

On Detecting Pollution Attacks in Inter-Session Network Coding

Anh Le, Athina Markopoulou
University of California, Irvine
{anh.le, athina}@uci.edu

Abstract—Dealing with pollution attacks in inter-session network coding is challenging due to the fact that sources, in addition to intermediate nodes, can be malicious. In this work, we first define precisely corrupted packets in inter-session pollution based on the commitment of the source packets. We then propose three detection schemes: one hash-based and two MAC-based schemes: $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$. $\text{InterMac}_{\text{CPK}}$ is the first multi-source homomorphic MAC scheme that supports multiple keys. Both MAC schemes can replace traditional MACs, e.g., HMAC, in networks that employ inter-session coding. All three schemes provide in-network detection, are collusion-resistant, and have very low online bandwidth and computation overhead.

I. INTRODUCTION

Network coding involves packets being combined at intermediate nodes inside the network. Depending on whether packets from the same or different sessions are combined, network coding is classified as *intra-session* or *inter-session*, respectively. Inter-session coding, that is the focus of this paper, has been implemented in practice, such as in wireless mesh networks [1], [2] and streaming gestures [3].

The mixing nature of network coding makes it extremely vulnerable to *pollution* attacks. In such an attack, malicious nodes inject corrupted packets that are combined and forwarded by downstream nodes, resulting in a large number of corrupted packets propagating in the network. This wastes network resources, such as bandwidth and CPU time. More critically, it prevents receivers from decoding the original packets. A large body of work has focused on pollution attacks in intra-session coding [4]–[25]; meanwhile, pollution attacks in inter-session coding have received significantly less attention [27]–[29].

In this paper, our goal is to detect pollution attacks in inter-session network coding using cryptographic primitives. This is particularly challenging because not only intermediate nodes but also sources can be malicious. Recently, Agrawal *et al.* [27] formulated the problem for the first time and presented a detection scheme based on homomorphic signatures. (Homomorphic property entails that signatures are combinable without the knowledge of the private key.) This scheme has high computation overhead due to many public-key signature verification and modular exponentiation operations performed at each node per packet. Furthermore, the signature size is large and does not scale (see Section II).

In this paper, we introduce three novel detection schemes: one hash-based and two message authentication code (MAC)-based schemes, all of which are significantly more efficient than the one proposed in [27]. The key ingredient of our approaches is the use of *commitment (to a trusted controller) of source packets*. This commitment allows us to precisely define corrupted packets, thereby enabling detection of all corrupted

packets, including some that [27] cannot detect. We build upon this idea and design three schemes:

- A hash-based detection scheme, that combines homomorphic [30] and traditional hash functions, e.g., SHA-1.
- $\text{InterMac}_{\text{CPK}}$, a multi-source homomorphic MAC scheme. To the best of our knowledge, this is the first homomorphic MAC scheme having tags generated under different keys.
- $\text{SpaceMac}_{\text{PM}}$, a combination of an existing inner-product homomorphic MAC scheme [25] (built for intra-session coding) and a private inner-product protocol [31].

Our schemes have several desired properties. The hash-based scheme allows nodes to detect corrupted packets right after they receive them, thus providing *in-network detection*. It is also arbitrarily *collusion-resistant*. Both of the MAC schemes can replace traditional MACs, e.g., HMAC [32], to provide *end-to-end detection*. Moreover, they can be extended to provide in-network detection. A detection scheme built on one of our MAC schemes could be either arbitrarily collusion-resistant or *c*-collusion-resistant, for a predetermined small *c*. We also custom design commitment schemes that offer high bandwidth efficiency for both MAC schemes. Most importantly, all proposed schemes have significantly higher *bandwidth* and *computational efficiency* than those of the state-of-the-art detection scheme for inter-session coding [27]. More specifically, simulation results show that for a detection scheme built on one of our MAC schemes, both the online bandwidth and computation overhead are low, as low as 3% and 4 ms, respectively.

The schemes proposed in this paper provide a range of approaches for detecting corrupted packets in inter-session network coding and provide different tradeoffs. In general, the MAC-based schemes have significantly lower computation overhead than the hash-based scheme (Section VI-B). $\text{SpaceMac}_{\text{PM}}$ offers lower commitment overhead (Section VI-A), but $\text{InterMac}_{\text{CPK}}$ is less vulnerable to colluding malicious receivers (see the end of Section V-D).

The rest of this paper is organized as follows. Section II presents related work. In Section III, we describe the network operations, threat model, and definition of corrupted packets. In Section IV and V, we describe our hash-based and MAC-based schemes respectively. In Section VI, we evaluate the performance of our schemes. Section VII concludes the paper.

II. RELATED WORK

Because pollution attacks pose a severe threat to the success of network coding, a large body of research has been devoted to designing defense mechanisms, including both information theoretic and cryptographic approaches. The existing approaches provide error-correction capability [4]–[7], attack detection [8]–[18], [20], [23], [24], [27], and attacker identification [21], [22], [25], [29]. Most of these approaches, including our prior work

This work was supported by an NSF CAREER award (0747110) and by an AFOSR MURI (FA9550-09-1-0643).

[23]–[25], are proposed for intra-session coding and are not applicable to inter-session coding, as discussed in Section III-C. We refer the reader to [23] for a comprehensive overview of intra-session defense mechanisms. Here, we focus on defense against pollution attacks in inter-session network coding.

Agrawal *et al.* [27] proposed a homomorphic signature scheme to provide in-network detection for inter-session network coding. In their scheme, the signature of a packet sent by a source S consists of g hash values of all g source packets sent by S , together with a public key signature of the hash values. The hash values are computed using a homomorphic hash function proposed in [30]. The signature of the hashes is computed using a secure signature scheme. The signature $\sigma_{\mathbf{y}}$ of a packet \mathbf{y} , that is a linear combination of packets belonging to ℓ different flows, is the concatenation of the signatures of ℓ different signatures. The main drawbacks of this scheme are (i) the expensive verification: the verification of $\sigma_{\mathbf{y}}$ involves ℓ public-key signature verification and one homomorphic hash verification, and (ii) the large signature size: the size of $\sigma_{\mathbf{y}}$ is large, including ℓ public-key signatures and $g\ell$ hash values.

The approaches proposed in this paper are inherently different from [27]. We leverage the commitment of source packets and build our detection schemes based on un-key and symmetric-key cryptographic primitives, as opposed to public-key primitives. We significantly improve the bandwidth and computation efficiency over [27] (Section VI). Furthermore, by precisely defining corrupted packets, our schemes are able to detect some corrupted packets that [27] cannot (Section III-D).

Dong *et al.* [29] proposed a scheme that allows for identifying malicious nodes in inter-session network coding. When a pollution is detected, a bit-level traceback procedure is executed to identify the attacker. Our detection schemes are orthogonal and complementary to this identification scheme. Finally, there are several schemes that use watchdogs, *i.e.*, trusted in-network nodes that can overhear packets in wireless networks, to detect malicious nodes [33]–[35]. Our work, however, assume that all intermediate nodes may be malicious.

III. PROBLEM FORMULATION

A. Network Model and Operation

Consider a graph denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. There are s pairs of source-receiver in the network, denoted by $(S_i, R_i), i \in [1, s]$. Each source, S_i , sends packets to its corresponding receiver, R_i , by first dividing the packets into generations. For simplicity, we assume that all sources use the same generation size, g . It is straightforward to extend our schemes to accommodate different generation sizes. S_i interprets its packets, $\hat{\mathbf{v}}_{i,j}, j \in [1, g]$, as vectors in an n -dimensional vector space over a finite field \mathbb{F}_q , *i.e.*, vectors with n data symbols. Before sending, S_i appends to $\hat{\mathbf{v}}_{i,j}$ its coding coefficient, forming g augmented packets, $\mathbf{v}_{i,1}, \dots, \mathbf{v}_{i,g}$:

$$\mathbf{v}_{i,j} = \left(\underbrace{-\hat{\mathbf{v}}_{i,j}}_{g \times (i-1)}, \underbrace{0, \dots, 0}_g, \underbrace{0, \dots, 0, 1, 0, \dots, 0}_g, \underbrace{0, \dots, 0}_{g \times (s-i)} \right).$$

We refer to the augmented packets, $\mathbf{v}_{i,j}$'s, as *source packets* and $\hat{\mathbf{v}}_{i,j}$ as data of $\mathbf{v}_{i,j}$. We use $\text{aug}(\mathbf{v}_{i,j})$ to denote the coding coefficients of $\mathbf{v}_{i,j}$.

