# POND: A Novel Protocol for Network Coding based on Hybrid Cryptographic Scheme

Cheng-Qiang Huang*†, Atsuko Miyaji†, Long-Hai Li*, and Shang-Mei Xu*
*School of Computer Science and Technology, Xidian University
†School of Information Science, Japan Advanced Institute of Science and Technology
cq.huang@stu.xidian.edu.cn miyaji@jaist.ac.jp lhli@xidian.edu.cn xsm@stu.xidian.edu.cn

***Abstract*—Network coding has been shown to have a lot of positive effects on networks. However, its mixing nature contributes to its vulnerability to pollution attacks. Two kinds of schemes, homomorphic MAC based schemes and homomorphic signature based schemes, have been proposed to achieve secure random linear network coding. In this paper, we present POND, a novel protocol using RIPPLE transmission protocol and homomorphic signature in the areas near and far from the source respectively. POND theoretically achieves better overall communication efficiency. And by utilizing neoteric symmetric key distribution method, it is immune to key related attacks. To the best of our knowledge, our work is the first solution which uses different authentication schemes in different parts of a network for network coding.**

***Keywords*-Network Coding, Homomorphic MAC, Homomorphic Signature, Hybrid Scheme**

## I. Introduction

Network coding, in recent years, has shown its great ability in maximizing the throughput of networks. By combining packets together, random linear network coding allows a router to enhance its performance as well as security. However, due to the combining nature of random linear network coding, it is inevitably vulnerable to *pollution attacks*, in which a malicious node tries to inject corrupted packets into or modify existing packets in the network. *Pollution attacks*, as we know, can cause significant effects on the performance of the network. In some specific cases, a single polluted packet can end up corrupting all the information reaching a destination. Hence, network coding schemes should utilize some strategies [3] - [7], such as source authentication , to deal with pollution attacks.

Previous researches, which focus on solving the pollution attack problem by in-network packet authentication, fall into two main categories: public key based schemes and symmetric key based schemes. Actually, various public key signature schemes [12] [13] have been proposed so far. However, they are quite slow when networks require fast processing of packets. On the other hand, symmetric key MAC based schemes are popular as well because of their good computation performance. Nevertheless, MAC based schemes always encounter key distribution problems. How to distribute the symmetric key for verifying MACs becomes a critical problem, because it greatly influences the performance of the whole network. Meanwhile, some MAC based schemes, such as [10], are deemed not secure enough when they utilize subspace key distribution method. That is, they are only $c$-collusion resistant, where the $c$ is pre-determined by the choice of key space and distribution method. Moreover, these schemes may also be vulnerable to *tag pollution attack* in which a MAC is tampered by an attacker and found far down the stream.

In this paper, we propose POND which is based on homomorphic MAC and homomorphic signature, two cryptographic primitives in in-network schemes for network coding. We try to take both the advantages of homomorphic MAC based and homomorphic signature based schemes. As a result, our scheme, similar to RIPPLE [1], provides arbitrarily collusion and tag pollution resistance. What's more, POND enables (a) **RIPPLE replay attack resistance**; and (b) **better overall efficiency**.

By proposing POND, we make three main contribution in this paper. Firstly, we theoretically compare RIPPLE and basic homomorphic signature scheme, showing that the performance of these schemes, to a large extend, relates to network conditions. Secondly, we show that RIPPLE, under some specific condition, is not applicable if pipeline strategy is utilized. Thirdly, to the best of our knowledge, our protocol, POND, is the first protocol which leverages different authentication schemes in different parts of network for network coding. And by properly choosing parameters, POND can be used in different kinds of networks to achieve better overall efficiency.

### A. Organization of the paper

In next section, we give our system setting of network coding and threat model. In section 3, we discuss the basic ideas of our protocol. The chosen homomorphic MAC scheme and homomorphic signature scheme in our protocol are presented afterwards. The details of POND, our new hybrid design of network coding system, will be presented in section 5. In section 6, we will discuss why better overall efficiency is possible and how to chose adequate parameters in POND. Section 7 lists our security analysis and efficiency analysis. We conclude in section 8 at last.

## II. Problem Statement

### A. System Setting

Our network model is a directed acyclic graph denoted as $G = (V, E)$. In the network, a source $S$ needs to multicast a stream of packets through intermediate nodes to multiple receivers. All intermediate nodes and receivers $v \in V - \{S\}$ perform random linear network coding.

We consider random linear network coding based on generations in this paper. Following the treatment in [1], we focus on the coding and transmission of a single generation which normally contains $m$ messages. Formally, a message $M \in \mathbb{F}_q^n$ is a vector of $n$ numbers from finite field $\mathbb{F}_q$. All arithmetic operations are done over $\mathbb{F}_q$ hereafter. Note that only messages in the same generation can be operated together. Messages from different generations are treated separately.

Actually, to help receivers decode the messages, each original message $M_i^{\text{origin}}$ is expanded as:

$$M_i = (M_i^{\text{origin}}, \overbrace{\underbrace{0, 0, \cdots, 0, 1}_{i}, 0, \cdots, 0}^{m}) \in \mathbb{F}_q^{n+m}. \quad (1)$$

Here, $i$ denotes the index of message in the whole generation.

The source in the network starts to multicast messages by sending them to its neighbors. Messages are received, random linearly combined, and then transmitted from one node to its children. More precisely, for a node $v \in V - \{S\}$ which gets $w$ messages of a same generation, it combines the messages random linearly with chosen coefficient $c_i \in \mathbb{F}_q$:

$$M = \sum_{i}^{w} c_i M_i. \quad (2)$$

Thus, the expanded part of a message carries the coefficients of random linear combination. Any node who receives the message with a full ranked extended part can recover the original messages of a whole generation successfully.

In the following parts of this paper, we call a message packet when it is expanded again by signature scheme or MAC scheme. And to measure the whole efficiency of any scheme, we consider the situation of transferring $N$ generations of packets using pipeline strategy. This is because pipeline strategy has been shown that it can help the network to achieve optimal multicast throughput.

