probability P_a. They show that by letting $|K| = e(c+1)\ln\frac{1}{\epsilon}$ and $P_a = \frac{1}{c+1}$, the probability that there is at least one safe key for a randomly chosen ω_i and A with $|A| = c$ can be made higher than $1-\epsilon$. However, to achieve this for any ω_i and A with $|A| = c$, i.e., to make the distribution mechanism c-secure with probability at least $1-\epsilon$, $|K|$ should be made no less than $e(c+1)^2 \ln N$. This means that each packet should carry $e(c+1)^2 \ln N$ MACs, which clearly does not scale when the network size N is large. To overcome this limitation, we propose a new approach using which the number of MACs per packet has no relation with N.

Double-Random Key Distribution. Our proposed scheme, termed as *Double-Random Key Distribution*, gets its name because MAC keys are distributed via two random procedures: the first procedure assigns each node with a random set of keys, just like in [19]; the second one randomly selects keys to be used for MAC calculations. More specifically, in the second procedure, the source randomly selects a subset of l MAC keys from K for each generation. Then the source calculates l MACs using these keys for each packet. In addition, to inform forwarders of the selected keys, the source attaches the indexes of these l keys to each packet.

The rationale of our proposed scheme is to introduce randomness when generating MACs at the source. This randomness can prevent the adversary from knowing the keys used for MAC calculation before hand, and hence prevent the adversary from electively compromising nodes. Theorem 5 shows that the number of MACs per packet has no relation with the N, meaning that the bandwidth overhead is scalable with the network size.

Theorem 5. Let the number of secret keys be $|K| = e(c+1)m$, with $m = \frac{2}{\delta^2}(\gamma+1)(c+1)\ln N$, $\gamma > 0$ and $0 < \delta < 1$. Let the number of MACs per packet be $l = \frac{1}{1-\delta}e(c+1)\ln\frac{1}{\epsilon}$, and the key assigning probability be $P_a = \frac{1}{c+1}$. Then the probability that the double-random key distribution is c-secure is no smaller than $1-\epsilon$, when $\gamma \to \infty$.

Proof: See the Appendix. ■

IV. The MacSig Authentication Scheme

As shown in the previous section, the HSS scheme is proven secure under the hardness assumption of discrete logarithm problem, and it incurs a lower bandwidth overhead than HSM. However, in most cases, especially when computation power is constrained (e.g., WSN), we prefer the HSM scheme, since it puts less burden on forwarders. However, the recent work [17] reports that such homomorphic MAC based schemes (including our HSM) may suffer from *tag pollution*. Fortunately, we observe that our HSS scheme can be utilized to help HSM thwart this attack. In this section, we first give preliminary to the problem of tag pollution, and then propose a novel scheme termed *MacSig* to solve it.

A. Preliminary to Tag Pollution

By tag pollution, an adversary aims to modify the tags (MACs for HSM) carried by packets rather than the contents of them. If a receiver of a packet with polluted tags does not have necessary keys to check at least one of them, it cannot detect and filter out this tag-polluted packet. It is possible that a packet with polluted tags travels multiple hops until it is finally detected and discarded, which can result in a waste of network bandwidth.