For each generation, there are $m \stackrel{\text{def}}{=} sg$ source packets. The sources send source packets into the network generation

by generation. Intermediate nodes in the network perform generation-based linear network coding, *i.e.*, they linearly combine packets that belong to the same generation. Packets sent from different sources may be combined by intermediate nodes. For example, when an intermediate node N receives ℓ packets, $\mathbf{w}_1, \dots, \mathbf{w}_\ell$, that are some linear combinations of the source packets sent by any set of sources, it chooses ℓ *local coding coefficients*, $\alpha_1, \dots, \alpha_\ell$ (depending on the coding scheme), and then transmits $\mathbf{y} = \sum_{i=1}^{\ell} \alpha_i \mathbf{w}_i$ to one or more of its outgoing edges. Note that if \mathbf{y} is a linear combination of the source packets $\mathbf{v}_{i,j}$'s then the last m symbols of \mathbf{y} contain its *global coding coefficients*. For clarity, we focus on the transmission of a single generation by all the sources.

When all nodes in the networks are benign, all packets in the network are linear combinations of the source packets. A receiver, R_i , can decode the original packets sent by its corresponding source S_i after collecting enough packets. In particular, after collecting m linearly independent packets, R_i can decode the original packets by applying Gaussian elimination on the $m \times (n + m)$ matrix formed by the collected packets. R_i may also be able to decode using less than m linearly independent packets because R_i is not interested in packets sent by the other sources.

B. Inter-Session Network Coding Characteristics

In inter-session network coding, intermediate nodes may often be able to decode source packets from the received coded packets. For instance, in COPE [1], every encoded packet is decoded at the next hop. There are also other coding schemes where encoded packet are decoded by either the first hop or the second hop, *e.g.*, see [2] and [3]. Furthermore, in inter-session coding, source packets of a source S_i may not traverse the whole network but only some parts of the network. For instance, in a directed acyclic graph, packets sent from S_i should not travel to nodes that have no path to R_i . We will exploit these observations later in the proposed schemes.

Finally, the most important observation is that, in inter-session coding, not only intermediate nodes but also some sources may be malicious. This differentiates the scenario we study in this work from single-source intra-session coding. We explicitly take this into account in our threat model below.

C. Threat Model

We assume that up to $s - 1$ sources could be malicious, any intermediate node may be malicious, and the receivers are trusted. To pollute the network, the malicious nodes may generate and inject any type of traffic into the network; they may also collude among themselves. We assume the attackers know about the construction of any cryptographic primitive used, but the their running time is polynomial in the security parameter of cryptographic primitives.

Example Attack. Fig. 1 depicts a pollution attack on the classic butterfly network coding across two unicast sessions.

Intra-Session Detection Failure. Both un-key and key-based cryptographic approaches developed for intra-session fail to detect corrupted packets in the inter-session threat model. The ways they fail are different. We first consider applying the hash-based scheme proposed in [12]. Prior to the transmission, A ,

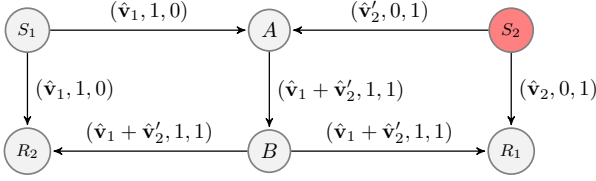


Fig. 1. An example of pollution attack in inter-session network coding. Source S_2 is malicious and all other nodes are benign. Generation size is 1. Local coding coefficients at A are fixed: $\alpha_1 = \alpha_2 = 1$. S_2 pollutes the flow S_1 - R_1 by injecting conflicting source packets $(\hat{v}'_2, 0, 1)$ and $(\hat{v}_2, 0, 1)$. R_1 decodes and recovers incorrect \hat{v}_1 : $\hat{v}_1 + \hat{v}'_2 - \hat{v}_2$.

B , R_1 , and R_2 download the hash of \hat{v}_1 from S_1 and hash of \hat{v}_2 from S_2 . S_2 can act maliciously by sending to R_1 the hash of \hat{v}_2 but sending to A and B the hash of \hat{v}'_2 . This makes A accept $(\hat{v}'_2, 1, 0)$, B accept $(\hat{v}_1 + \hat{v}'_2, 1, 1)$, and R_1 accept $(\hat{v}_2, 1, 0)$. Therefore, S_2 can still carry out the same attack.

Now, let us consider applying any of the proposed MAC or signature-based approaches, such as, [10], [15]–[18], [20], [23]. When using any one of these schemes, MAC tags or signatures of packets must be generated under the same (private or symmetric) secret key so that the homomorphic property of the scheme holds. But if this is the case, a malicious source knowing the key can generate a valid tag/signature of any packet of its interest and pollute the network. For instance, S_2 can send to R_1 $(\hat{v}'_1, 1, 0)$ and its valid tag/signature, where $\hat{v}_1 \neq \hat{v}'_1$, and R_1 will accept this corrupted packet.

D. Corrupted Packet

Loosely speaking, any packet that pollutes flows of benign sources can be considered corrupted. However, in order to detect a pollution attack, corrupted packets must be precisely defined. We first require that each source, S_i , commits to its source packets before the transmission, e.g., one way is by sending the hashes of the packets to a trusted controller. We then define a corrupted packet based on this commitment. (i) In our hash-based scheme, we require each source to commit to the data of each of its packets. We call the space spanned by the committed data of all the sources *committed source data space* and denote it by $\hat{\Pi}$. (ii) In our MAC-based schemes, we require each source to commit to its whole packets as opposed to just the data. We call the space spanned by all the committed source packets the *committed source space* and denote it by Π .

Definition 1. Let $\hat{\Pi}$ and Π be the committed source data space and committed source space, respectively. A packet \mathbf{y} is considered corrupted if $\hat{\mathbf{y}} \notin \hat{\Pi}$ or $\mathbf{y} \notin \Pi$.

This definition helps us to design detection schemes capable of detecting all corrupted packets. For instance, in Fig. 1, if S_2 commits to \hat{v}_2 then our schemes will help nodes A to drop $(\hat{v}'_2, 0, 1)$, thus avoiding having $(\hat{v}_1 + \hat{v}'_2, 1, 1)$. In contrast, the scheme in [27] only helps a node to detect conflicting packets and does not detect all corrupted packets. For instance, if [27] is used, A and B still accept \hat{v}'_2 and $(\hat{v}_1 + \hat{v}'_2, 1, 1)$, respectively. $(\hat{v}_1 + \hat{v}'_2, 1, 1)$ is detected as corrupted at R_1 if R_1 receives \hat{v}_2 first or vice versa.

E. Trusted Controller

Trusted controllers have been used explicitly in previous work that identify and eliminate attackers [21], [22], [25]. They

have also been introduced implicitly by other detection schemes [9]–[20], where a *trusted source* setups and distributes hash values, MAC tags, and keys. In this work, we explicitly uses a trusted controller to support the commitment.

IV. HASH-BASED DETECTION

A. Key Observations and Approach

Observation 1. Let us revisit the discussion of applying homomorphic hash functions to inter-session network coding in Section III-C. We observe that the main reason why S_2 can successfully pollute flow S_1 - R_1 is that S_2 is able to distribute different hash values of \hat{v}_2 and \hat{v}'_2 to A , B , and R_1 . If all nodes in the network receive the same hash value, either hash of \hat{v}_2 or \hat{v}'_2 , then S_2 will not be able to carry out the attack because one of the two will be dropped due to incorrect hash.

Observation 2. As mentioned in Section III-B, in inter-session network coding, it is often the case that intermediate nodes completely decode coded packets and recover their corresponding source packets. We exploit this fact and propose to use traditional hash functions to check for the integrity of these decodable packets. Note that a traditional hash verification is two to three orders of magnitude less expensive than a homomorphic one.

Approach. Denote the trusted controller by C . The scheme is based on the above observations and works as follows:

Setup: C sends to every node the description of a homomorphic hash (e.g., \mathcal{H} -DL, described in the next section) as well as a traditional hash function, e.g., SHA-1. Before sending, each source, S_i , $i \in [1, s]$, augments its data following the augmentation scheme described in Section III. For every source packet, \mathbf{v}_{ij} , $j \in [1, g]$, S_i computes a homomorphic hash value, h_{ij} , and a traditional hash value, \bar{h}_{ij} . Each source then sends both h_{ij} and \bar{h}_{ij} to C . The commitment of each source are the pairs (h_{ij}, \bar{h}_{ij}) . Every node downloads these pairs from C . We assume the hash descriptions and values are distributed through authentic (tampering resistant) channels as usual applications of hash. Fig. 2 illustrates how the hashes are distributed for the network of Fig. 1.

Sending: At each node, sending packets, including linearly combining incoming packets, is performed as usual.

Receiving and Verification: Upon receiving a packet \mathbf{y} , if a node is specified to decode by the coding scheme, it checks if it can recover a source packet by decoding using \mathbf{y} and its previously received packets. (i) If it can, it uses the traditional hash check to verify the integrity of the packet. (ii) If it cannot or in the case the node is not specified to decode, it uses the homomorphic hash check to verify the integrity of \mathbf{y} . If the recovered source packet (case (i)) or \mathbf{y} (case (ii)) passes the verification, the node marks \mathbf{y} as legitimate and uses it in subsequent transmissions; otherwise, it drops \mathbf{y} .