### B. Threat Model

In POND, the adversaries we considered can only perform polynomial-time algorithms. However, they can control arbitrary numbers of nodes in the network and observe, modify, or even change the packets in the network. More precisely, the malicious behaviors the adversaries may attempt to do includes injecting corrupted packets into the network, or just simply modifying existing packets in the network. Moreover, we assume that sources in the network is trusted and an adversary does not have access to the key materials hold by sources. In our setting, an adversary is more prone to fake packets and deceive the other nodes to pass the corrupted packets for it so as to corrupt the information flow. In other words, the adversary may not want to totally block out the network. As it is clear that, if the adversaries can control an arbitrary subset of nodes in the network, blocking out the network would be no hardness.

## III. Design Goals and Basic Ideas

### A. Design Goals

Under our settings, the goal of our protocol, i.e. POND, are as follows:

- *In-network source authentication*. The source and integrity of data can be verified by any node in the network.
- *Arbitrary collusion resistant*. No matter how many adversaries are there in the network, the authenticity of data can still be verified by honest nodes.
- *Replay attack resistance*. Any adversary can not use previous keys and corresponding packets to fool the nodes who have received the packets before.
- *High system efficiency*. The materials used for authentication in data are limited, thus causing little communication overhead even when the network is large. And the authentication of data requires adequate computing resources, efficiently helping forward the data to its destination. Hence, the whole system performs in an efficient manner.

### B. Basic Ideas

Normally, arbitrary collision resistant network coding schemes, which are based on homomorphic MAC, need to separate the time for sending packets and the keys for authenticating the corresponding packets. A good example is RIPPLE [1]. In RIPPLE, to flush data from one level to the next level, corresponding key materials should be flushed as well to drive data forward. However, the key disclosure delay greatly reduces the communicate efficiency when the delay time is longer than the processing time of using homomorphic signature. Our idea comes when we try to utilize homomorphic signature instead of homomorphic MAC in some specific areas of a network. That is, when a network is big enough, we may want to use homomorphic MAC in the area near sender. But, instead of using homomorphic MAC, we may want to chose homomorphic signature as the scheme to perform network coding in the areas which are far from the sender. By leveraging better schemes in specific areas, we could probably achieve pretty good overall efficiency for a specific network. Through properly choosing different schemes, we can successfully pick diverse properties for a network.

POND, thus, utilizes the forementioned ideas and contains three main components: homomorphic MAC, homomorphic signature, and POND transmission protocol. The homomorphic MAC is the same as the one in RIPPLE [1] for now, because we want the property of arbitrary collision resistance without centralized authority. We will recall the details in section IV. The homomorphic signature is alterable as well. In this paper, we use the homomorphic signature scheme similar to that in [2] for explanation because of its simplicity.

## IV. Review

Specifically, in POND, two kinds of schemes are used in different zones of network to achieve better efficiency. We briefly describe the two schemes in this part and omit them when the details of POND transmission protocol are elaborated.

### A. *RIPPLE Homomorphic MAC*

Firstly, POND currently inherits the homomorphic MAC scheme from RIPPLE. More specifically, for a message $M \in \mathbb{F}_q^{n+m}$ the scheme works as follows.

- **Generate:** Sample a sequence $K = (K^1, K^2, \cdots, K^L)$, where $K^j \overset{R}{\leftarrow} \mathbb{F}_q^{n+m+L-j}$, $L$ is the max level of the whole network, and $j \in [1, L]$.
- **MAC:** Given a message $M \in \mathbb{F}_q^{n+m}$ and the keys $K$, compute $L$ tags:

$$\begin{aligned} t^L &= < M, K^L > \\ t^{L-1} &= < (M, t^L), K^{L-1} > \\ &\vdots \\ t^1 &= < (M, t^L, t^{L-1}, \cdots, t^2), K^1 > . \end{aligned} \tag{3}$$

  Hence the final output is a packet which is formally defined as: $P \triangleq (M, t^L, t^{L-1}, \cdots, t^1) \in \mathbb{F}_q^{n+m+L}$.
- **Verify:** Given $P$, check whether the following equation holds: $t^j =< (M, t^L, t^{L-1}, \cdots, t^{j+1}), K^j >$.
- **Combine:** Given input $(M_i, t_i^L, t_i^{L-1}, \cdots, t_i^l, c_i)_{i=1}^w$ with $w \leq m$, compute a tag $t = (t^L, t^{L-1}, \cdots, t^l)$, where $j \in [l, L]$ and $t^j = \sum_{i=1}^w c_i t_i^j$, for $M = \sum_{i=1}^w c_i M_i$.

### B. *Homomorphic Signature*

Secondly, POND currently uses the basic homomorphic signature scheme similar to that in [2]. It works as follows.

- **Setup:** Find a multiple cyclic group $\mathbb{G}$ of order $q$ and one of its generator $g$. Then sample the secret key like $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{m+n+1}) \overset{R}{\leftarrow} \mathbb{F}_q^{m+n}\mathbb{F}_q^*$. Compute the public key $\beta = g^\alpha \triangleq (g^{\alpha_1}, g^{\alpha_2}, \cdots, g^{\alpha_{m+n}}, g^{\alpha_{m+n+1}})$.
- **Sign:** Given message $M = (x_1, \cdots, x_{n+m})$ and secret key $\alpha$, compute signature as: $\sigma = -(\sum_{i=1}^{n+m} x_i \alpha_i)/\alpha_{i+1}$. Hence, the output packet is like $P \triangleq (M, \sigma) \in \mathbb{F}_q^{n+m+1}$.
- **Verify:** Given packet $P = (x_1, x_2, \cdots, x_{n+m}, \sigma)$ and public key $\beta = (\beta_1, \beta_2, \cdots, \beta_{n+m+1})$, check whether the following equation holds: $\prod_{i=1}^{n+m} \beta_i^{x_i} \cdot \beta_{n+m+1}^{\sigma} = 1$.
- **Combine:** Given $(M_i, \sigma_i, c_i)_{i=1}^w$ with $w \leq m$, compute signature $\sigma = \sum_{i=1}^w c_i \sigma_i$ for $M = \sum_{i=1}^w c_i M_i$.

Note that, although we describe these two schemes separately, they are both utilized by POND in a nested manner. That is, POND produces signature for a message in the first place and then conducts MACs to authenticate the message and its signature. After successfully signed and MAC generated, an original message $M \in \mathbb{F}_q^{n+m}$ should be turned into a packet $P \triangleq (M, \sigma, t^L, t^{L-1}, \cdots, t^1) \in \mathbb{F}_q^{n+m+1+L}$.

