Locating Byzantine Attackers in Intra-Session Network Coding using SpaceMac

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

Abstract—**Intra-session network coding is known to be vulnerable to Byzantine attacks: malicious nodes can inject bogus packets, which get combined with legitimate blocks at downstream nodes, thus preventing decoding of original packets and degrading the overall performance. In this paper, we provide a novel approach that can identify the *precise location* of *all* Byzantine attackers in systems with intra-session network coding. A key ingredient of our approach is a novel homomorphic MAC scheme for expanding subspaces (SpaceMac) that allows to eliminate any uncertainty in identifying attackers via subspace properties. To the best of our knowledge, our scheme is the first that can identify precisely all Byzantine attackers, and at the same time, has both low computation (sub-millisecond) and communication overhead (20 bytes per data block). Simulation results show that even when there are multiple colluding attackers in a network, all of them can be successfully identified in a very short time.**

I. Introduction

In this paper, we are interested in content distribution systems, such as [3] or [15], that employ random network coding[1] to increase throughput and facilitate distributed scheduling. However, network coding is also inherently vulnerable to Byzantine (a.k.a. pollution) attacks: a malicious node can inject corrupted packets to its outgoing links, which then get combined with legitimate packets in downstream nodes, to prevent decoding of the original data and eventually degrade the overall performance. The effect of a single corrupted packet is amplified due to mixing with other packets.

The severity of pollution attacks in systems with network coding has generated a significant amount of work in this area. Proposed defense mechanisms can be classified into three categories: error correction, attack detection, and attacker identification. First, error-correcting mechanisms, such as, [8], [11], deal with the problem end-to-end: they utilize redundancy at the source to correct corrupted packets at the receivers. Second, attack detection mechanisms can be used at intermediate nodes and aim at detecting whether a linear combination is legitimate or corrupted; detected corrupted packets are dropped so they do not further pollute the system. Work in this area includes both information theoretic [5] and cryptographic approaches; the latter leverage null keys [10], homomorphic hashes [4], signatures [1], or message authentication codes (MACs) [2]. Third, there is also work that aims at identifying the nodes where Byzantine attackers are located, either approximately via subspace properties [7] or exactly using a non-repudiation protocol [16].

In this paper, we are interested in the third problem: our goal is to design a system that can accurately identify the location of all Byzantine attackers. This allows to remove the attackers from the system and stop the pollution attack at its root – without wasting network resources on handling corrupted traffic. To the best of our knowledge, none of the existing solutions to this problem possesses all the following desired properties:

(1) Can exactly identify all pollution attackers.
(2) Have low communication and computation overhead.
(3) Can deal with a large number of colluding attackers.

In this paper, we introduce a novel scheme that meets all the above three requirements. Our scheme builds on and extends the work of Jafarisiavoshani *et al.* [7], which exploits subspace properties to approximately identify the location of attackers. To eliminate the uncertainty in identifying pollution attackers via subspace properties, we design a novel homomorphic MAC scheme for expanding subspaces that enforces nodes to report their true incoming subspaces. Our homomorphic MAC scheme is inspired by the scheme by Agrawal and Boneh [1] and has equivalent low computation overhead. However, unlike [1], our scheme allows intermediate nodes to sign subspaces that *expand* over time, as opposed to [1], which allows for signing a fixed space. In addition, to prevent malicious nodes from denying their behaviors and disparaging benign nodes, we incorporate the light-weight non-repudiation transmission protocol by Wang *et al.* [16], which has significantly less communication overhead than most existing schemes. Finally, by leveraging multiple generations, our scheme is the first that can identify all pollution attackers even when there is a large number of colluding attackers. To support the identification process, our scheme requires a reliable low-bandwidth channel between a controller and each node.

We evaluate our scheme via simulation in networks with 52 nodes and up to 18 attackers, some of which are colluding, and demonstrate that we can rapidly identify the location of all attackers. The contributions of this paper lie in the design of (i) the novel homomorphic MAC scheme for expanding subspaces, SpaceMac, in Section IV, and (ii) the overall system described in Section V that combines several components so as to be able to accurately locate all attackers.

II. Related Work

We review here work directly relevant to the problem of attacker location identification we study in this paper. Due to space constraints, we refer the reader to [12] for a detailed review of error-correction and attack detection schemes.