B. Homomorphic Hash Scheme

A homomorphic hash scheme consists of three polynomial-time algorithms:

- $\text{HashSetup}(1^\lambda, n)$: Input: unary representation of the security parameter, λ , and the dimension of the data space, n . Output: public parameters, pp .

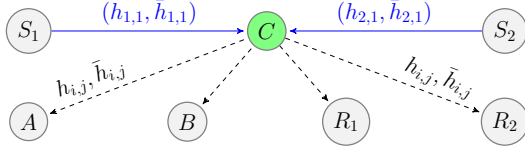


Fig. 2. Commitment and hash distribution for the network of Fig. 1. The homomorphic and regular hash values are first committed by the sources to the trusted controller; then, they are distributed to the other nodes.

- $\text{Hash}(\text{pp}, \hat{\mathbf{v}})$: Input: public parameters, pp , and a data vector, $\hat{\mathbf{v}} \in \mathbb{F}_q^n$. Output: hash value, $h \in \mathbb{F}_q$, of $\hat{\mathbf{v}}$.
 - The hash of $\hat{\mathbf{y}}$, a linear combination of m source data vectors, $\hat{\mathbf{v}}_i, i \in [1, m]$, is a hash vector, $\mathbf{h} = (h_1, \dots, h_m)$, where $h_i = \text{Hash}(\text{pp}, \hat{\mathbf{v}}_i)$.
- $\text{Test}(\text{pp}, \hat{\mathbf{y}}, \bar{\beta}, \mathbf{h})$: Input: public parameters, pp , a vector, $\hat{\mathbf{y}} \in \mathbb{F}_q^n$, a vector of coefficient, $\bar{\beta} \in \mathbb{F}_q^m$, and a hash vector, $\mathbf{h} \in \mathbb{F}_q^m$. Output: \top (true) or \perp (false).

Correctness. For all $\text{pp} \leftarrow \text{HashSetup}(1^\lambda, n)$, we require the following properties for the correctness of the scheme:

- For all $\hat{\mathbf{v}} \in \mathbb{F}_q^n$, if $h = \text{Hash}(\text{pp}, \hat{\mathbf{v}})$ then for all $i \in [1, m]$, $\text{Test}(\text{pp}, \hat{\mathbf{v}}, \mathbf{e}_i, \mathbf{h}) = \top$, where \mathbf{e}_i is the i -th unit vector of the space \mathbb{F}_q^m and the j -th component of \mathbf{h} , $\mathbf{h}^{(j)}$, is defined as follows: $\mathbf{h}^{(j)}$ equals h if j equals i and equals r_j otherwise, where r_j is any value in \mathbb{F}_q .
- For all $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2 \in \mathbb{F}_q^n$, $\bar{\beta}_1, \bar{\beta}_2 \in \mathbb{F}_q^m$, and $\alpha_1, \alpha_2 \in \mathbb{F}_q$, let $\hat{\mathbf{y}} = \alpha_1 \hat{\mathbf{y}}_1 + \alpha_2 \hat{\mathbf{y}}_2$ and $\bar{\beta} = \alpha_1 \bar{\beta}_1 + \alpha_2 \bar{\beta}_2$. We require that if $\text{Test}(\text{pp}, \hat{\mathbf{y}}_i, \bar{\beta}_i, \mathbf{h}) = \top$ for $i = 1, 2$ then $\text{Test}(\text{pp}, \hat{\mathbf{y}}, \bar{\beta}, \mathbf{h}) = \top$.

Security. Let $\mathcal{H} = (\text{HashSetup}, \text{Hash}, \text{Test})$ be a homomorphic hash. Let \mathcal{A} be a probabilistic polynomial time (PPT) adversary that takes as input $\text{pp} \leftarrow \text{HashSetup}(1^\lambda, n)$ and outputs $\mathbf{v}^* \in \mathbb{F}_q^{n+m}$, an m -dimensional space V represented as basis vectors, $\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_m$, and a hash vector, $\mathbf{h} \in \mathbb{F}_q^m$.

Definition 2. We say that \mathcal{A} breaks the homomorphic hash scheme \mathcal{H} if (i) $\hat{\mathbf{v}}^* \notin V$, (ii) $\text{Test}(\text{pp}, \hat{\mathbf{v}}_i, \mathbf{e}_i, \mathbf{h}) = \top$ for $i = 1, \dots, m$, and (iii) $\text{Test}(\text{pp}, \hat{\mathbf{v}}^*, \text{aug}(\mathbf{v}^*), \mathbf{h}) = \top$. We define the advantage $\text{Hash-Adv}[\mathcal{A}, \mathcal{H}]$ of \mathcal{A} to be the probability that \mathcal{A} breaks \mathcal{H} . We say that \mathcal{H} is secure if for all PPT \mathcal{A} , $\text{Hash-Adv}[\mathcal{A}, \mathcal{H}]$ is negligible in the security parameter λ .

Example Homomorphic Hash \mathcal{H} -DL. This construction is based on \mathcal{VH} -DL [27] but customized to work with our augmentation scheme.

- $\text{HashSetup}(1^\lambda, n)$:
 - Choose a finite cyclic group \mathbb{G} of prime order $q > 2^\lambda$.
 - Choose generators $g_i \stackrel{\mathbb{R}}{\leftarrow} \mathbb{G} \setminus \{1\}$ for $i = 1, \dots, n$.
 - Output $\text{pp} := q, (g_1, \dots, g_n)$ and the description of \mathbb{G} .
- $\text{Hash}(\text{pp}, \hat{\mathbf{v}})$:
 - Output $h := \prod_{i=1}^n \exp(g_i, \mathbf{v}^{(i)})$, where $\exp(a, b) = a^b$.
- $\text{Test}(\text{pp}, \hat{\mathbf{y}}, \bar{\beta}, \mathbf{h})$: If

$$\prod_{i=1}^n \exp(g_i, \mathbf{y}^{(i)}) = \prod_{i=1}^m \exp(\mathbf{h}^{(i)}, \bar{\beta}^{(i)})$$

then output \top ; otherwise, output \perp .

Theorem 1. The homomorphic hash \mathcal{H} -DL is secure assuming the discrete logarithm problem in \mathbb{G} is hard. In particular, let \mathcal{A} be a PPT adversary that breaks \mathcal{H} -DL, then there

exists a polynomial-time algorithm \mathcal{B} that computes discrete logarithms in \mathbb{G} such that $\text{Hash-Adv}[\mathcal{A}, \mathcal{H}\text{-DL}] \leq 2 \times \text{DL-Adv}[\mathcal{B}, \mathbb{G}]$, where $\text{DL-Adv}[\mathcal{B}, \mathbb{G}]$ is the probability that \mathcal{B} computes discrete logarithms in \mathbb{G} (formally defined in [36]).

Due to lack of space, we refer to [26] for the proofs of correctness and security of \mathcal{H} -DL.

C. Detection Guarantees

Using the downloaded hashes, all nodes in the network can verify the integrity of all packets on-the-fly. The following theorem summarizes the security guarantee of our hash-based detection scheme.

Theorem 2. If a secure homomorphic hash scheme and a secure traditional hash function is used in the detection scheme, then the probability of a benign node accepting a corrupted packet is negligible in the security parameter.

We refer the reader to [26] for the proof. Finally, our hash-based detection scheme is collusion-resistant because collusion does not help to break the discrete log assumption or a secure traditional hash function.

V. MAC-BASED DETECTION

A. Key Observation

Observation 3. Let us revisit the discussion of applying homomorphic MAC scheme in Section III-C. From the attack, we observe that it is necessary that (i) each source generates tags of its packets using its own secret key as opposed to using a common key, or (ii) the controller generates all the tags under a key secret to all the sources.

B. Homomorphic Multi-Source MAC (InterMac)

In this section, we present a novel multi-source homomorphic MAC scheme, called InterMac, that allows different sources to generate tags using different keys. Nonetheless, the tags are combinable, and the malicious nodes cannot generate valid tags of corrupted packets.

Definitions: A (q, n, s, g) multi-source homomorphic MAC scheme is defined by four PPT algorithms:

- $\text{Generate}(\text{id}, k, \Pi)$: Input: a source space identifier, id , a secret key, $k \in \mathcal{K}$, and a committed source space, Π . Output: a key set $\mathcal{K} = \{k_1, \dots, k_s\}$.
- $\text{Sign}(k_i, \mathbf{v})$: Input: a key $k_i \in \mathcal{K}$ used by source S_i and a source packet \mathbf{v} sent by source S_i . Output: tag t of \mathbf{v} .
- $\text{Combine}((\mathbf{y}_1, t_1, \alpha_1), \dots, (\mathbf{y}_\ell, t_\ell, \alpha_\ell))$: Input: ℓ vectors, $\mathbf{y}_1, \dots, \mathbf{y}_\ell \in \mathbb{F}_q^{n+m}$; their tags, $t_1, \dots, t_\ell \in \mathbb{F}_q$; and coefficients, $\alpha_1, \dots, \alpha_\ell \in \mathbb{F}_q$. Output: tag t of $\mathbf{y} \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} \alpha_i \mathbf{y}_i$.
- $\text{Verify}(\mathcal{K}, \mathbf{y}, t)$: Input: a key set, \mathcal{K} , a vector, $\mathbf{y} \in \mathbb{F}_q^{n+m}$, and its tag, $t \in \mathbb{F}_q$. Output: 0 (reject) or 1 (accept).