## V. POND Transmission Protocol

In POND, a source $S$ hierarchically organizes the network by assigning nodes with different levels according to the distance of nodes. The distance of a node is measured by the maximum hop count from the source to the node. Hence a node is further called in level-$j$ if it takes at most $j$ hops from the source to the node. And, actually in our protocol, the maximum level $S$ cares about is $L_{max}$. Thus, the whole network is organized as two different zones: a layered zone with $L_{max}$ levels and a free zone outside level-$L_{max}$. To help comprehend the notions, Fig.1 illustrates the basic skeleton of a network in the view of a sender. As you may find out, we name our protocol POND just because of the structure of the layered network with a boundary, and we currently use RIPPLE unchanged inside the boundary. Homomorphic signature is utilized outside the boundary.

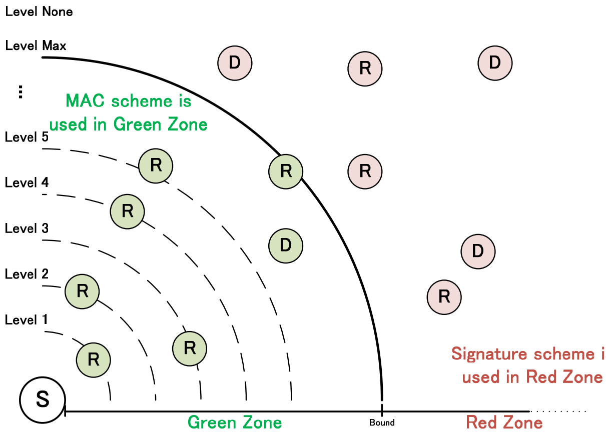


Fig. 1. Network Overview of POND

Besides organizing the network, the source $S$ also divides time into uniform intervals just like the way RIPPLE did. $S$ could send zero or multiple packets in a single interval. We also note different intervals with the index of them among all the intervals in a session. Thus, in a specific session, interval-$i$ would be the $i$th interval. We further assume that some basic knowledge about the network, such as the average transmission time between two contiguous hops $T_{trans}$, are well known. In fact, to obtain the tight estimation of the time consumption of sending data in the network, it is needed that the interval length $\delta$ is equal to average transmission time $T_{trans}$.

We now firstly describe POND in a high level and then elaborate the details in the next several sub-sections. For ease of understanding, we will simply use some parameters without detailed elaboration and leave the details about how to choose them in next section.

To begin with, a source $S$ chooses adequate parameters to setup its multicast session. After setting up the network, $S$ broadcasts a batch of packets in each interval of a session. Packets are driven by delayed keys till they reach level bound $L_{max}$. After that, packets are transmitted regardless of the delayed keys. More precisely, assume that $S$ sends a batch of packets in interval-$k$. Each packet contains a real message $m$, a signature $\sigma$ of $m$, and $L_{max}$ MACs $t_i$ $(1 \leq i \leq L_{max})$. The signature $\sigma$ can be verified by any node which has the announced public key of $S$. The MACs $t_i$ are used by nodes in level-$i$ to authenticate the packet. $S$ will disclose the seed of the symmetric key for authenticating packets sent in interval-$k$ and arrive at level-$i$ after a fixed amount of time. Upon receiving the seed, level-$i$ nodes generate the corresponding key for MAC $t_i$, verify its correctness, and encode the authenticated packets to forward. Encoding packets utilizes the homomorphic nature of homomorphic MAC and homomorphic signature. Thus the properly encoded packets are valid packets. Hence, packets are buffered, verified, encoded, and forwarded level by level till the level bound level-$L_{max}$. After that, packets peel off all the MACs and are authenticated only by signatures. Eventually, all the packets will reach their destination, even though some destinations are reached before level-$L_{max}$, some are after.

### A. Sender Setup

Now we describe the first stage of POND transmission protocol. Recall that time in a session (multicast transmission of a file) is divided into uniform intervals. Same notions as that in RIPPLE are used here: $T_0$ is the starting time of a session, $T_i$ is the starting time of interval-$i$ in the session, and $\delta$ is the interval length. Hence we have the following equation holds:

$$T_i = T_0 + i \cdot \delta, \qquad 1 \leq i \leq N. \tag{4}$$

$N$ denotes the maximum number of intervals in a session. In addition, in the setup stage, $S$ decides the proper interval length $\delta$, the maximum network level $L_{max}$, key disclosure delay $d$, and a seed chain as well as a public key pair $(sk, pk)$. Also, we use $W$ to denote the maximum number of packets sent in an interval. We now describe them into details.

*1) Interval Length $\delta$:* As we have mentioned, to have a tight estimation of our protocol, currently the interval length $\delta$ is set as the average transmission time $T_{trans}$ between two contiguous hops. Note that in our network, we assume that the transmission time between two adjacent routers are nearly the same and it is well known. Hence, we have:

$$\delta = T_{trans}. \tag{5}$$

*2) Maximum Network Level $L_{max}$:* In our protocol, $S$ will set an adeqaute[1] maximum network level $L_{max}$. We actually inherit the definition about level in RIPPLE which need to be measured by running Dijistra's algorithm or other existing shortest path algorithms in the network, and redefine the level of node $v$ with $L_{max}$:

$$L_v \triangleq \begin{cases} \mathrm{Lv}(v) & \text{if} \quad \mathrm{Lv}(v) \leq L_{max}, \\ \text{none} & \text{if} \quad \mathrm{Lv}(v) > L_{max}. \end{cases} \tag{6}$$

Here, for node $v \in V - \{S\}$, $\mathrm{Lv}(v)$ denotes the longest path, i.e. maximum hop count, from $S$ to $v$. For example, in Fig. 1, the network is divided into two zones, red zone and green zone, based on the boundary, i.e. level max. In red zone, nodes do not have exact levels. While in green zone, nodes are assigned levels based on their distance from the source. And this distance is measured by hop count.