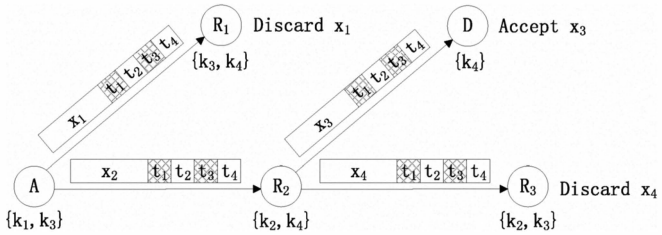

Fig. 2: An example of tag pollution in HSM

For a concrete understanding, consider the example given by Fig. 2, in which there are one adversary A, one receiver D, and some relay routers. Each packet is attached with four MACs. A pollutes two MACs t_1 and t_3 in both its output packets x_1 and x_2. As R_1 can verify the packet x_1 against its MAC t_3 using k_3, it can detect the tag pollution and discard x_1. However, since R_2 has neither k_1 nor k_3, x_2 will pass the verification of R_2 and encode into another two packets x_3 and x_4. As D just extracts the content of x_3, it is not affected by the polluted MACs. On the other hand, packet x_4 will be discarded at node R_3. As a result, the bandwidth of link $R_2 \to R_3$ is wasted. For a worse case in which a tag-polluted packet can travel more hops and infect more packets before being detected, the bandwidth waste will be considerable.

B. MacSig: The Proposed Scheme

We propose a novel scheme termed *MacSig*, which uses both homomorphic MACs and signatures for packet authentication. The basic idea is shown in Fig. 3(a), where the packet content is authenticated by homomorphic MACs, and these MACs are further authenticated by a homomorphic signature. In real implementation of SigMac, we also let the signature authenticate part of the packet content (reasons to be given in Section IV-C). Specifically, we let the signature authenticate both the MACs and coding coefficients of the packet, as shown in Fig. 3(b).

In the following, we give the details of our MacSig scheme. For convenience of demonstration, we first assume the key distribution scheme in [19], and then show how our double-random approach can be used in MacSig. Similar to the construction of HSS and HSM introduced in Section III, our MacSig consists of four probabilistic polynomial-time (PPT) algorithms.

Setup: The source performs the following five steps: (1) find a multiplicative cyclic group $\mathbb{G}$ of order p, and select a generator g for $\mathbb{G}$. (2) sample the secret key $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{m+l+1}) \xleftarrow{R} \mathbb{F}_p^{m+l}\mathbb{F}_p^*$. (3) compute the public key $\boldsymbol{h} = g^{\boldsymbol{\beta}} \triangleq (g^{\beta_1}, \ldots, g^{\beta_{m+l+1}})$. (4) sample a pool of MAC keys $K = \{\boldsymbol{\gamma}_i\}_{i=1}^{l}$, where $\boldsymbol{\gamma}_i = (\gamma_{i,1}, \ldots, \gamma_{i,m+n+1}) \xleftarrow{R} \mathbb{F}_p^{m+n}\mathbb{F}_p^*$.

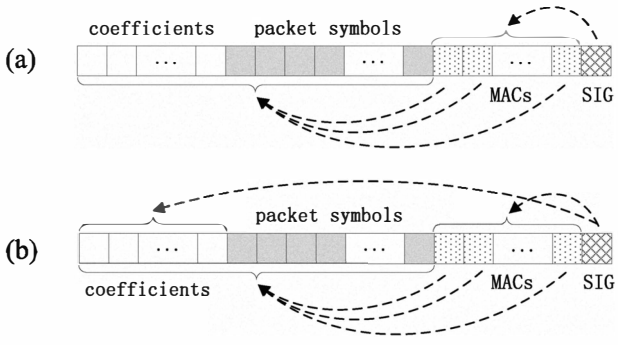

Fig. 3: Basic idea of the MacSig authentication scheme.

(5) for each MAC key, assign it to each node with an equal probability P_a.

MAC+Sign: The source performs the following three steps: (1) for the i^{th} source packet $\boldsymbol{x}_i$, attach l MACs $\{t_{i,1}, \ldots, t_{i,l}\}$, where $t_{i,j}$ is calculated as:

$$t_{i,j} = \frac{-\sum_{r=1}^{m+n} \gamma_{j,r} x_{i,r}}{\gamma_{j,m+n+1}} \tag{6}$$

(2) attach the signature σ_i to $\boldsymbol{x}_i$, where σ_i is calculated as:

$$\sigma_i = \frac{-\sum_{j=1}^{m} \beta_j x_{i,j} + \sum_{j=1}^{l} \beta_{j+m} t_{i,j}}{\beta_{m+l+1}} \tag{7}$$

(3) output the resultant packet $\bar{\boldsymbol{x}}_i = (\boldsymbol{x}_i, t_{i,1}, \ldots, t_{i,l}, \sigma_i)$, which has length $m+n+l+1$.