Correctness: The scheme must satisfy the following correctness requirement: Let $t_{i,j} = \text{Sign}(k_i, \mathbf{v}_{i,j})$ and $\alpha_{i,j} \in \mathbb{F}_q$, for all $i \in [1, s]$ and $j \in [1, g]$. Let $t = \text{Combine}((\mathbf{v}_{1,1}, t_{1,1}, \alpha_{1,1}), \dots, (\mathbf{v}_{s,g}, t_{s,g}, \alpha_{s,g}))$. Then $\text{Verify}(\mathcal{K}, \sum_{i=1}^s \sum_{j=1}^g \alpha_{i,j} \mathbf{v}_{i,j}, t) = 1$.

Security: We define security using the following game:

Attack Game. For $\mathcal{T} = (\text{Generate}, \text{Mac}, \text{Combine}, \text{Verify})$, denote the challenger and adversary by \mathcal{C} and \mathcal{A} , respectively:

- *Setup:* \mathcal{C} generates a random key $k \xleftarrow{R} \mathcal{K}$.
- *Queries:* \mathcal{A} adaptively queries \mathcal{C} . Each query is of the form (id_l, Π_l) , where Π_l is a linear subspace represented by a basis of m vectors, $\mathbf{v}_{i,j}$, $i \in [1, s]$, $j \in [1, g]$, and id_l is the space identifier. We require that all identifiers id_l submitted by \mathcal{A} be distinct. To respond to a query (id_l, Π_l) , \mathcal{C} runs $\text{Generate}(\text{id}_l, k, \Pi_l)$ to produce a key set $\mathcal{K}_l = \{k_1, \dots, k_s\}$, computes $t_{i,j} = \text{Sign}(k_i, \mathbf{v}_{i,j})$, and sends $(t_{1,1}, \dots, t_{s,g})$ and all keys in \mathcal{K}_l but one to \mathcal{A} .
- *Output:* \mathcal{A} outputs a triplet $(\text{id}_*, \mathbf{y}_*, t_*)$. \mathcal{A} wins the attack game if (i) $\text{id}_* = \text{id}_l$ for some l , (ii) $\mathbf{y}_* \notin \Pi_l$, and (iii) $\text{Verify}(\mathcal{K}_l, \mathbf{y}_*, t_*) = 1$.

Requirement (i) is necessary as a corrupted packet is only defined when there is a committed source space. Requirement (ii) indicates that the output packet by \mathcal{A} is indeed a corrupted packet. Finally, (iii) indicates that \mathcal{A} successfully forges a valid tag of the corrupted packet. Let $\text{Adv}[\mathcal{A}, \mathcal{T}]$ denote the probability that \mathcal{A} wins the above attack game. We define a secure multi-source homomorphic MAC scheme as follows:

Definition 3. A (q, n, s, g) multi-source homomorphic MAC scheme \mathcal{T} is secure if and only if for all PPT adversaries \mathcal{A} , $\text{Adv}[\mathcal{A}, \mathcal{T}]$ is negligible.

The Construction of InterMac. The key ingredient of this construction is the generation of the key set \mathcal{K} so that each source can compute tags of its source packets using its own key; nonetheless, the tags are still combinable.

- $\text{Generate}(\text{id}, k, \Pi)$:
 - Let $\mathbf{v}_{1,1}, \dots, \mathbf{v}_{s,g} \in \mathbb{F}_q^{n+m}$ be the committed source packets that span Π and are represented as row vectors. For each $p \in [1, s]$, let M_p be a matrix whose rows are vectors in the following set

$$\{\mathbf{v}_{i,j} \mid i = 1, \dots, s; i \neq p; j = 1, \dots, g\}.$$

In other words, M_p is a matrix consisting of source packets of all sources but S_p . Note that $\text{rank}(M_p) = m - g$. Let Π_{M_p} denote the space spanned by the rows of M_p .

- The null space of the matrix M_p , denoted as $\Pi_{M_p}^\perp$, is the set of all row vectors $\mathbf{z} \in \mathbb{F}_q^{n+m}$ for which $M_p \mathbf{z}^T = \mathbf{0}$. For any $(m - g) \times (n + m)$ matrix M_p , we have

$$\text{rank}(M_p) + \text{nullity}(M_p) = n + m$$

known as rank-nullity theorem, where $\text{nullity}(M_p)$ is the dimension of $\Pi_{M_p}^\perp$. Thus,

$$\dim(\Pi_{M_p}^\perp) = n + m - (m - g) = n + g.$$

- Let $\mathbf{b}_1, \dots, \mathbf{b}_{n+g} \in \mathbb{F}_q^{n+m}$ be a basis of $\Pi_{M_p}^\perp$. This basis can be found by solving $M_p \mathbf{z}^T = \mathbf{0}$. Let F be a Pseudo Random Function (PRF): $\mathcal{K} \times (\mathcal{I} \times [1, s] \times [1, n + g]) \rightarrow \mathbb{F}_q$, where \mathcal{I} denotes the domain of the source space identifier. To generate key k_p for source S_p , the controller computes

- $r_i \leftarrow F(k, \text{id}, p, i) \in \mathbb{F}_q, \forall i \in [1, n + g]$.
- $k_p \leftarrow \sum_{i=1}^{n+g} r_i \mathbf{b}_i \in \mathbb{F}_q^{n+m}$.

- **Output:** a key set $\mathcal{K} \stackrel{\text{def}}{=} \{k_1, \dots, k_s\}$, where each key, k_p , $p \in [1, s]$, is generated as above.

- $\text{Sign}(k_i, \mathbf{v})$: Outputs $t \leftarrow k_i \cdot \mathbf{v} \in \mathbb{F}_q$.

- $\text{Combine}((\mathbf{y}_1, t_1, \alpha_1), \dots, (\mathbf{y}_\ell, t_\ell, \alpha_\ell))$: Outputs the sum $t \leftarrow \sum_{i=1}^\ell \alpha_i t_i \in \mathbb{F}_q$.
- $\text{Verify}(\mathcal{K}, \mathbf{y}, t)$: Compute $t' = \mathbf{y} \cdot (k_1 + \dots + k_s)$, where $k_i \in \mathcal{K}$. If $t = t'$, output 1; otherwise, output 0.

Correctness: Recall from the correctness requirement that

$$t = \sum_{i=1}^s \sum_{j=1}^g \alpha_{i,j} t_{i,j} = \sum_{i=1}^s \sum_{j=1}^g \alpha_{i,j} (\mathbf{v}_{i,j} \cdot k_i).$$

Also, t' computed by the verification algorithm equals

$$\sum_{i=1}^s \sum_{j=1}^g \alpha_{i,j} \mathbf{v}_{i,j} \cdot (k_1 + \dots + k_s) \stackrel{(1)}{=} \sum_{i=1}^s \sum_{j=1}^g \alpha_{i,j} (\mathbf{v}_{i,j} \cdot k_i).$$

Equality (1) is because by construction, for all $i \neq p$, $i \in [1, s]$, $p \in [1, s]$, and $j \in [1, g]$, $\mathbf{v}_{i,j} \cdot k_p = 0$. As computed, $t' = t$.

Security: We prove the security of InterMac assuming F is a secure PRF. For a PRF adversary \mathcal{B} , we let $\text{PRF-Adv}[\mathcal{B}, F]$ denote \mathcal{B} 's advantage in winning the PRF security game w.r.t. F . The definition of the PRF security game is provided in [36].

Theorem 3. For any fixed q, n, s, g , InterMac is a secure (q, n, s, g) multi-source homomorphic MAC, assuming F is a secure PRF. In particular, for every multi-source homomorphic MAC adversary \mathcal{A} , there is a PRF adversary \mathcal{B} who has similar running time to \mathcal{A} , such that $\text{Adv}[\mathcal{A}, \text{InterMac}] \leq \text{PRF-Adv}[\mathcal{B}, F] + \frac{1}{q}$.