*3) Symmetric Seed Disclosure Delay $d$:* Symmetric seed disclosure delay is set as follows:

$$d = \lceil \big(L_{max} T_{trans} + T_p\big)/\delta \rceil + 1. \tag{7}$$

Here, $T_p$ denotes the maximum processing time of packets in a single node using homomorphic MAC scheme like RIPPLE. The processing details may include authenticating all the $W$ packets sent in an interval and encoding authenticated ones. Hence, for packets sent in interval-$i$, $S$ delays the symmetric seed for level-$j$ nodes until interval $i + d \cdot j$. This is because we are sure that by the interval $i + d \cdot j$, nodes in level $j$ have received the interval-$i$ packets. Note that $j \leq L_{max}$, and for level none, we do not have seeds for them.

*4) One-way Seed Chain:* In POND, we use a one-way seed chain to generate symmetric keys used for authenticating MACs of messages. For each interval in a multicast session, we assign a seed to that interval. Thus, there are $N$ seeds in total. These seeds are a sequence of random values generated by a pseudo-random function $F$. More precisely, seeds $s_1, s_2, \cdots, s_{N-1}$ are generated by the following equation:

$$s_i = F(s_{i+1}, sid, timestamp), \tag{8}$$

where $1 \leq i < N$, $sid$ is the identity of $S$ , $timestamp$ is the identifier of a multicast session, and $s_N$ are randomly chosen by $S$. Seeds are used reversely as key materials in each level for different interval. The usage of seeds is discussed later.

All the seeds form a one-way seed chain whose first (actually last) element $s_1$ is signed by normal public key signature scheme, such as DSA [14]. To verify that a value $s_i$ is in the one-way seed chain, one can check that

$$s_1 = F^{i-1}(s_i, sid, timestamp), \tag{9}$$

where $F^n(x, y, z)$ denotes $n$ consecutive application of function $F$ with the given $y$ and $z$ unchanged, and every intermediate result as $x$. Of course, the first $x$ is $s_i$ and $F^0(x, y, z) = x$.

*5) Asymmetric Key Pair $(sk, pk)$:* To setup the asymmetric key pair, $S$ finds a multiplicative cyclic group $\mathbb{G}$ of order $q$, and select a $g$ which is a generator of $\mathbb{G}$. A secret key is:

$$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{m+n+1}) \overset{R}{\leftarrow} \mathbb{F}_q^{m+n} \mathbb{F}_q^*. \tag{10}$$

Its corresponding public key is like:

$$\beta = g^{\alpha} \triangleq (g^{\alpha_1}, g^{\alpha_2}, \cdots, g^{\alpha_{m+n+1}}). \tag{11}$$

Hence, the final key pair is $(sk, pk) = (\alpha, \beta)$.

### B. Initializing Nodes

Before any transmission is performed, all the nodes in the network should loosely synchronize their clocks. This is because we currently utilize RIPPLE as our scheme inside the boundary. Thus, all the nodes run a synchronization protocol before other operation. And the assumption is also inherited that clock drifts among nodes are negligible during a multicast session.

Besides time synchronization, an authorized initialization packet should be sent by $S$ to each other nodes in the network.

[1] We will discuss how to chose $L_{max}$ in section VI.

This packet contains all the public parameters chosen by $S$. That is, by the end of initializing nodes, any node $v \in V-\{S\}$ who accepts the initialization packet will obtain the following materials: $T_0, \delta, N, L_v, d$, authenticated $s_1$, and public key $\beta$ for homomorphic signature.

### *C. Sending Hybrid Authenticated Packets*

To apply different strategies in different network zones, $S$ leverages two methods to authenticate a message. It firstly signs a message with its secret key $\alpha$ and then generate $L_{max}$ MACs for the message and its signature in a nested manner for different levels. For a message $M \in \mathbb{F}_q^{m+n}$ which includes the interval index $i$, its corresponding packet would be like:

$$P \triangleq (M, \sigma, t^L, t^{L-1}, \cdots, t^1) \in \mathbb{F}^{m+n+1+L}, \tag{12}$$

where $L$ denotes the maximum level, i.e. $L_{max}$, $\sigma$ is the signature of $M$, and $t^i$ represents the MAC for level-$i$ nodes. Note that, the mentioned signature and MACs are homomorphic. In later subsections, we will describe how to generate the exact keys used for producing and verifying MACs.

### *D. Packet Authentication and Coding*

A node $v \in V - \{S\}$ in POND should acts accordingly based on its level. We now describe different actions for distinct levels respectively.

Upon receiving a packet $P = (M, \sigma, t^L, t^{L-1}, \cdots, t^{L_v})$, a node with normal level should buffer valid packets, authenticate them, and forward the authenticated ones. More precisely, for a node $v \in V - \{S\}$ whose $L_v \leq L_{max}$, it will buffer the packet $P$ only when the seed for the sending interval of the packet, say $i$, and $v$'s level, say $L_v$, is not released by $S$. In other words, if the equation

$$T_i \leq T_k + \Delta_k \leq T_{i+d\cdot L_v} \tag{13}$$

holds, node $v$ buffers the packet and waits for the symmetric seed to generate the key for authenticating it. In the above equation, $T_i$ and $T_{i+d\cdot L_v}$ are the starting time of interval-$i$ and interval-$(i + dL_v)$ respectively, $T_k$ represents the local time of node $v$ when it receive the packet, and $\Delta_k$ denotes the synchronization error, i.e. time difference. After buffering the packet, node $v$ waits for the seeds to generate the symmetric key to verify the MAC $t^{L_v}$. After being verified, authenticated packets of the same interval will be combined together and forwarded. Here we omit the details of how to verify MACs and how to combine them, please refer to section IV for the exact process. Note that key generation will be told in next subsection.

On the other hand, if a node $v$'s level is equal to none, then it should verify packets by their signature $\sigma$ instead of MACs. Using the public key $\beta$ of $S$, it can authenticate the packet correctly. In this case, node $v$ does not need to wait for the seeds to generate any keys for MACs. Similarly, after authenticating the packet, $v$ combines the related packets and forwards them to its children.

Upon receiving a seed packet $K = (i, s_i, sid, timestamp)$, a node $v$ stores it when $L_v \leq i$, and forwards it to the next level $(L_v + 1)$ when $L_v < i$. The details about how to deal with the seeds will be told in next subsection.