In [7], Jafarisiavoshani *et al.* observe that vectors sent from a node belong to the space spanned by the vectors sent by the source and also the space spanned by the vectors the node receives. Using this observation, in a general network topology

[1]Our scheme actually works for both random and deterministic network coding. We choose to describe the scheme in the randomized setting, which is the case in practical content distribution systems.

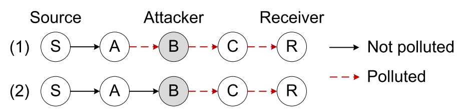


Fig. 1. An example of inferring an attacker's location using information about polluted edges from [7]. The attacker is at node B. Scenarios (1) and (2) correspond to the sets of polluted edges when the attacker lies and is honest about its incoming subspace, respectively. The controller can narrow down the attacker to two nodes: A and B, as they initiate polluted edges.

having a single attacker, the attacker can be located with an uncertainty of at most two nodes. When there are multiple attackers, the uncertainty is within a set of nodes including the attackers and their parents and children. Our scheme builds on and extends this work: we make it possible to pinpoint the exact location of the attackers, even in the case when there are multiple colluding attackers.

Wang *et al.* [16] introduce a light-weight non-repudiation protocol to identify malicious nodes. Their protocol ensures that a malicious node that injected a polluted vector cannot deny its behavior and cannot disparage any innocent node. However, using the approach in [16], the identification requires the distribution of multiple checksums of all source vectors to all the nodes for every generation that experiences pollution attack. This incurs a significant communication overhead. Moreover, this approach is unable to identify all attackers. In contrast, our scheme can identify all attackers rapidly without the need of checksums.

A key ingredient of our solution is the design of a novel homomorphic MAC scheme (SpaceMac) that allows intermediate nodes to sign their own –*expanding* over time– subspaces, and thus become accountable for the packets they send, which is necessary to identify malicious nodes. This is different from the homomorphic MAC scheme by Agrawal and Boneh [1], which allows for signing only fixed subspaces. A detailed comparison is provided in Section IV-B.

Independently and at the same time as our paper, a similar homomorphic MAC construction has been proposed for a different problem [13]. In particular, the *inner-product* MAC scheme was introduced in [13] to facilitate nested authentications, prevent tag pollution attacks, and support Byzantine attack detection. We propose SpaceMac for authenticating expanding subspaces and identifying the location of attackers.

III. Problem Formulation

A. Network Operation

Consider a directed-acyclic-graph (DAG) network that employs linear network coding. When sending a message, a source node S first breaks the message into m vectors, $\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_m$, in an n-dimensional linear space $\mathbb{F}_q^n$, where m, n and q are fixed ahead of time and $q \gg 1$. The source then augments every vector $\hat{\mathbf{v}}_i$ with m additional components, which form the coefficient vector of $\hat{\mathbf{v}}_i$. The resulting vectors $\mathbf{v}_i$'s, called *augmented vectors*, have the following form:

$$\mathbf{v}_i = (—\hat{\mathbf{v}}_i—, \overbrace{\underbrace{0, \cdots, 0, 1}_{i}, 0, \cdots, 0}^{m}) \in \mathbb{F}_q^{n+m} .$$

Let $\Pi_S \subseteq \mathbb{F}_q^{n+m}$ be the subspace spanned by vectors $\mathbf{v}_i$'s. Each intermediate node i in the network receives from its parents some vectors, which are linear combinations of the $\mathbf{v}_i$'s. It then creates linear combinations of the received vectors and sends them to its adjacent downstream nodes. We use $\Pi_i^{(j)}(t) \subseteq \mathbb{F}_q^{n+m}$ to denote the space spanned by the vectors received by node i from node j up to time t. When there is no ambiguity, we omit the time index t.

When all the intermediate nodes are benign, the spaces $\Pi_i^{(j)}$'s are subspaces of Π_S. Assume that there is a pollution attacker in the network. The attacker combines a subspace Π_E with its incoming space and sends the resulting vectors to its children; thus, these children' incoming spaces are not subspaces of Π_S. Formally, for every node i, its incoming space $\Pi_i^{(j)}$ from its parent j can be decomposed as follows:

$$\Pi_i^{(j)} = \Pi_{S_i}^{(j)} \oplus \hat{\Pi}_i^{(j)} ,$$

where $\oplus$ denotes the direct sum of spaces, $\Pi_{S_i}^{(j)} \overset{\text{def}}{=} \Pi_S \cap \Pi_i^{(j)}$, and $\hat{\Pi}_i^{(j)}$ is the rest, which are vectors not belonging to Π_S.