Combine: Relay nodes performs standard random network coding over incoming packets to generate output ones.

Verify: For each incoming packet $\bar{\boldsymbol{y}}$, the relay node calculates the value of δ using

$$\delta = \prod_{i=1}^{m} h_i^{\bar{y}_i} \prod_{i=1}^{l+1} h_{m+i}^{\bar{y}_{m+n+i}} \tag{8}$$

$\bar{\boldsymbol{y}}$ is rejected if $\delta \neq 1$; otherwise the verification process continues. Using each of its MAC keys $\bar{\gamma}_i$, the relay node calculates the value of ξ_i using:

$$\xi_i = \sum_{r=1}^{m+n} \gamma_{i,r} \bar{y}_r + \gamma_{i,m+n+1} \bar{y}_{m+n+j} \tag{9}$$

$\bar{\boldsymbol{y}}$ is rejected if these exists a $\xi_i \neq 0$; otherwise it is accepted.

To employ the double-random key distribution in MacSig, we alternatively let the source sample $|K| > l$ MAC keys, and randomly select l from them to calculate MACs in each generation. Also, the source should attach the indexes of the l selected keys to each packet $\bar{\boldsymbol{x}}_i$. The value of $|K|$ and l can be determined according to Theorem 5.

C. Security Analysis

Theorem 6. The MacSig authentication scheme is secure against tag pollution.

Proof: First, we give some notations in this proof. Let $\bar{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{t})$ denote a packet, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ are messages symbols, and $\boldsymbol{t} = (t_1, t_2, \ldots, t_l, \sigma)$ are tags. Let $\bar{V}$ be the subspace spanned by the m source packets $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_m$, and T be the subspace spanned by their respective tags $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_m$.

We consider an attack scenario comprised of an adversary node A, and an innocent node I. Let K_I be the set of MAC keys assigned to I, with $b = |K_I| > 0$. Without loss of generality, we assume $K_I = \{\gamma_1, \gamma_2, \ldots, \gamma_b\}$, with γ_i corresponding to the i^{th} MAC of a packet. The goal of A is defined as: given a packet $\bar{\boldsymbol{y}} = (\boldsymbol{y}, \boldsymbol{t}) \in \bar{V}$, construct tags $\boldsymbol{t}' \neq \boldsymbol{t}$ so that the packet $\bar{\boldsymbol{y}}' = (\boldsymbol{y}, \boldsymbol{t}')$ passes I's verification.

Suppose that the authentication approach given in Fig. 3(a) is employed. Then $\boldsymbol{t}'$ should satisfy the following two conditions: (1) $\boldsymbol{t}' \in T$ (by Theorem 2); (2) $t'_i = t_i$ for each $i = 1, 2, \ldots, b$ (since message symbols are not modified). Let $\bar{\boldsymbol{y}}_1, \bar{\boldsymbol{y}}_2, \ldots, \bar{\boldsymbol{y}}_m$ be any m linearly independent packets belonging to $\bar{V}$, with $\bar{\boldsymbol{y}}_i = (\boldsymbol{y}_i, \boldsymbol{t}_i)$. Then any $\boldsymbol{t}'$ that satisfy the above two conditions can be represented as:

$$\boldsymbol{t}' = \boldsymbol{\alpha}(\boldsymbol{t}_1^T, \boldsymbol{t}_2^T, \ldots, \boldsymbol{t}_m^T)^T \tag{10}$$

with $\boldsymbol{\alpha}$ subjected to:

$$\boldsymbol{\alpha} \cdot \begin{pmatrix} t_{1,1}, & t_{1,2}, & \cdots & t_{1,b} \\ t_{2,1}, & t_{2,2}, & \cdots & t_{2,b} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1}, & t_{m,2}, & \cdots & t_{m,b} \end{pmatrix} = (t_1, t_2, \ldots, t_b) \tag{11}$$