### *E. Key Distribution*

In this subsection, we illustrate how to generate symmetric keys by using the seed packets received in a node. Firstly, we describe how $S$ generates all the keys. How to reproduce the keys in a node is given later. For ease of understanding, we assume $d = 1$ and consider packets sent in a single interval $i$.

In $S$, once it has determined $d, N$, and the one-way seed chain, it starts to generate all the keys for packets sent in different intervals. Here we assume $d = 1$ for simplicity. The keys are generated by following equations.

$$\left\{ \begin{array}{rcl} F'(s_1, sid, timestamp, 1) & = & K_1^1 \end{array} \right.$$

$$\left\{ \begin{array}{rcl} F'(s_2, sid, timestamp, 1) & = & K_1^1 + K_2^1 \\ F'(s_2, sid, timestamp, 2) & = & K_1^2 \end{array} \right.$$

$$\vdots$$

$$\left\{ \begin{array}{rcl} F'(s_k, sid, timestamp, 1) & = & K_1^1 + K_2^1 + \cdots + K_k^1 \\ F'(s_k, sid, timestamp, 2) & = & K_1^2 + \cdots + K_{k-1}^2 \\ & \vdots & \\ F'(s_k, sid, timestamp, k-1) & = & K_1^{k-1} + K_2^{k-1} \\ F'(s_k, sid, timestamp, k) & = & K_1^k. \end{array} \right.$$

Here, $K_i^j$ denotes the key used for producing MAC $t^j$ of interval-$i$ messages. In other words, $K_i^j$ is used by level-$j$ nodes to verify interval-$i$ messages. $s_i$ is the seed whose index is $i$ in the one-way seed chain. $sid$ and $timestamp$ are $S$'s identity and timestamp respectively. And we should also note that, $F'(x, y, z, k) \in \mathbb{F}_q^{m+n+k}$ because, normally, a message $M \in \mathbb{F}_q^{m+n}$ and $(M, \delta) \in \mathbb{F}_q^{m+n+1}$. Hence, from the above equations, we can easily compute $K_i^j$ as

$$\begin{array}{rcl} K_i^j & = & F'(s_{j+i-1}, sid, timestamp, j) \\ & - & F'(s_{j+i-2}, sid, timestamp, j), \end{array} \tag{14}$$

for $i > 1$ and $j \geq 1$. When $i = 1$ and $j \geq 1$, we have

$$K_1^j = F'(s_j, sid, timestamp, j). \tag{15}$$

Actually, here we infer these equations because we assume that $d = 1$. For other $d$, condition will be similar. Thus we omit them for brevity. By this way, $S$ can compute all the keys for different intervals and different levels. It then uses these keys to generate MACs accordingly.

For a level-$j$ node, it need to have $K_i^j$ to verify $t^j$ of interval-$i$ packets. Hence, it calculate $K_i^j$ according to (14)(15) as well. The problem is that the node should get $s_j$ for $i = 1$, or $s_{j+i-1}$, $s_{j+i-2}$ for $i > 1$. According to seed disclosure delay $d = 1$, $s_{j+i}$ is disclosed in interval $j + i + 1$. Thus, level-$j$ nodes can gain $K_i^j$ through $s_{j+i-1}$ and $s_{j+i-2}$ which are released in interval $i + j$ and $i + j - 1$ respectively. This matches the result that $K_i^j$ can be gain after interval $i + j$ which can be regarded as the key disclosure interval $i + dj$ with $d = 1$.

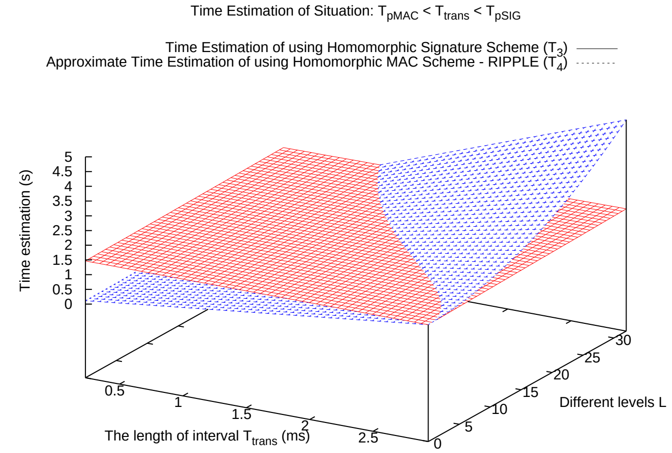


Fig. 2. When $T_{pMAC} < T_{trans} < T_{pSIG}$, using which scheme is better depends on the choice of $T_{trans}$ and $L_{max}$.

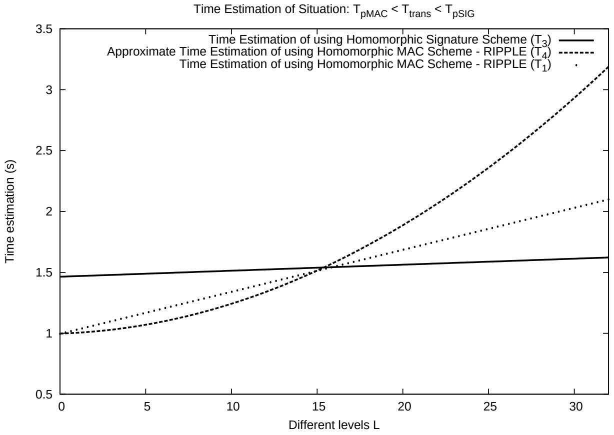


Fig. 3. Estimate $L_{max}$ and time consumption comparison

## VI. Discussion

As we have mentioned before, in POND, different strategies are used in different zones to improve the efficiency of the whole network. Because, we notice that homomorphic signature schemes, such as [2], trigger high computation delay for networks using network coding. However, homomorphic MAC schemes, such as [1], always have difficulties in distributing symmetric keys which cause a lot of communication delay. Thus, we try to combine two main kinds of schemes and use them under different occasions. In a specific network, to transmit packets from a source to its surrounding nodes, homomorphic MAC schemes are used because the communication delay will be smaller than computation delay. If packets are supposed to be transmitted to a node which is far from the source, in which case the communication delay will be dominant, homomorphic signature schemes are preferred. But where is the boundary for using these two kinds of schemes? In other words, how to set $L_{max}$?