We define a *polluted directed edge* as follows:

Definition 1. *A directed edge is polluted if it transmits any vector which is not a linear combination of* $\mathbf{v}_i$*'s.*

The below lemma is a direct consequence of the definition:

Lemma 1. *The following statements are equivalent:*
(1) A directed edge $e(j, i)$ *is polluted*
(2) $\Pi_i^{(j)} \not\subseteq \Pi_S$
(3) $\hat{\Pi}_i^{(j)} \neq \emptyset$

B. Key Observation

Jafarisiavoshani *et al.* [7] have shown that when there is a single pollution attacker, its location can be narrowed down to a set of at most two nodes. This is by analyzing the polluted edges identified based on the incoming subspaces reported by all the nodes to a central controller. An example is shown in Fig. 1. However, when there are multiple attackers in a general network topology, the number of suspected nodes increases.

We observe that the uncertainty about the location of the attackers originates from the fact that the attackers can lie about their incoming spaces. Consequently, by ensuring that all nodes in the network cannot lie about their received spaces, we can exactly locate the attackers. For example, if the attacker cannot lie in the example given in Fig. 1, then the only possible scenario is scenario (2). Thus, one can determine that the attacker is at node B.

IV. Homomorphic MAC for Expanding Subspaces

In order to enforce nodes to report their true incoming spaces, we require that when a node reports a space of one of its parents, it has to report a MAC tag (or simply *tag*) of the space as well. The tag is generated based on the secret key shared by the parent and the controller. Since an attacker does not know its parent's secret key, it cannot generate a valid tag for a space not from the parent. We note that from the perspective of an intermediate node i, the subspace $\Pi_i^{(j)}$ it receives from a parent node j potentially *expands* when it receives more vectors from node j. As a result, the tag of the space $\Pi_i^{(j)}$ dynamically *changes over time*.

A. Subspace Properties

The following lemma shows for each subspace $\Pi_i^{(j)}$, node i may report a randomly chosen vector, $\mathbf{y}_r$, of the space instead of the space itself; and by checking if $\mathbf{y}_r \in \Pi_S$, the controller can determine if $\Pi_i^{(j)} \subseteq \Pi_S$ to identify the polluted edges.

Lemma 2 (Jafarisiavoshani *et al.* [6], [7]). *Let Π_1 and Π_2 be two subspaces of $\mathbb{F}_q^{n+m}$ and assume that $\mathbf{y}_r$ is a randomly selected vector from Π_1. Then, for $q \gg 1$, $\mathbf{y}_r \in \Pi_2$ if and only if $\Pi_1 \subseteq \Pi_2$.*

*B. Homomorphic MAC Scheme (*SpaceMac*)*

To allow a child node to generate a valid tag for a randomly chosen vector, $\mathbf{y}_r$, of the parent's space, for every vector, $\mathbf{y}$, that the parent sends to the child, the parent also sends a tag of $\mathbf{y}$. The tag is computed based on a homomorphic MAC scheme so that the child can combine its received tags to calculate a valid tag for $\mathbf{y}_r$.

Recently, Agrawal and Boneh [1] introduced an elegant homomorphic MAC scheme for network coding. However, this scheme requires the sending node to know in advance all the basis vectors of the space it is sending so that it can sign them (*i.e.*, generating tags for the basis vectors.) This is required because the tag for a random vector in the space is generated based on the tags of the basis vectors. In other words, the scheme in [1] requires that the sending space is *fixed* in advance; as such, it is not applicable here because the space spanned by vectors $\mathbf{y}$'s is not fixed but rather continuously expands. We now describe the construction of our homomorphic MAC scheme, called SpaceMac, which is customized to support expanding spaces.

1) Constructions: Let $\mathcal{K}$ and $\mathcal{I}$ denote the domains of the keys and the id's of the spaces sent by the source, respectively. Let $[n]$ denote $\{1, \cdots, n\}$. Our scheme uses a pseudorandom function (PRF) $F : \mathcal{K} \times \mathcal{I} \times [n+m] \to \mathbb{F}_q$.