Let λ be the column rank of the coefficient matrix. Assume that $b < m$, then we can obtain $p^{m-\lambda}$ solutions for $\boldsymbol{\alpha}$, meaning that there will be $p^{m-\lambda}$ possible $\boldsymbol{t}'$. As only one of these $\boldsymbol{t}'$ is equal to $\boldsymbol{t}$, the tag pollution will succeed with a probability of $1 - 1/p^{m-\lambda}$. Thus, the verification approach given by Fig. 3(a) is not secure.

Now suppose the MacSig scheme is employed instead. In this case, Condition (2) remains the same, while Condition (1) should be changed to: $(y_1, y_2, \ldots, y_m, \boldsymbol{t}') \in T'$, where T' is the subspace spanned by the first m and the last $l+1$ symbols of source packets $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_m$. As a result, the equations that $\boldsymbol{\alpha}$ is subjected to becomes:

$$\begin{aligned} &\boldsymbol{\alpha} \cdot \begin{pmatrix} y_{1,1}, & \ldots, & y_{1,m}, & t_{1,1}, & \ldots & t_{1,b} \\ y_{2,1}, & \ldots, & y_{2,m}, & t_{2,1}, & \ldots & t_{2,b} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{m,1}, & \ldots, & y_{m,m}, & t_{m,1}, & \ldots & t_{m,b} \end{pmatrix} \\ =&(y_1, \ldots, y_m, t_1, \ldots, t_b) \end{aligned} \tag{12}$$

where the first m columns are introduced by the new Condition (1). As these m columns are linearly independent, the column rank of the coefficient matrix will be m. Then there is only one solution for $\boldsymbol{\alpha}$, using which A can only construct a $\boldsymbol{t}' = \boldsymbol{t}$. Thus tag pollution is not possible. ■

V. Performance Evaluation

This section studies the performance of our MacSig authentication scheme in terms of bandwidth and computation overhead.

A. Bandwidth Overhead

We neglect the bandwidth consumed by the key distribution in the **Setup** stage, since it can be done offline. The online bandwidth overhead per packet includes l MACs, l MAC key indexes, and a signature. Since MACs and signatures are defined over $\mathbb{F}_p$, just as message symbols, they both have size of $|p| = \lceil \log_2 p \rceil$ bits. Next, we determine the size of a MAC key index. As there are $|K| = \frac{2}{\delta^2} e(c+1)^2(\gamma+1)\ln N$ MAC keys in total, a key index can be represented by $\lambda = \lceil \log_2 |K| \rceil$ bits. To get a sufficiently large λ, we assume an extreme case where $c = 10$, $\gamma = 1000$ and $N = 1000$ (then the probability h in the proof of Theorem 5 will be less than 10^{-12}). For this case, λ equals 29, but we choose $\lambda = 32$ instead for sake of byte alignment.

From the above analysis, the bandwidth overhead of the MacSig scheme is $O_b = \frac{l+1}{m+n} + \frac{32l}{|p|(m+n)}$, where $l = \frac{1}{1-\delta} e(c+1)\ln\frac{1}{\epsilon}$. We calculate O_b by fixing $\delta = 0.1$, $n = 20m$, $|p| = 128$, and varying the values of n, c, ϵ. Fig. 4 shows the relationship between bandwidth overhead per packet and packet size n (the number of symbols in each packet) for different c and ϵ. We observe that the bandwidth overhead decreases with the packet size. Especially, when the packet size is larger than 700 symbols and the number of colluding adversaries is less than 3, the per-packet bandwidth overhead sits between 5% and 10%.