Recall that the following notions are important when comparing the time consumption of different schemes.

1) $T_{pMAC}, T_{pSIG}$: the average time of processing a generation of packets in a node using the scheme of homomorphic MAC and homomorphic signature respectively.
2) $T_{trans}$: the average time of transmitting a generation of packets from one level to the next level. Actually it is needed that $\delta = T_{trans}$, so as to get a tight boundary. Recall that $\delta$ is the interval length.
3) $T_d$: the delay time when using homomorphic MAC scheme. Formally, that is $T_d = d\delta = dT_{trans}$, where $d = \lceil \big(L_{max}T_{trans} + T_{pMAC}\big)/\delta \rceil + 1$.
4) $L_v$: the level of node $v$ in POND.
5) $N$: the number of total generations in a multicast session.

Normally, in a specific network with similar routers, $T_{pMAC}$ and $T_{pSIG}$ are fixed. In addition, we also assume $T_{trans}$ is constant in a specific network. Thus, $N$, $L_v$, $L_{max}$ are essential variables.

### A. Estimating the time consumption of different schemes

To estimate the time consumption of RIPPLE and basic homomorphic signature scheme(HSS), Fig.4 tries to depict the transmission processes of generations of packets from a sender to nodes in each level under different occasions. Note that in all the sub-figures in Fig.4, $\delta = T_{trans}$ is set to get the tight estimation for time consumption.

When using RIPPLE, the time consumption $T$ of successfully transmitting $N$ generations of packets to the nodes in $L_v$ is like follows. When $T_{pMAC} < T_{trans}$, key disclosure delay $d = 5$, thus the time consumption can be measured straightforwardly from Fig.4(a) as:

$$\begin{aligned} T_1 &= (N + L_v - 1) \cdot T_{trans} + L_v \cdot T_{pMAC} \\ &+ L_v \cdot T_d. \end{aligned} \tag{16}$$

When $T_{pMAC} > T_{trans}$, key disclosure delay $d = 6$, and we can realize from Fig.4(b) that the queueing delay in a single router can cause fatal consequence which makes its children reject all the packets because they are outdated (their corresponding key materials have already been released). This is obvious when $T_{pMAC}$ is far bigger than $T_{trans}$. Thus, under this situation, RIPPLE transmission protocol is not applicable at all.

Similarly, when using basic homomorphic signature scheme (HSS), the time consumption $T$ can be easily measured from Fig.4(c) and (d). When $T_{pSIG} < T_{trans}$,

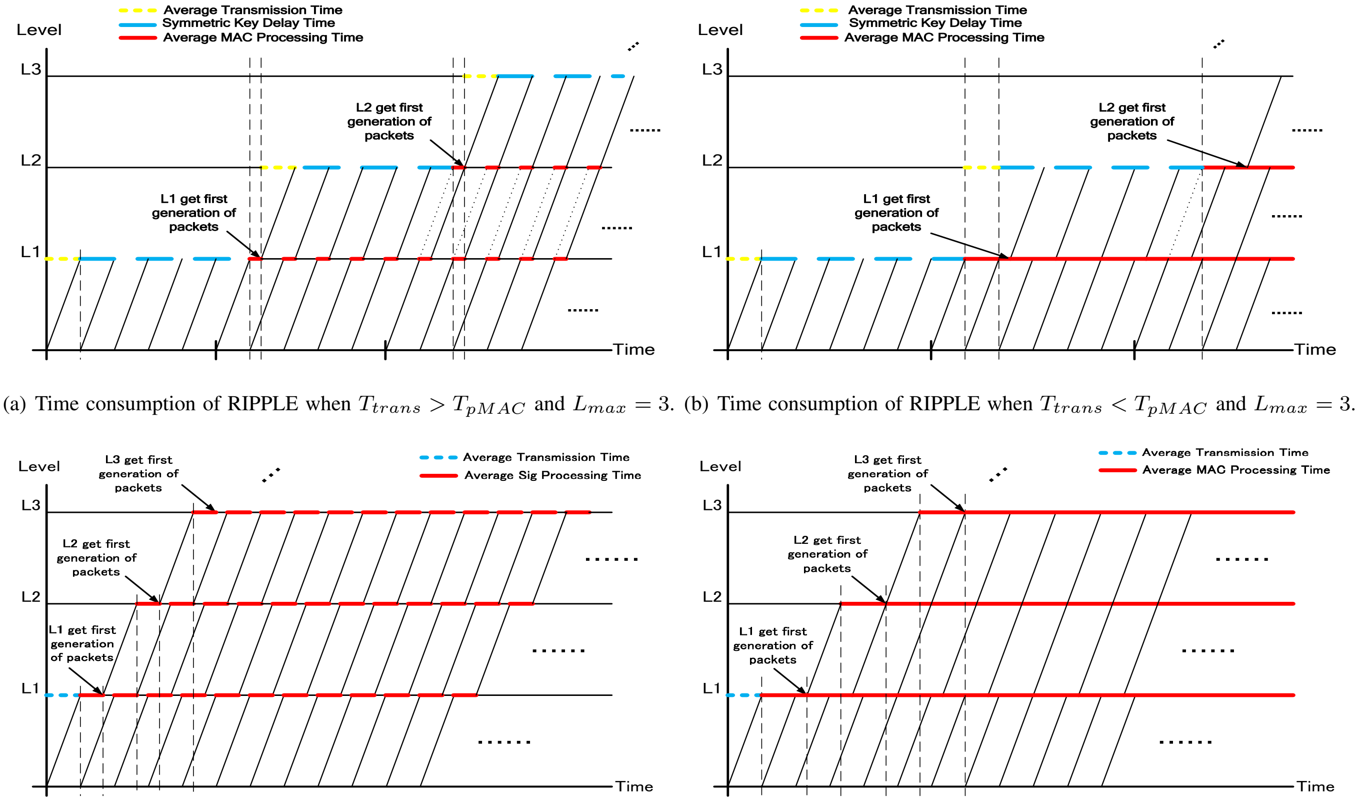
$$T_2 = (N + L_v - 1) \cdot T_{trans} + L_v \cdot T_{pSIG}; \tag{17}$$

When $T_{pSIG} > T_{trans}$,

$$T_3 = (N + L_v - 1) \cdot T_{pSIG} + L_v \cdot T_{trans}. \tag{18}$$

### B. Determining $L_{max}$

*1) :* It is obvious that, while $T_{pMAC} < T_{pSIG} < T_{trans}$, $T_1 > T_2$. This indicates that HSS is preferred in the whole network because of its lower time consumption. Hence, we set $L_{max} = 0$ to show that POND utilizes HSS entirely.