- $\mathsf{Mac}(k, \mathsf{id}, \mathbf{y})$:
  - Input: a secret key k, the identifier id of the vector space Π_S sent by the source, and a vector $\mathbf{y} \in \mathbb{F}_q^{n+m}$.
  - Output: tag t for $\mathbf{y}$, where t is computed as follows:
    - ◦ $\mathbf{r} \leftarrow (F(k, \mathsf{id}, 1), \cdots, F(k, \mathsf{id}, n+m))$
    - ◦ $t \leftarrow \mathbf{y} \cdot \mathbf{r} \in \mathbb{F}_q$
- $\mathsf{Combine}((\mathbf{y}_1, t_1, \alpha_1), \cdots, (\mathbf{y}_p, t_p, \alpha_p))$:
  - Input: p vectors $\mathbf{y}_1, \cdots, \mathbf{y}_p$, their tags $t_1, \cdots, t_p$ under key k, and their coefficients $\alpha_1, \cdots, \alpha_p \in \mathbb{F}_q$.
  - Output: tag t for vector $\mathbf{z} \overset{\text{def}}{=} \sum_{i=1}^{p} \alpha_i \mathbf{y}_i$:
    - ◦ $t \leftarrow \sum_{i=1}^{p} \alpha_i t_i \in \mathbb{F}_q$
- $\mathsf{Verify}(k, \mathsf{id}, \mathbf{z}, t)$:
  - Input: a key k, the identifier id of Π_S, a vector $\mathbf{z} \in \mathbb{F}_q^{n+m}$, and its tag t
  - Output: 0 (reject) or 1 (accept) as follows:
    - ◦ $\mathbf{r} \leftarrow (F(k, \mathsf{id}, 1), \cdots, F(k, \mathsf{id}, n+m))$
    - ◦ $t' \leftarrow \mathbf{z} \cdot \mathbf{r}$
    - ◦ If $t' = t$, output 1; otherwise, output 0

Assume $\mathbf{z} = \sum_{i=1}^{p} \alpha_i \mathbf{y}_i$, the correctness is given by

$$t' = \mathbf{z} \cdot \mathbf{r} = (\sum_{i=1}^{p} \alpha_i \mathbf{y}_i) \cdot \mathbf{r} = \sum_{i=1}^{p} \alpha_i (\mathbf{y}_i \cdot \mathbf{r}) = \sum_{i=1}^{p} \alpha_i t_i \, .$$

Compared to the homomorphic MAC scheme in [1], our scheme replaces the Sign algorithm, which signs a fixed vector space, *i.e.*, generates tags for all bases of the fixed space, with the Mac algorithm, which generates tags for any vectors in $\mathbb{F}_q^{n+m}$. The Combine and Verify algorithms show that tags generated by our Mac algorithm can be combined naturally to produce a valid tag for an arbitrary linear combination.

2) Attack Game 1: We consider the following attack game for a homomorphic MAC $\mathcal{T} = (\mathsf{Mac}, \mathsf{Combine}, \mathsf{Verify})$, a challenger $\mathcal{C}$, and an adversary $\mathcal{A}$:

- *Setup.* $\mathcal{C}$ generates a random key $k \overset{\text{R}}{\leftarrow} \mathcal{K}$
- *Queries.* $\mathcal{A}$ adaptively queries $\mathcal{C}$, where each query is of the form $(\mathsf{id}, \mathbf{y})$. For each query, $\mathcal{C}$ replies to $\mathcal{A}$ with the corresponding tag $t \leftarrow \mathsf{Mac}(k, \mathsf{id}, \mathbf{y})$.
- *Output.* $\mathcal{A}$ eventually outputs a tuple $(\mathsf{id}^*, \mathbf{y}^*, t^*)$.

Up to the time $\mathcal{A}$ outputs, it has queried $\mathcal{C}$ multiple times. Let l denote the number of times $\mathcal{A}$ queried $\mathcal{C}$ using id* and get tags for l vectors, $\mathbf{y}_1^*, \cdots, \mathbf{y}_l^*$, of these queries. We consider that the adversary wins the security game if and only if

(i) $\mathbf{y}^* \neq \mathbf{0}$ (trivial forge otherwise),
(ii) $\mathsf{Verify}(k, \mathsf{id}^*, \mathbf{y}^*, t^*) = 1$, and
(iii) $\mathbf{y}^* \notin \mathsf{span}(\mathbf{y}_1^*, \cdots, \mathbf{y}_l^*)$.