Proof: The proof is by using a sequence of games denoted as Game 0 and 1. Let W_0 and W_1 denote the events that \mathcal{A} wins the multi-source homomorphic MAC security in Game 0 and 1, respectively. Let Game 0 be identical to the Attack Game. Hence, $\Pr[W_0] = \text{Adv}[\mathcal{A}, \text{InterMac}]$ (Eq.1). In Game 1, the PRF F is replaced by a truly random function, i.e., to respond to the queries, the challenger computes $k_p = \sum_{i=1}^{n+g} r_i \mathbf{x}_i$, where $r_i \xleftarrow{R} \mathbb{F}_q$. Everything else remains the same. Then, there exists a PRF adversary \mathcal{B} such that $|\Pr[W_0] - \Pr[W_1]| = \text{PRF-Adv}[\mathcal{B}, F]$ (Eq.2). The complete challenger in Game 1 works as follows:

- *Queries:* \mathcal{A} submits MAC queries (id, Π) , where $\Pi = \text{span}(\mathbf{v}_{1,1}, \dots, \mathbf{v}_{s,g})$. For each $p \in [1, s]$, \mathcal{C} computes a basis of $\Pi_{M_p}^\perp: \mathbf{x}_1, \dots, \mathbf{x}_{n+g}$. Then, in order to generate k_p , \mathcal{C} computes $r_i \xleftarrow{R} \mathbb{F}_q, \forall i \in [1, n+g]$ and then $k_p \leftarrow \sum_{i=1}^{n+g} r_i \mathbf{x}_i \in \mathbb{F}_q^{n+m}$. In other words, each k_p is chosen uniformly at random from $\Pi_{M_p}^\perp$, a subspace of size q^{n+g} . The challenger \mathcal{C} then computes tags for the committed source packets: $t_{i,j} \leftarrow k_i \cdot \mathbf{v}_{i,j}, i \in [1, s], j \in [1, g]$. Finally, \mathcal{C} sends all the tags and all the keys but one to \mathcal{A} . W.l.o.g., assume that \mathcal{C} keeps k_1 secret to \mathcal{A} .

- *Output.* \mathcal{A} eventually outputs a triplet $(\text{id}_*, \mathbf{y}_*, t_*)$. Assume that $\text{id}_* = \text{id}_l$, for some l . Let $\mathcal{K}_l = \{k_1, \dots, k_s\}$ denote the key set generated for query (id_l, Π_l) . \mathcal{A} wins the game, i.e., W_1 happens, if $\mathbf{y}_* \notin \Pi_l$ and $t_* = \mathbf{y}_* \cdot (k_1 + \dots + k_s)$. Note that \mathcal{A} knows k_2, \dots, k_s , therefore, if $\mathbf{y}_* \cdot k_1$ is known, \mathcal{A} will be able to forge a valid t_* . In what follows, we will show that $\mathbf{y}_* \cdot k_1$ is indistinguishable from a random value in \mathbb{F}_q .

Let $\Pi_l = \text{span}(\mathbf{v}_{1,1}, \dots, \mathbf{v}_{s,g})$. Consider the following system of linear equations: $\mathbf{v}_{1,1} \cdot k_1 = t_{1,1}; \dots; \mathbf{v}_{1,g} \cdot k_1 = t_{1,g}; \mathbf{v}_{2,1} \cdot k_1 = 0; \dots; \mathbf{v}_{s,g} \cdot k_1 = 0; \mathbf{y}_* \cdot k_1 = t_* - \mathbf{y}_* \cdot (k_2 + \dots + k_s)$. All but the last equation represent all information that \mathcal{A} learns about k_1 from its query (id_l, Π_l) .

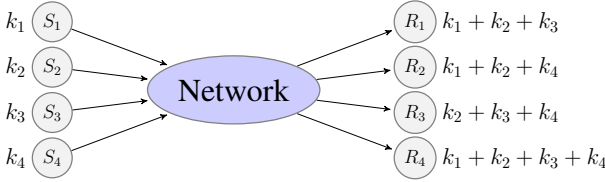


Fig. 3. An example demonstrating a sufficient amount of information for carrying out the verification at each receiver when using InterMac. In this setting, R_1 and R_2 only receive linear combinations of source packets sent by S_1 and S_2 ; R_3 only receives combinations of source packets sent by S_2 and S_3 ; R_4 receives linear combinations of all source packets; the maximum number of malicious sources is $M = 2$.

Since $\mathbf{y}_* \notin \Pi_l$, \mathbf{y}_* and $\mathbf{v}_{i,j}$'s are linearly independent. Hence, the above system of equations is consistent regardless of the value of t_* because the coefficient matrix has rank $sg + 1$, that equals the number of equations. Furthermore, for a fixed \mathbf{y}_* , for any value $t_* \in \mathbb{F}_q$, the solution space always has the same size $q^{n+sg-(sg+1)} = q^{n-1}$. Because k_1 is chosen uniformly at random from $\Pi_{M_1}^\perp$, and all solutions to the above system of equations are in $\Pi_{M_1}^\perp$, for a fixed \mathbf{y}_* , its valid tag t_* could be any value in \mathbb{F}_q equally likely. Thus, the probability the adversary chooses a correct t_* is $\Pr[W_1] = \frac{1}{q}$ (Eq.3).

(Eq.1), (Eq.2), and (Eq.3) together prove the theorem. ■

Theorem 3 expresses that an adversary \mathcal{A} can only forge a valid tag of a corrupted packet with probability $\frac{1}{q}$. When using ℓ tags, the security is $\frac{1}{q^\ell}$.

Remarks. When using InterMac, a node only needs to know the sum of the keys for the verification, and when there is an upper bound M on the number of possible malicious sources, it may suffice for a verifying node to know the sum of just $M + 1$ keys to carry out the verification. An example is given in Fig. 3. In Fig. 3, the reason why $(k_1 + k_2 + k_3)$ is sufficient for R_1 to verify a packet \mathbf{y} is twofold: (i) if \mathbf{y} is a benign packet, $k_4 \cdot \mathbf{y} = 0$ as $k_4 \in \Pi_{M_4}^\perp$; thus, $\mathbf{y} \cdot (k_1 + \dots + k_4) = \mathbf{y} \cdot (k_1 + \dots + k_3)$. As a result, R_1 does not need to know k_4 to verify a valid packet. (ii) If \mathbf{y} is corrupted, since there is at least one key secret to the adversary ($M = 2$), we can use the same line of arguments as in the proof of Theorem 3 to show that the probability of forging a valid MAC tag for \mathbf{y} is only $\frac{1}{q}$. Showing that the other sums are sufficient for R_2 , R_3 , and R_4 can be done with similar arguments.

C. Efficient Commitment

The role of the committed source packets in InterMac is to enable the controller to generate MAC keys orthogonal to the matrices Π_{M_p} 's. Here, we design a more efficient commitment scheme that does not require each source to send all their source packets to the controller, but it still allows the controller to generate these orthogonal vectors. To this end, we leverage two key techniques: padding for orthogonality [20] and private inner product computation [31]. We use padding to make a random vector chosen by the controller, that will serve as a MAC key, orthogonal to the Π_{M_p} 's. We use the private inner product protocol to allow the controller to compute the padding elements while keeping the random chosen vector private.

Private Inner Product Protocol. Let $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$ be a semantically secure homomorphic public-key cryptosystem. In general, the private inner product protocol (PIP) proposed

PRIVATE INPUTS: Private vectors $\mathbf{r}, \mathbf{v} \in \mathbb{F}_q^n$. PRIVATE OUTPUTS: Inner product $\mathbf{r} \cdot \mathbf{v} \in \mathbb{F}_q$. • C generates a key pair (sk, pk) and sends pk to S . • C sends $c_i = \text{Enc}_{\text{pk}}(r_i)$ to S , $\forall i \in [1, n]$. • S sends $w \leftarrow \prod_{i=1}^n c_i^{v_i}$ to C . • C computes $\mathbf{r} \cdot \mathbf{v} = \text{Dec}_{\text{sk}}(w)$.
--

TABLE I
PRIVATE INNER PRODUCT (PIP) PROTOCOL

in [31] works with various public-key cryptosystems [37]–[39] that have the following homomorphic properties:

- $\text{Dec}(\text{Enc}(m_1) \text{Enc}(m_2)) = m_1 + m_2$, and
- $\text{Dec}(\text{Enc}(m_1)^{m_2}) = m_1 m_2$.

We choose Benaloh system [39] in our instantiation of the PIP protocol. Let q be prime, $\mathbf{r} = (r_1, \dots, r_n)$ be a random vector chosen by the controller C , and $\mathbf{v} = (v_1, \dots, v_n)$ be a source vector of source S . C and S carry out the PIP protocol described in Table I. With PIP, C can learn the inner product $\mathbf{r} \cdot \mathbf{v}$ while S does not learn any information about \mathbf{r} , thanks to the security guarantee of the encryption.