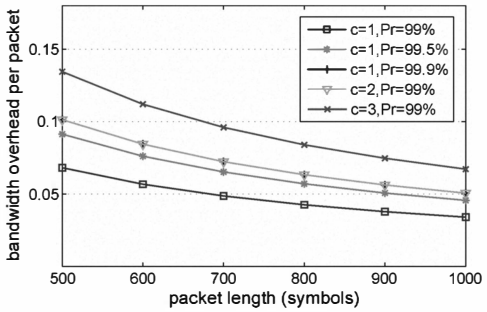

Fig. 4: The per-packet bandwidth overhead of our MacSig scheme (c is the number of coalition, and $\Pr = 1 - \epsilon$ is the security probability).

B. Computation Overhead

For a similar reason with the last subsection, we will only consider the online computation overhead incurred by the last three stages.

MAC+Sign. For each packet, the source needs $(m+n+1)l$ multiplications to calculate l MACs, and $m + l + 1$ multiplications to generate the signature for these MACs. Thus the computation overhead in this stage is $(m+n+1)l+(m+l+1)$ multiplications per packet.

Combine. Let w be the average number of packets combined in each network coding round, then an average of $w(l+1)$ multiplications are needed to combine the MACs and signatures of the corresponding packets. The computation overhead in this stage will not burden the relay nodes much more, considering the standard network coding still requires $w(m+n)$ multiplications, with the packet size $m+n$ much larger than the number of MACs l.

Verify. First of all, we need to determine the computation complexity of exponentiation over $\mathbb{F}_p$, since it is a key operation required by MacSig verification. We utilize the typical "square and multiple" method to calculate y^x over $\mathbb{F}_p$ as follows. First we compute the value of y^{2^z} for $1 \le z < |x|$, where $|x|$ is the size of x in bits. Since half of the bits of x are zero on average, we need another $\frac{1}{2}\log_2|x|$ multiplications to obtain y^x. Thus an exponentiation over $\mathbb{F}_p$ takes $\frac{3}{2}|p|$ multiplications. Secondly, to obtain a benchmark for the running time of multiplications over $\mathbb{F}_p$, we implement the Montgomery multiplication algorithm [23] on a 2.00GHz Intel Core 2 CPU. For the case of $|p| = 128$, we observe that roughly 2.5×10^5 multiplications over $\mathbb{F}_p$ can be performed per second.

Recall that to verify a packet in MacSig, $m + l + 1$ exponentiations and $(m+n+1)l$ multiplications are needed. Using the fact that an exponentiation is equivalent to $\frac{3}{2}|p|$ multiplications, the overhead of this stage is $\frac{3}{2}|p|(m+l+1)+(m+n+1)l$ multiplications. Then we use the benchmark developed above to evaluate the verification time of our MacSig scheme. By fixing $\delta = 0.1$, $n = 20m$, $|p| = 128$, $\epsilon = 1/100$, and varying n, c, we obtain the results as shown in Fig. 5. For comparison, we also include the other three signature-based schemes [12]–[14], all of which require no less than $m+n$ exponentiations to verify a packet. From Fig. 5, we observe that verification process in our MacSig scheme is 2 to 4 times faster than those of the other three.

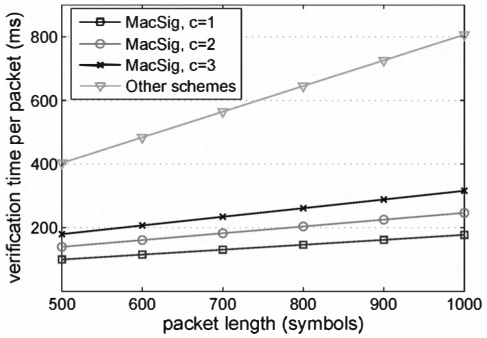

Fig. 5: The per-packet verification time of our MacSig scheme and those of the other three schemes [12]–[14] (represented as a single curve)

VI. Discussion

In the above of this paper, we have introduced three authentication schemes (HSS, HSM, and MacSig) by assuming single generation. If the transmission consists of multiple generations, the adversary can launch repetitive attack: collect legitimate packets of previous generations and use them to fake packets for subsequent ones. In the following, we discuss how to improve HSM and HSS so that they can resist this attack.