Let Adv$[\mathcal{A}, T]$ denote the probability that $\mathcal{A}$ wins this attack game 1. We define a secure homomorphic MAC as follows:

Definition 2. *A (q, n, m) homomorphic MAC scheme $\mathcal{T}$ is secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$,* Adv$[\mathcal{A}, T]$ *is negligible.*

3) Security: Let $\mathcal{B}$ denote a PRF adversary, and PRF-Adv$[\mathcal{B}, F]$ denote $\mathcal{B}$'s advantage in winning the PRF security game with respect to F. (The definition of PRF security game can be found in [9].)

Theorem 1. *For any fixed q, n, m,* SpaceMac *is a secure (q, n, m) homomorphic MAC assuming F is a secure PRF. In particular, for every homomorphic MAC adversary $\mathcal{A}$, there is a PRF adversary $\mathcal{B}$ with similar running time to $\mathcal{A}$ such that*

$$\mathsf{Adv}[\mathcal{A}, \mathsf{SpaceMac}] \leq \mathsf{PRF\text{-}Adv}[\mathcal{B}, F] + \frac{1}{q} \, .$$

Proof: See the Appendix.

Theorem 1 implies that an adversary $\mathcal{A}$ can break the scheme with probability $\frac{1}{q}$. To improve the security, one could either increase the field size or use multiple tags. The security of our scheme using l tags is $(\frac{1}{q})^l$.

C. Non-repudiation Transmission Protocol

SpaceMac forces the attacker to report only true spaces received from its parents since it is computationally difficult (to forge a valid tag) otherwise; however, it does not prevent a malicious node from sending invalid tags to its children to prevent the children from reporting the polluted spaces.

For example, an attacker j can send a polluted vector $\mathbf{y}_e \notin \Pi_S$ and a bogus tag $t_e \not\leftarrow \mathsf{Mac}(k_j, \mathsf{id}, \mathbf{y}_e)$, to its child i. When i reports the space $\Pi_i^{(j)}$, if the randomly chosen vector $\mathbf{y}_r$ was formed by a linear combination involving $\mathbf{y}_e$, then the aggregated tag t_r of $\mathbf{y}_r$ that i generates using the algorithm Combine will be very likely invalid because of the bogus tag t_e. As a result, the controller will reject $\mathbf{y}_r$. Consequently, the attacker j successfully prevents its benign child i from reporting the polluted space $\Pi_i^{(j)}$.

To address this, we utilize an efficient non-repudiation transmission protocol proposed by Wang *et al.* [16]: For a parent j and a child i, the controller generates a pool of secret keys $\mathcal{S}$ based on the private key k_j of the parent and the ID i of the child. After that, the controller randomly selects a set of keys $\mathcal{P}$ from $\mathcal{S}$ based on the private k_i of the child and the

ID j of the parent; then, it sends $\mathcal{P}$ to the child. We denote $\mathcal{S} \setminus \mathcal{P}$ as $\overline{\mathcal{P}}$; also, let $\lambda \overset{\text{def}}{=} |\mathcal{S}|$ and $\delta \overset{\text{def}}{=} |\mathcal{P}|$.

When sending a vector, j is required to generates λ tags instead of one, using the algorithm Mac and all keys in $\mathcal{S}$. When receiving a vector, i uses its set of keys $\mathcal{P}$ and the algorithm Verify to verify δ out of λ tags. Finally, when receiving a randomly chosen vector $\mathbf{y}_r$ representing $\Pi_i^{(j)}$ and its λ tags from the child, the controller uses all keys in $\overline{\mathcal{P}}$ and the algorithm Verify to verify all $\lambda - \delta$ tags. The controller, in this case, keeps track of a counter θ, $\theta \leq \lambda - \delta$. If at least θ tags pass the verification then the controller accepts the report.

We describe below the two theorems (based on theorems in [16]) that provide the security of the non-repudiation transmission protocol when applying to our context.