Average Transmission Time
Symmetric Key Delay Time
Average MAC Processing Time
Level
L3
L2
L1
Time
L2 get first generation of packets
L1 get first generation of packets
......
Average Transmission Time
Symmetric Key Delay Time
Average MAC Processing Time
Level
L3
L2
L1
Time
L2 get first generation of packets
L1 get first generation of packets
......
Level
L3 get first generation of packets
Average Transmission Time
Average Sig Processing Time
L3
L2 get first generation of packets
L2
L1 get first generation of packets
L1
Time
......
Level
L3 get first generation of packets
Average Transmission Time
Average MAC Processing Time
L3
L2 get first generation of packets
L2
L1 get first generation of packets
L1
Time
......

(a) Time consumption of RIPPLE when $T_{trans} > T_{pMAC}$ and $L_{max} = 3$.

(b) Time consumption of RIPPLE when $T_{trans} < T_{pMAC}$ and $L_{max} = 3$.

(c) Time consumption of HSS when $T_{trans} > T_{pSIG}$.

(d) Time consumption of HSS when $T_{trans} < T_{pSIG}$.

Fig. 4. Estimating the time consumption for different schemes under different situation.

*2) :* When $T_{pMAC} < T_{trans} < T_{pSIG}$, differences between $T_1$ and $T_3$ are complicated. Another function as follows is set to simplify the comparison.

$$\begin{aligned} T_4 &= (N + L_v - 1) \cdot T_{trans} + L_v \cdot T_{pMAC} \\ &+ L_v \cdot \big(L_v T_{trans} + T_{pMAC} + T_{trans}\big). \end{aligned} \tag{19}$$

By fixing variable $N = 500$, the data we obtain in section VII-B for $T_{pMAC}$ and $T_{pSIG}$, and setting $T_{trans}$ as x-axis, $L_v$ y-axis, and $T$ z-axis, relations between $T_3$ and $T_4$ are depicted in Fig.2. Further, when $T_{trans}$ of a network is stable, for example $T_{trans} = 2ms$, Fig.3 is presented. An intersection point is found in Fig.3 whose x-coordinate $L_{inter}$ is:

$$\begin{aligned} L_{inter} &= (A + \sqrt{A^2 - B})/(2T_{trans}), \text{where} \\ A &= T_{pSIG} - T_{trans} - 2T_{pMAC}, \text{and} \\ B &= 4T_{trans}(N-1)(T_{trans} - T_{pSIG}). \end{aligned} \tag{20}$$

It can be easily prove that $T_3 > T_1$ before the intersection point by verifying $(T_3 - T_4) > (T_1 - T_4)$. And after the intersection point, $(T_3 - T_4) < (T_1 - T_4)$, hence $T_3 < T_1$. This indicates that, under this condition, $L_{max} = L_{inter}$.

Note that, in equation (20), when variable $N$ is too large, setting $L_{max} = L_{inter}$ is not applicable because $L_{inter}$ may have exceed the maximum hop counts in a network. In this case, $L_{max}$ should be set as the maximum hop count from a sender to the farthest node which means the whole network utilizes RIPPLE only.

## VII. Analysis

### A. Security Analysis

***Theorem 1:*** The hybrid scheme of POND is a secure homomorphic scheme.

*Proof:* ***Homomorphism:*** Given $(M_i, \sigma_i, t_i^1, \cdots, t_i^L, \alpha_i)_{i=1}^w$, where $M_i = (x_{i1}, \cdots, x_{i(n+m)})$, we have combination results $M = \sum_{i=1}^w M_i = (x_1, \cdots, x_{n+m})$, where $x_j = \sum_{i=1}^w x_{ij}$, $j \in [1, n+m]$; and $\sigma = \sum_{i=1}^w \alpha_i \sigma_i$; $t = (t^1, \cdots, t^L)$ where $t^k = \sum_{i=1}^w \alpha_i t_i^k$, $k \in [1, L]$. Firstly, with the given public key $\beta = (\beta_1, \cdots, \beta_{n+m})$ and the equation $\beta_j^{\sum_i \alpha_i x_i} = \prod_i \beta_j^{\alpha_i x_i}$, the homomorphic signature holds:

$$\begin{aligned} & \prod_{j=1}^{n+m} \beta_j^{x_j} \beta_{n+m+1}^{\sigma} \\ = & \prod_{j=1}^{n+m} \beta_j^{\sum_{i=1}^w \alpha_i x_{ij}} \beta_{n+m+1}^{\sum_{i=1}^w \alpha_i \sigma_i} \\ = & \prod_{i=1}^w \Big( \prod_{j=1}^{n+m} \beta_j^{x_{ij}} \beta_{n+m+1}^{\sigma_i} \Big)^{\alpha_i} \\ = & \prod_{i=1}^w 1^{\alpha_i} = 1 \end{aligned} \tag{21}$$

Secondly, the homomorphic MAC holds:

$$\begin{aligned} t^k &= \sum_{i=1}^w \alpha_i t_i^k \\ &= \sum_{i=1}^w \alpha_i \langle (M_i, \sigma_i, t_i^L, \cdots, t_i^{k+1}), K^k \rangle \\ &= \langle \sum_{i=1}^w \alpha_i (M_i, \sigma_i, t_i^L, \cdots, t_i^{k+1}), K^k \rangle \\ &= \langle (M, \sigma, t^L, \cdots, t^{k+1}), K^k \rangle. \end{aligned} \tag{22}$$

Hence, $(M, \sigma, t^L, \cdots, t^1) = \sum_{i=1}^w \alpha_i (M_i, \sigma_i, t_i^L, \cdots, t_i^1)$ pass the homomorphic verification for $\sigma$ and $t^k$.

| Scheme | **Generate** | **Verify** and **Combine** | Size |
|---|---|---|---|
| **MAC** | 2.96 ms | 231 us | $16 \cdot L_{max}$ bytes |
| **SIG** | 129 us | 2.935 ms | 16 bytes |

TABLE I
COMPUTATION AND COMMUNICATION OVERHEAD FOR DIFFERENT SCHEME

***Security:*** An adversary who can forge a valid packet for $M \notin Span(M_1, \cdots, M_w)$ should be able to forge $\sigma$ and $t^k$ where $k \in [1, L]$. However, as proved in [1] and [2], one could forge $\sigma$ or $t^k$ with a probability at most $1/q$, where $q$ is a big prime number. Thus our scheme is secure. □

***Arbitrary Collusion and Tag Pollution Attack Resistance:*** POND inherits this property directly from the nested inner-product scheme in RIPPLE and normal signature scheme.