For the HSM scheme, we assume that the source and other nodes share a common pseudorandom number generator (PRNG). At the Setup stage, the source samples a pool of

random seeds, and assigns these seeds to other nodes the same way as distributing MAC keys. Before each generation, all nodes run PRGN using their seeds to generating new MAC keys. In this way, different generation will use different MAC keys, and thus repetitive attack is not possible.

For the HSS scheme, we consider the following two approaches. In the first approach, prior to each generation, the source randomly alters one element in the secret key, and informs other nodes of the changed element in the public key, just like the scheme in [13]. This approach can be effective when the adversary is unable collect legitimate packets with many zeros. In the second approach, we let the signature of a source packet $\boldsymbol{x}_i$ be $\sigma_i = g^{-\sum_{j=m+1}^{m+n} \beta_j x_{i,j}}$. For each generation, the source signs and sends the m signatures $\{\sigma_i\}_{i=1}^{m}$ of its packets, and forwarders check whether $\prod_{i=m+1}^{m+n} h_i^{y_i} \prod_{i=1}^{m} \sigma_i^{y_i} = 1$. Alternatively, since this signature scheme is homomorphic, we can have each signature travel with the packet it signs. In this way, we don't need an extra channel to transmit signatures in advance. Details of the implementation would be left for our future work.

VII. Related Work

In the context of network coding, security threats such as traffic analysis, eavesdropping attacks, and entropy attacks have been studied in [24], [25], and [26], respectively. Among all the threats considered so far, pollution attacks are perhaps the most concerned ones. To thwart this kind of attacks, many schemes have been proposed. We classify them into two categories: Information-theoretic schemes and cryptography-based schemes.

Information-theoretic schemes aim to detect or correct polluted packets at sink nodes using information-theoretic approaches. For example, Ho et al. [8] present a scheme which extends the standard random network coding to support Byzantine modification detection, and Jaggi et al. [9] study the designing issue of error-correction codes which can help sink nodes recover corrupted packets. Although these information-theoretic schemes are effective, they can only passively tolerate pollution at sinks.

To actively prevent pollution from propagating among intermediate nodes, several **cryptography-based schemes** have been proposed. Some of these schemes use **public-key cryptographic approaches**. For example, in the innovative work of Krohn et al. [10], *homomorphic hash function* is proposed to enable on-the-fly verification for erasure codes. As the verification process requires nodes to compute expensive homomorphic hash functions, the technique of *batched verification* is employed. Gkantsidis et al. [11] extend this scheme to network-coding-based P2P networks, and further reduce the computation overhead by enabling cooperative verification among peers. One common limitation of these two schemes is that the hash values of the whole file should be computed at the source and delivered to downstream nodes in advance. To address this problem, Yu et al. [12] propose a *homomorphic signature* scheme on basis of homomorphic hash function and RSA cryptosystem. In their scheme, each packet carries a RSA-encrypted homomorphic hash, which functions as its homomorphic signature. A recent work [27], however, shows that this signature scheme is only conditionally valid, and may be vulnerable to trivial no-message attacks. From quite a different perspective, Zhao et al. [13] propose a signature scheme in which relay nodes check the integrity of packets by verifying whether they belong to the subspace of source packets. Boneh et al. [14] introduces another signature scheme similar to [13], but use signatures of a smaller size. However, both [13] and [14] still require extra secure channels to pre-distribute signatures, just like [10], [11]. To sum up the above public-key cryptographic approaches, they are not computationally efficient due to the expensive operations of homomorphic hashing or signatures.