Commitment, Padding, and Key Generation (CPK) Protocol. Let $k \in \mathcal{K}$ and F be a PRF: $\mathcal{K} \times (\mathcal{I} \times [1, s] \times [n + s - 1 + m]) \rightarrow \mathbb{F}_q$. Each source packet will be padded with $s - 1$ elements. Using PIP, the controller C generates the MAC keys and computes the padding as follows:

1. *Setup:* Let id be the subspace identifier. For $i \in [1, s]$ and $j \in [1, n + s - 1 + m]$, C computes $r_i^{(j)} \leftarrow F(k, \text{id}, i, j)$. Let $\mathbf{r}_i = (r_i^{(1)}, \dots, r_i^{(n+s-1+m)})$ and $\hat{\mathbf{r}}_i = (r_i^{(1)}, \dots, r_i^{(n)})$.
2. *Commitment:* For each $i \in [1, s]$, C and $S_{i'}$, $i' \in [1, s] \setminus \{i\}$, carry out the PIP protocol so that C learns $\hat{\mathbf{r}}_i \cdot \hat{\mathbf{v}}_{i',j}$, $\forall j \in [1, g]$. The encryption of these dot products sent from each source to the controller in the PIP protocol represent the commitment made by the sources.
3. *Padding:* Let $p_{i,j}^{(1)}, \dots, p_{i,j}^{(s-1)}$ denote the padding elements for a source packet $\mathbf{v}_{i,j}$ sent by source S_i . The padded source packet, denoted by $\mathbf{p}_{i,j}$, has the following form:

$$(\hat{\mathbf{v}}_{i,j}, p_{i,j}^{(1)}, \dots, p_{i,j}^{(s-1)}, \underbrace{0, \dots, 0, 1, 0, \dots, 0}_m) \in \mathbb{F}_q^{n+s-1+m}$$

The padding elements are computed by solving the following system of $s - 1$ linear equations:

$$\{\mathbf{r}_{i'} \cdot \mathbf{p}_{i,j} = 0\}_{i' \in [1, s] \setminus \{i\}} \quad (1)$$

- For $s > 1$, this system has $s - 1$ unknowns and consists of $s - 1$ linearly independent equations. Therefore, there is a unique solution for $p_{i,j}^{(1)}, \dots, p_{i,j}^{(s-1)}$. C then sends the padding elements to S_i . S_i now sends $\mathbf{p}_{i,j}$ instead of $\mathbf{v}_{i,j}$.
4. *MAC keys:* C uses \mathbf{r}_i as MAC key k_i . Equations in (1) indicate that the chosen key k_i is orthogonal to Π_{M_i} .

When using the CPK protocol, the sources no longer need to send all of their source packets to the controller. Instead, they only need to send an encryption of the inner product for every source packet, thereby significantly reducing the communication cost. Fig. 4 illustrates the CPK protocol for the network of Fig. 1. We use $\text{InterMac}_{\text{CPK}}$ to denote the InterMac construction when using the CPK protocol to generate MAC keys instead of Generate.

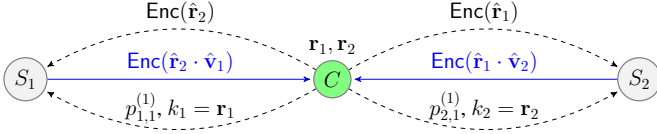


Fig. 4. Keys generation using the CPK protocol for the network of Fig. 1. k_1 is orthogonal to the padded vector $\mathbf{p}_{2,1}$ and k_2 is orthogonal to $\mathbf{p}_{1,1}$ thanks to the padding. At the same time, k_1 is secret to S_2 and k_2 is secret to S_1 .

Security. The security of $\text{InterMac}_{\text{CPK}}$ in the *semi-honest model*, where it is assumed that both parties follow the protocol but they are curious and try to deduce information from all exchanged data, comes from the security of PIP and InterMac . Due to lack of space, the formal security proof is provided in the technical report [26]. In the *malicious model*, where malicious sources may not follow the protocol, the security guarantee of $\text{InterMac}_{\text{CPK}}$ could still be achieved by adding appropriate controller's responses for malicious behaviors. Malicious behaviors of the sources are limited to (i) not sending a well-formed encryption back for each query of C , and (ii) not padding the source packets appropriately. For (i), the controller could exclude any source with this behavior from the source list and only calculate MAC keys for the remaining sources. For (ii), not-properly padded packets will be dropped with high probability as they are highly likely to be outside of the committed source space.

D. Private Inner Product MAC

In this section, we explore the second direction of Observation 3, which suggests that all tags of the source packets be generated by the trusted controller, and the MAC key be secret to the sources. In particular, we show how the PIP protocol could be combined with SpaceMac previously proposed for intra-session network coding [25] to provide an alternative MAC-based scheme for detecting corrupted packets.

SpaceMac consists of a triplet of algorithms: Mac , Combine , and Verify . The construction of SpaceMac uses a PRF $F : \mathcal{K} \times (\mathcal{I} \times [1, n+m]) \rightarrow \mathbb{F}_q$ and is as follows:

- $\text{Mac}(k, \text{id}, \mathbf{y})$: The tag $t \in \mathbb{F}_q$ of an input vector $\mathbf{y} \in \mathbb{F}_q^{n+m}$ is computed by the following steps:
 - $\mathbf{r} \leftarrow (F(k, \text{id}, 1), \dots, F(k, \text{id}, n+m))$.
 - $t \leftarrow \mathbf{y} \cdot \mathbf{r} \in \mathbb{F}_q$.
- $\text{Combine}((\mathbf{y}_1, t_1, \alpha_1), \dots, (\mathbf{y}_\ell, t_\ell, \alpha_\ell))$: The tag $t \in \mathbb{F}_q$ of $\mathbf{y} \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} \alpha_i \mathbf{y}_i \in \mathbb{F}_q^{n+m}$ is computed as follows:
 - $t \leftarrow \sum_{i=1}^{\ell} \alpha_i t_i \in \mathbb{F}_q$.
- $\text{Verify}(k, \text{id}, \mathbf{y}, t)$: To verify if t is a valid tag of \mathbf{y} using key k , we do the following:
 - $\mathbf{r} \leftarrow (F(k, \text{id}, 1), \dots, F(k, \text{id}, n+m))$.
 - $t' \leftarrow \mathbf{y} \cdot \mathbf{r}$.
 - If $t' = t$, output 1 (accept); otherwise, output 0 (reject).

Private MAC (PM) Protocol. The controller and the sources carry the PM protocol to compute tags of the source packets. The PM protocol consists of the following steps:

1. *Setup*: Let id be the current subspace identifier. C computes $r^{(i)} \leftarrow (F(k, \text{id}, i), \forall i \in [1, n+m])$. Let $\hat{\mathbf{r}} = (r^{(1)}, \dots, r^{(n)})$ and $\mathbf{r} = (r^{(1)}, \dots, r^{(n+m)})$.
2. *Commitment*: For each $i \in [1, s]$, C and S_i carry out the PIP protocol that allows C to learn $\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}_{i,j}, \forall j \in [1, g]$. The

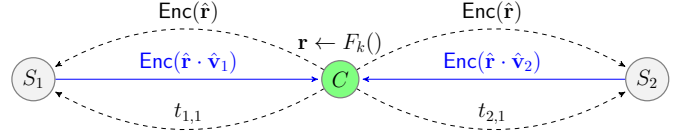


Fig. 5. Tags generation using the PM protocol for the network of Fig. 1. The tags are generated by the controller, and the key k is secret to the sources.

encryption of the inner products sent by the sources to the controller are the commitment.

3. *MAC tags*: For $\mathbf{v}_{i,j}$, C computes its tag $t_{i,j} = \mathbf{r} \cdot \mathbf{v}_{i,j} = \hat{\mathbf{r}} \cdot \hat{\mathbf{v}}_{i,j} + r^{(g(i-1)+j)}$.

PM helps the controller compute the tags on behalf of the sources without leaking the MAC key. Fig. 5 illustrates how the $\text{SpaceMac}_{\text{PM}}$ MAC tags are computed for the network of Fig. 1. We use $\text{SpaceMac}_{\text{PM}}$ to denote the SpaceMac scheme when used with the PM protocol to generate tags for the source packets as opposed to the Mac algorithm.

Security. The security of $\text{SpaceMac}_{\text{PM}}$ in the semi-honest model comes from the security of PIP and SpaceMac . The formal security proof is provided in [26]. In the malicious model, a malicious source S_i is limited to not sending back an encryption of the inner product of \mathbf{r} and the appropriate $\mathbf{v}_{i,j}$ or sending back a mal-form encryption. In response to these behaviors, the controller can ignore $\mathbf{v}_{i,j}$ in its tag computation and thus, do not send the tag of $\mathbf{v}_{i,j}$ back to S_i . The source S_i , without knowing the key, k , will not be able to generate a valid tag for $\mathbf{v}_{i,j}$.