***RIPPLE Replay Attack Resistance:*** In RIPPLE, a one-way key chain is generated to provide keys for producing and verifying MACs. However, the security of the key chain is only protected by the signature of the last key. Without other authentication methods, an adversary can easily replay messages of a sender by signing a key in the key chain and use the partial key chain (the chosen key and its previous keys). That is, the adversary can peel off the MACs in a message of the sender which is not generated by keys in the partial key chain, and keep the remaining part as its own message. In this case, the adversary can replay messages of the original sender without knowing the details of the original message. If the original messages are sensitive, for example, commands from director in an army, it would cause fatal effects.

In our scheme, a one-way seed chain is utilized. Although an adversary can still sign a seed in the seed chain, it would not have any practical effect due to the time-related and identity-related key generating method. The key generating method also links seeds. To fool a node with partial seeds, an adversary should be able to fool nodes with the first chosen seed as the first seed. This is impossible because level is a parameter in the key generating function and even though the adversary can fool nodes with their level, the seed verification would indicate the mismatch of the index of seeds and levels. Note that the replay attack of signing the first seed or key can be easily detected by recording previous first seeds or keys.

### *B. Efficiency Analysis*

TABLE I presents the comparison between the homomorphic MAC scheme and homomorphic signature scheme in POND. We set in our experiment that $q = 2^{128}$, original message size $n = 64 \cdot 16$ bytes, size of generation $m = 6$, levels $L = 32$, and the number of parents per node $w = 6$. In our experiment, OpenSSL library [15] is used on the platform of 32bits GNU/Linux system with 1.73GHz Intel Pentium(R) Dual CPU T2370. By theoretically comparing using different schemes for network coding, we gain the theoretical estimation of the time consumption for different schemes when leveraging pipeline strategy. It is obvious in Fig.3 that under the condition of $T_{pMAC} < T_{trans} < T_{pSIG}$, hybrid scheme is able to gain better efficiency than any single scheme. Note that, in a network with many levels, POND is able to limit the number of MACs in a message by setting $L_{max}$ so as to reduce the communication overhead.

## VIII. CONCLUSION

In this paper, we present POND, a novel hybrid scheme for network coding. POND theoretically achieve better communication efficiency by using different schemes in different parts of a network and it disables RIPPLE replay attack by proposing new key distribution method. We prove that POND is correct and secure, and give exact regulations of how to choose scheme in POND. Also, we discover that, under some situations, RIPPLE is not applicable because of the message queuing delay.

## ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China under grant 61101142 and the Fundamental Research Funds for the Central Universities under grant K50510030012.

## REFERENCES

[1] Li Y., Yao H., Chen M., Jaggi S., Rosen A. *RIPPLE authentication for network coding*. In INFOCOM, 2010 Proceedings IEEE (pp. 1-9).

[2] Zhang P., Jiang Y., Lin C., Yao H., Wasef A., Shen X. *Padding for orthogonality: Efficient subspace authentication for network coding*. In INFOCOM, 2011 Proceedings IEEE (pp. 1026-1034).

[3] Guangjun L., Bin W. *Secure network coding against intra/inter-generation pollution attacks*. Communications, China, 10(8), 100-110.

[4] Perrig A., Canetti R., Song D., Tygar J. D. *Efficient and secure source authentication for multicast*. In Network and Distributed System Security Symposium, NDSS (Vol. 1, pp. 35-46).

[5] Le A., Markopoulou, A. *Cooperative defense against pollution attacks in network coding using SpaceMac*. Selected Areas in Communications, IEEE Journal on, 30(2), 442-449.

[6] Le A., Markopoulou, A. *TESLA-based defense against pollution attacks in p2p systems with network coding*. In Network Coding (NetCod), 2011 International Symposium on (pp. 1-7). IEEE.

[7] Agrawal S., Boneh D., Boyen X., Freeman D. M. *Preventing pollution attacks in multi-source network coding*. In Public Key CryptographyCP-KC 2010 (pp. 161-176). Springer Berlin Heidelberg.

[8] Perrig A., Canetti R., Tygar J. D., Song D. *Efficient authentication and signing of multicast streams over lossy channels*. In Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on (pp. 56-73).

[9] Le A., Markopoulou A. *Locating byzantine attackers in intra-session network coding using spacemac*. In Network Coding (NetCod), 2010 IEEE International Symposium on (pp. 1-6). IEEE.

[10] Agrawal S., Boneh, D. *Homomorphic MACs: MAC-based integrity for network coding*. In Applied Cryptography and Network Security (pp. 292-305). Springer Berlin Heidelberg.

[11] Cheng C., Jiang T. *An efficient homomorphic MAC with small key size for authentication in network coding*. IEEE Transactions on Computers, VOL. 62, No. 10. Oct. 2013.

[12] Charles D., Jain K., Lauter K. *Signatures for network coding*. International Journal of Information and Coding Theory, 1(1), 3-14.

[13] Boneh D., Freeman D., Katz J., Waters B. *Signing a linear subspace: Signature schemes for network coding*. In Public Key CryptographyCP-KC 2009 (pp. 68-87). Springer Berlin Heidelberg.

[14] U. S. National Institute of Standards and Technology(NIST). *Digital Signature Standard (DSS), Federal Register 56. FIPS PUB 186*.

[15] https://www.openssl.org