The inefficiency of public-key cryptographic approaches motives the use of **symmetric-key cryptographic approaches**, which involve very simple and efficient operations. In the scheme proposed by Yu et al. [15], the source attaches to each packet multiple MACs, each of which authenticates part of the packet. These MACs are encrypted using different keys at the source, and relay nodes can cooperatively check different MACs using their respectively shared keys. Agrawal et al. [16] propose a similar MAC-based scheme, which differs from [15] in that each MAC authenticates the whole packet. However, this scheme is unfortunately vulnerable to tag pollution, which can also degrade the system performance. Towards this problem, Li et al. [17] propose RIPPLE, a time-based authentication protocol which utilizes delayed release of secret keys. One problem with RIPPLE is that it requires global synchronization among all nodes, which is not easy to be realized in distributed settings. Kehdi et al. [18] propose another symmetric-key based scheme, which utilizes *null keys* for verification. A null key is just a vector from the null space of the matrix formed by the source packets; legitimate packets are supposed to map null keys to zero. This scheme may incur a high bandwidth overhead since it injects into the network with multiple null keys for each generation. In sum, these symmetric-key cryptographic approaches are quite efficient in computation, but they have must carefully manage the keys, and can incur a relatively large bandwidth overhead.

VIII. Conclusion

In this paper, we study how to achieve network coding authentication in the presence of normal pollution and tag pollution attacks. The basic idea is to pad each source packet with an extra symbol to make it orthogonal to a given vector. Under this idea, we propose a signature-based scheme HSS. HSS doesn't need to pre-distribute signatures for each generation, and hence incurs no start-up latency. We also propose a MAC-based scheme HSM, which employs a double-random key distribution approach. The main advantages of HSM is that the number of MACs attached to each packet scales with the network size. By utilizing the techniques of both HSS and HSM, we finally propose a hybrid-key based authentication scheme MacSig. We demonstrate that the MacSig scheme can effectively resist both normal pollution and tag pollution

attacks, while incurring a relatively low overhead in bandwidth and computation.

ACKNOWLEDGMENT

This work is supported by the National Basic Research Program of China (No. 2010CB328105, 2011CB302703, 2007CB807900, and 2007CB807901), the National Natural Science Foundation of China (No. 60970101, 60932003, 61033001,61073174, and 61061130540), Hi-Tech Research & Development Program of China Grant 2006AA10Z216, and NSF grant CNS-0905615. Part of Hongyi Yao's work was done when he was in Tsinghua University.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] D. Lun, M. Médard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proc. of IEEE International Symposium on Information Theory*, Sept. 2005.

[3] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. on Communications*, vol. 54, no. 11, Nov. 2005.

[4] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. of IEEE International Symposium on Information Theory*, Jun. 2003.

[5] C. Gkantsidis and P. Rodriguez, "Network coding for large scale file distribution," in *Proc. of IEEE INFOCOM*, Mar. 2005.

[6] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale operational on-demand streaming with random network coding," in *Proc. of IEEE INFOCOM*, Mar. 2010.

[7] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of ACM SIGCOMM*, Aug. 2007.

[8] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proc. of IEEE IEEE International Symposium on Information Theory*, Jun. 2004.

[9] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proc. of IEEE INFOCOM*, May 2007.

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

[11] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *Proc. of IEEE INFOCOM*, Apr. 2006.

[12] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing network coding against pollution attacks," in *Proc. of IEEE INFOCOM*, Apr. 2008.

[13] F. Zhao, T. Kalker, M. Médard, and K. J. Han, "Signatures for content distribution with network coding," in *Proc. of IEEE IEEE International Symposium on Information Theory*, Jun. 2007.

[14] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Proc. of International Conference on Practice and Theory in Public Key Cryptography*, 2009.

[15] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing XOR network coding against pollution attacks," in *Proc. of IEEE INFOCOM*, Apr. 2009.

[16] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Proc. of International Conference on Applied Cryptography and Network Security*, Jun. 2009.