Comparison. Compared to $\text{InterMac}_{\text{CPK}}$, $\text{SpaceMac}_{\text{PM}}$ is simpler in terms of initialization. This is because $\text{InterMac}_{\text{CPK}}$ operates on s MAC keys instead of one key. $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$ have similar efficient Combine and Verify operations as both of them only involve simple field addition and multiplication as opposed to exponentiation. When using $\text{SpaceMac}_{\text{PM}}$, all receivers must know the MAC key k in order to verify their received packets. As a result, as soon as an adversary compromises a receiver and learns k , it can fool all other receivers into accepting corrupted packets. We stress that this is not necessarily the case when using $\text{InterMac}_{\text{CPK}}$. For instance, consider Fig. 3. Assume that S_1 and S_2 are malicious, thus keys k_1 and k_2 are leaked. If the adversary compromises R_1 , it learns k_3 by subtracting the sum $(k_1 + k_2 + k_3)$ from $(k_1 + k_2)$. However, it still cannot fool R_2 , R_3 , or R_4 into accepting a corrupted packet as the verification at these receivers involves k_4 , that is still secret to the adversary.

$\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$, as described, could be used as a drop-in replacement for traditional MACs, e.g., HMAC, for networks that use inter-session network coding – they allow the receivers to detect corrupted packets. As when using a traditional MAC scheme, we assume the keys distribution is through secure (athentic and private) channels. We also assume the communication between the sources and the controllers in the CPK and PM protocols is through authentic channels.

E. In-Network Detection

Both our MAC schemes currently provide end-to-end detection. However, if combined with a mechanism for broadcast

authentication, they could be extended to provide in-network detection. We discuss two such options below:

Delayed Key Disclosure (TESLA) [40]: This approach leverages the time dimension to achieve broadcast authentication and has been adapted to intra-session network coding to provide in-network detection [11], [18], [24]. In this approach, nodes are required to loosely synchronize their time. Both $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$ could be used with the approaches proposed in [18] and [24] to provide in-network detection for fixed directed acyclic networks and dynamic peer-to-peer networks, respectively. We note that the detection schemes based on [18] and [24] are fully collusion-resistant and tag-pollution-resistant (an attack on schemes that use multiple MAC tags [18]).

Cover-Free Set Systems [41]: This approach leverages cover-free set systems to probabilistically distribute keys to all nodes such that any collusion of c nodes or less does not leak all the keys used in the whole system. This approach has been adapted to intra-session network coding to provide in-network detection [20], [27]. Both $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$ are suitable to be used with this approach. Detection schemes based on [20], [27] are c -collusion-resistant. To address tag pollution, we propose using our homomorphic hash-based detection scheme to protect the coding coefficients and the tags of the packets.

VI. PERFORMANCE EVALUATION

A. Bandwidth Overhead

1) *Hash-Based Detection:* Our hash-based scheme does not incur any online bandwidth overhead per packet as there is no additional symbol attached to each packet sent in the network. The off-line bandwidth overhead of this scheme is dominated by the bandwidth required to distribute both the homomorphic and traditional hashes from the controller to all the nodes. The size of a homomorphic hash is $|q|$. Let $|\bar{h}|$ denote the size of the traditional hash (for SHA-1, $|\bar{h}|=160$ bits). The total off-line bandwidth overhead is $sg|\mathcal{V}|(|\bar{h}| + |q|)$.

2) *MAC-Based Detection:* The off-line bandwidth overhead of $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$ come from the packets exchanged during the execution of the CPK and PM protocols. The off-line bandwidth overhead of $\text{InterMac}_{\text{CPK}}$ includes the overhead of the encryptions of the randomly chosen vectors sent by the controller, the encryptions of the inner products sent back by the sources, and the padding sent by the controller, which is $s(s-1)(ne|q| + ge|q|) + sg(s-1)|q|$, where e is the expansion factor of the encryption scheme and equals $\frac{N}{|q|}$ (N is the size of the modulo of the encryption in bits). The off-line bandwidth overhead of $\text{SpaceMac}_{\text{PM}}$ includes the overhead of the encryption of the randomly chosen vector and the encryptions of the inner products, which is $s(ne|q| + ge|q|)$. To be concrete, for $N = 256$, $n = 1024$, $s = 5$, and $g = 100$, the off-line bandwidth overhead *per source packet* of $\text{InterMac}_{\text{CPK}}$ and $\text{SpaceMac}_{\text{PM}}$ range from 36% to 1% as the field size increases from 32 to 256 bits. Using CPK or PM could help to save more than 90% of the commitment bandwidth. We refer the reader to [26] for more details.

The online overhead comes from the tags accompanied with each packet. To provide end-to-end detection, using a single tag suffices. In this case, the overhead of both $\text{InterMac}_{\text{CPK}}$

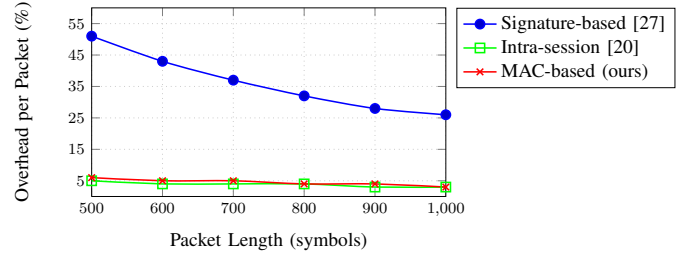


Fig. 6. Per-packet bandwidth overhead of the homomorphic signature scheme [27], the hybrid intra-session scheme [20] ($c = 1$, $\delta = 0.1$, $\epsilon = 0.01$), and our $\text{InterMac}_{\text{CPK}}/\text{SpaceMac}_{\text{PM}}$ when used with RIPPLE [18].

and $\text{SpaceMac}_{\text{PM}}$ is $\frac{|q|}{n|q|}$ (0.1% for $n = 1024$). To provide in-network detection for a directed acyclic network, let one of our MAC schemes be used with the delayed key disclosure technique in RIPPLE [18]. Let S_0 be a virtual node that has an edge pointing toward every source node. Define a level of a node as the length of the longest path from S_0 to the node. Let L be the maximum among the levels of the nodes. Each packet carries L MAC tags initially; then one or more tags are peeled off at every node the packet goes through. The average online overhead per packet is $\frac{L|q|}{2n|q|}$ %.

In comparison, on average, the online overhead per packet of [27] is $\frac{s(g|q| + |\sigma|)}{2n|q|}$ %, where $|\sigma|$ is the size of a regular public key signature. We stress that this overhead depends on the number of source packets whereas ours does not. To be concrete, if we set $L = 16$ (as in [18]), $|\sigma| = 320$ (DSA), $|q| = 128$, $s = 5$, $g = 100$, then the overhead per packet of [27] is 32 times larger than ours ($\frac{s(g|q| + |\sigma|)}{L|q|} \simeq 32$). Fig. 6 plots the average online overhead per packet of [27], a state-of-the-art intra-session detection scheme [20], and our MAC-based scheme as a function of packet length. The range of the packet length is chosen according to [20] for ease of comparison. This plot shows that not only is our overhead *one order of magnitude smaller* than that of [27], but it is also small, as small as 3%. Our overhead is comparable to that of [20].

B. Computational Overhead

We focus on the online overhead introduced by the operation performed at each node per packet and we neglect other overhead, *e.g.*, computing the hashes and MAC keys, as these are negligible in the number of packets in the network. Similar to [20], we calculate the computation overhead by approximating various operations by the number of finite field multiplications. To calculate the computation time, for ease of comparison, we adopt the benchmark obtained in [20] on a 2.0 GHz Intel Core 2 CPU, where approximately 2.5×10^5 multiplications can be performed per second for $|q| = 128$.

1) *Hash-Based Detection:* For each packet, the worst case scenario is that the node needs to perform a homomorphic hash check, *i.e.*, performing the Test algorithm of \mathcal{H} -DL. This algorithm entails $n + m$ modular exponentiations (recall $m = sg$). In comparison, in the worst case, the scheme in [27] requires $n + m$ exponentiations plus s public-key signature verifications. In the best scenario, where the received packet is decodable, our scheme requires just one traditional hash check.

2) *MAC-Based Detection:* Let one of our MAC schemes be used with RIPPLE as described in Section VI-A. For each

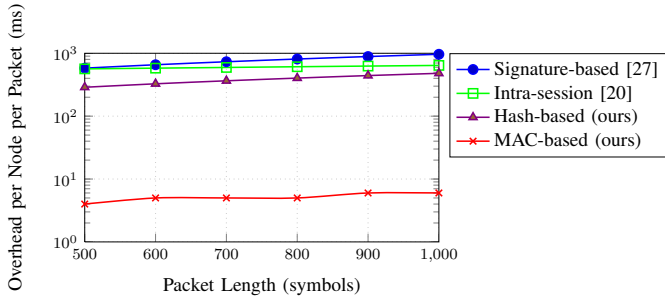


Fig. 7. Per-packet per-node computation overhead of the signature scheme [27], the intra-session scheme [20] ($c = 1, \delta = 0.1, \epsilon = 0.01$), our hash-based scheme, and InterMac_{CPK}/SpaceMac_{PM} when used with RIPPLE [18].

packet, the overhead includes one Combine (to generate the tag of the packet) and one Verify (to verify the integrity of the packet). Let w be the average number of packets combined by each node. Then, on average, the Combine algorithm entails $w(\frac{L-1}{2})$ multiplications. Meanwhile, the Verify algorithm entails $n + m + \frac{L-1}{2}$ multiplications. The total average overhead is $w(\frac{L-1}{2}) + (n + m + \frac{L-1}{2})$ multiplications.