[17] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE authentication for network coding," in *Proc. of IEEE INFOCOM*, Mar. 2010.

[18] E. Kehdi and B. Li, "Null keys: limiting malicious attacks via null space properties of network coding," in *Proc. of IEEE INFOCOM*, Apr. 2009.

[19] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *Proc. of IEEE INFOCOM*, Mar. 1999.

[20] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of the Allerton Conference*, Oct. 2003.

[21] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *Proc. of CRYPTO'99*, Aug. 1999.

[22] J. Katz and Y. Lindell, *Introduction to modern cryptography*. Chapman & Hall/CRC, 2008.

[23] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.

[24] Y. Fan, Y. Jiang, H. Zhu, and X. Shen, "An efficient privacy-preserving scheme against traffic analysis in network coding," in *Proc. of IEEE INFOCOM*, Apr. 2009.

[25] P. Zhang, Y. Jiang, C. Lin, Y. Fan, and X. Shen, "P-Coding: Secure network coding against eavesdropping attacks," in *Proc. of IEEE INFOCOM*, Mar. 2010.

[26] Y. Jiang, Y. Fan, X. Shen, and C. Lin, "A self-adaptive probabilistic packet filtering scheme against entropy attacks in network coding," *Computer Networks*, vol. 53, no. 18, pp. 3089–3101, Dec. 2009.

[27] A. Yun, J. H. Cheon, and Y. Kim, "On homomorphic signatures for network coding," *IEEE Trans. on Computers*, vol. 59, no. 9, pp. 1295–1296, Mar. 2010.

APPENDIX
PROOF OF THEOREM 5

Proof: For each node ω_i and any set A of c nodes, the probability that a specific key k is safe can be calculated by:

$$\Pr(k \text{ is safe}|\omega_i, A) = (1-\frac{1}{c+1})^c \frac{1}{c+1} < \frac{1}{e(c+1)} \quad (13)$$

Then the expected number of safe keys can be derived by:

$$E(\#\text{ of safe keys}|\omega_i, A) = \frac{|K|}{e(c+1)} = m \quad (14)$$

Using Chernoff bound, we have:

$$\Pr(\#\text{ of safe keys} < (1-\delta)m|\omega_i, A) < \exp(-\frac{\delta^2}{2}m) \quad (15)$$

Based on this, the probability that there exists a node ω_i and a set A of c nodes, such that the number of safe keys is less than $(1-\delta)m$, can be calculated by:

$$\begin{aligned}
h=&\Pr(\bigcup_{A,\omega_i}(\#\text{ of safe keys} < (1-\delta)m)|\omega_i, A)\\
<&\sum_{\omega_i,A}\Pr(\#\text{ of safe keys} < (1-\delta)m|\omega_i, A)\\
<&n\binom{N}{c}\exp(-\frac{\delta^2}{2}m) < N^{c+1}\exp(-\frac{\delta^2}{2}m)\\
=&\exp((c+1)\ln N - \frac{\delta^2}{2}(\frac{2}{\delta^2}(\gamma+1)(c+1)\ln N))\\
=&\exp(-\gamma(c+1)\ln N)\\
=&N^{-\gamma(c+1)}
\end{aligned}$$

As $\gamma \to \infty$, $h \to 0$, meaning that the number of safe keys given any ω_i and A is no less than $(1-\delta)m$. Then the probability that a randomly chosen key from K is safe for all A and ω_i is

$$r > \frac{(1-\delta)m}{|K|} = \frac{1-\delta}{e(c+1)} \quad (16)$$

If the source randomly chooses l keys from K, then the probability that no safe key is selected can be calculated by:

$$u = (1-r)^l < (1-\frac{1-\delta}{e(c+1)})^{\frac{1}{1-\delta}e(c+1)\ln\frac{1}{\epsilon}} < \epsilon. \quad (17)$$

Thus, the double-random key distribution scheme is c-secure with probability no less than $1-\epsilon$. ■