In comparison, the average overhead of [27] is $n + \frac{sg}{2}$ exponentiations plus $\frac{s}{2}$ public-key verification. We approximate the cost of one public-key verification (DSA) by two modular exponentiations. Utilizing the “square and multiple” method, each exponentiation over \mathbb{F}_q takes approximately $\frac{3}{2}|q|$ multiplications on average [20]. The total average overhead is $\frac{3}{2}|q|(n + \frac{sg}{2} + s)$ field multiplications.

To be concrete, let $L = 16, w = 4, n = 1024, s = 5, g = 100$, and $|q| = 128$. We approximate a traditional hash check by 80 field multiplications (1 per iteration of SHA-1) and let the decodable probability be 50%. Fig. 7 plots the average online computation overhead per packet per node of the signature-based scheme [27], the intra-session detection scheme [20], and our hash- and MAC-based schemes. This plot shows that the overhead of our hash-based scheme is half of that of [27]. (The computation efficiency would increase with the decodable probability.) It also demonstrates that the overhead of our MAC-based scheme is small, ranging from 4 to 6 ms, and is *two orders of magnitude less* than those of [27] and [20].

VII. CONCLUSION

In this paper, we introduce three efficient schemes for detection of pollution attacks in inter-session network coding. The main idea behind our schemes is the use of commitment of source packets to a trusted controller. The first scheme is a novel combination of homomorphic and traditional hash functions. The other two schemes are novel MAC schemes for inter-session network coding: InterMac_{CPK} and SpaceMac_{PM}. InterMac_{CPK} is the first multi-source homomorphic MAC scheme that supports multiple keys. The proposed schemes provide alternative ways for detection of pollution attacks in inter-session coding. Among the three, we recommend using the MAC schemes since they have significantly lower computation overhead. Finally, we recommend using InterMac_{CPK} over SpaceMac_{PM} when there may exist malicious receivers.

VIII. ACKNOWLEDGMENT

We would like to thank Prof. Gene Tsudik at UC Irvine for insightful discussions on the InterMac scheme.

REFERENCES

- [1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, “XORs in the Air: Practical Wireless Network Coding,” in *SIGCOMM’06*, 2006.
- [2] S. Omiwade, R. Zheng, and C. Hua, “Butterflies in the Mesh: Lightweight Localized Wireless Network Coding,” in *NetCod’08*, 2008.
- [3] Y. Feng, Z. Liu, and B. Li, “GestureFlow : Streaming Gestures to an Audience,” in *IEEE INFOCOM’11*, 2011.
- [4] N. Cai and R. W. Yeung, “Secure Network Coding,” in *ISIT’02*, 2002.
- [5] Z. Zhang, “Network Error Correction Coding in Packetized Networks,” in *Info Theory Workshop*, 2006.
- [6] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, “Resilient Network Coding in the Presence of Byzantine Adversaries,” in *INFOCOM’07*.
- [7] R. Koetter and F. R. Kschischang, “Coding for Errors and Erasures in Random Network Coding,” in *ISIT’07*, 2007.
- [8] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. R. Karger, “Byzantine Modification Detection in Multicast Networks using Randomized Network Coding,” in *ISIT’04*, 2004.
- [9] E. Kechdi and B. Li, “Null Keys : Limiting Malicious Attacks Via Null Space Properties of Network Coding,” in *INFOCOM’09*, 2009.
- [10] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, “An Efficient Scheme for Securing XOR Network Coding against Pollution Attacks,” in *INFOCOM’09*, 2009.
- [11] J. Dong, R. Curtmola, and C. Nita-Rotaru, “Practical Defenses Against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks,” in *WiSec’09*.
- [12] C. Gkantsidis and P. R. Rodriguez, “Cooperative Security for Network Coding File Distribution,” in *INFOCOM’06*, 2006.
- [13] Q. Li, D.-M. Chiu, and J. C. Lui, “On the practical and security issues of batch content distribution via network coding,” in *ICNP’06*, 2006.
- [14] F. Zhao, T. Kalkert, M. Medard, and K. J. Han, “Signatures for Content Distribution with Network Coding,” in *ISIT’07*, 2007.
- [15] D. Charles, K. Jain, and K. Lauter, “Signatures for network coding,” in *Info Sciences and Systems*, vol. 1, no. 1, 2006.
- [16] D. Boneh, D. Freeman, J. Katz, and B. Waters, “Signing a Linear Subspace : Signature Schemes for Network Coding,” in *PKC’09*, 2009.
- [17] S. Agrawal and D. Boneh, “Homomorphic MACs : MAC-Based Integrity for Network Coding,” in *ACNS’09*, 2009.
- [18] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, “RIPPLE Authentication for Network Coding,” in *INFOCOM’10*, 2010.
- [19] Y. Jiang, H. Zhu, M. Shi, X. S. Shen, and C. Lin, “An efficient dynamic-identity based signature scheme for secure network coding,” *Computer Networks*, vol. 54, no. 1, pp. 28–40, Jan. 2010.
- [20] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. S. Shen, “Padding for Orthogonality : Efficient Subspace Authentication for Network Coding,” in *INFOCOM’11*.
- [21] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, “On Locating Byzantine Attackers,” in *NetCod’08*, 2008.
- [22] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, “Identifying Malicious Nodes in Network-Coding-Based Peer-to-Peer Streaming Networks,” in *Mini INFOCOM’10*.
- [23] A. Le and A. Markopoulou, “Cooperative Defense Against Pollution Attacks in Network Coding Using SpaceMac,” in *JSAC on Cooperative Networking Challenges and Applications*, 2012.
- [24] —, “TESLA-Based Defense Against Pollution Attacks in P2P Systems with Network Coding,” in *NetCod’11*, 2011.
- [25] —, “Locating Byzantine Attackers in Intra-Session Network Coding using SpaceMac,” in *NetCod’10*, 2010.
- [26] —, “On Detecting Pollution Attacks in Inter-Session Network Coding,” in *Technical Report*. [Online]. Available: <http://arxiv.org/abs/1108.0377>
- [27] S. Agrawal, D. Boneh, X. Boyen, and D. Freeman, “Preventing Pollution Attacks in Multi-Source Network Coding,” in *PKC’10*, 2010.
- [28] W. Yan, M. Yang, L. Li, and H. Fang, “Short Signatures for Multi-source Network Coding,” in *MINES’09*, 2009.
- [29] J. Dong, R. Curtmola, C. Nita-Rotaru, and D. Yau, “Pollution Attacks and Defenses in Wireless Inter-flow Network Coding Systems,” in *WiNC’10*, 2010.
- [30] M. N. Krohn, M. J. Freedman, and D. Mazieres, “On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution,” in *SP’04*, 2004.
- [31] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, “On Private Scalar Product Computation for Privacy-Preserving Data Mining,” in *ICISC’04*, 2004.
- [32] M. Bellare, R. Canetti, H. Krawczyk, “Keying Hash Functions for Message Authentication,” in *CRYPTO’96*, 1996.
- [33] M. Kim, R. Kottter, M. Medard, and J. Barros, “An Algebraic Watchdog for Wireless Network Coding,” in *ISIT09*, 2009.
- [34] M. Kim, M. Medard, and J. Barros, “A multi-hop multi-source algebraic watchdog,” in *ITW’10*, 2010.
- [35] G. Liang, R. Agarwal, and N. Vaidya, “When Watchdog Meets Coding,” in *INFOCOM’10*, 2010.
- [36] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC Press, 2007.
- [37] S. Goldwasser and S. Micali, “Probabilistic Encryption,” *Journal of Computer and System Sciences*, vol. 28, pp. 270–299, 1984.
- [38] P. Paillier, “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,” in *EUROCRYPT’99*, 1999.
- [39] J. Benaloh, “Dense probabilistic encryption,” in *Workshop on Selected Areas of Cryptography*, vol. 28, no. 2, pp. 120–128, Apr. 1994.
- [40] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The TESLA Broadcast Authentication Protocol,” *RSA CryptoBytes*, vol. 5, 2002.
- [41] R. Canetti, J. Garay, G. Itkid, D. Micciancio, M. Naore, and B. Pinkas, “Multicast security: a taxonomy and some efficient constructions,” in *INFOCOM ’99*.
- [42] P. L. Montgomery, “Modular Multiplication Without Trial Division,” *Mathematics of Computation*, vol. 44, no. 170, p. 519, Apr. 1985.