Theorem 2 (Non-repudiation of the receiver–Wang *et al.* [16])**.** *The probability that a malicious child node can successfully report to the controller that its parent sends it a vector* $\mathbf{y}$*, which is never sent by the parent, is at most*

$$\sum_{i=\theta}^{\lambda-\delta} \binom{\lambda-\delta}{i} \frac{1}{q^i} \left(1 - \frac{1}{q}\right)^{\lambda-\delta-i} .$$

Theorem 3 (Non-repudiation of the sender–Wang *et al.* [16])**.** *The probability that a malicious parent can make the controller reject the parent's space reported by one of its children by sending the child some vectors with invalid tags is at most*

$$\max_{0 \leq x \leq \delta+\theta-1} p(x), \text{ where } p(x) \leq \sum_{i=max(x-\theta+1,0)}^{min(\delta,x)} \frac{\binom{\delta}{i}\binom{\lambda-\delta}{x-i}}{\binom{\lambda}{x} q^{\delta-i}} .$$

Because of the space constraint, we refer the readers to our technical report [12] for full proofs of both theorems. Finally, we remark that both probabilities above can be made very small by choosing the appropriate values for q, λ, δ, and θ.

V. Full Description of Our Proposed Scheme

We assume that there is a controller who knows the complete topology and the source space Π_s; each node knows the identifiers of its adjacent nodes; both the source and the receivers are trusted; and there is a reliable low-bandwidth end-to-end communication path between the controller and each node. Let (pk, sk) and (pk_j, sk_j) be the public and secret key pairs of the controller and node j, respectively. We assume that each node has an authentic copy of pk, and the controller has authentic copies of pk_j's. Let Enc_k denote symmetric-key encryption using key k.

1. Initialization. Every node j sends k_j to the controller using a key transport protocol which provides strong authentication, such as, the X.509 protocol [14]. Let $\mathcal{I}_j$ be the set of IDs of adjacent downstream nodes of j. For $i \in \mathcal{I}_j$, the controller generates a set $\mathcal{S}_{ji}$ of λ keys using a PRF F_1: $\mathcal{K} \times \mathcal{I} \times [\lambda] \rightarrow \mathcal{K}$, where $\mathcal{K}$ is the domain of key k_j, and $\mathcal{I}$ is the domain of the identifiers of the nodes: $\mathcal{S}_{ji} \leftarrow \{F_1(k_j, i, l), \text{ for } l = 1, \cdots, \lambda\}$.

For $i \in \mathcal{I}_j$, consider an array L whose elements are distinct subsets of size δ of $\mathcal{S}_{ji}$. Note that L has length $\binom{\lambda}{\delta}$. The controller uses another PRF F_2: $\mathcal{K} \times \mathcal{I} \rightarrow [\binom{\lambda}{\delta}]$ to select from L a subset of size δ: $\mathcal{P}_{ji} = L[t]$, where $t \leftarrow F_2(k_i, j)$. The controller sends $\mathsf{Enc}_{k_i}(\mathcal{P}_{ji})$ to node i. Denote $\mathcal{S}_{ji} \setminus \mathcal{P}_{ji}$ as $\overline{\mathcal{P}_{ji}}$.

2. Sending and Receiving. Let id be the identifier of the current source space Π_s. When a node j sends a vector $\mathbf{y}$ to its downstream node i, beside the id, it has to send along λ tags, which are computed using the Mac algorithm and keys in $\mathcal{S}_{ji}$. Let $\mathcal{T}_{ji}(\mathbf{y})$ denote this set of tags. Node j sends $(\text{id}, \mathbf{y}, \mathcal{T}_{ji}(\mathbf{y}))$. When node i receives $(\text{id}, \mathbf{y}, \mathcal{T}_{ji}(\mathbf{y}))$ from node j, it uses $\mathcal{P}_{ji}$ and the Verify algorithm to check the validity of δ out of λ tags of $\mathcal{T}_{ji}$. It drops $\mathbf{y}$ as long as there is an invalid tag. Otherwise, it stores the received tuple in its buffer.

3. Pollution Detection and Alert. One way to detect pollution at the receivers is to implement the homomorphic MAC scheme proposed in [1], where, in this case, only the source and the receivers share the same MAC key. A receiver, upon detecting a pollution, sends an alert (id, r, t) to the controller, where id is the identifier of the source space; r is the receiver ID; and t is a tag computed for (id, r) using a traditional MAC, *e.g.*, HMAC, with key k_r. When the controller receives an alert, it first verifies the tag t and drops the alert if t is invalid. Then, it determines if id is reported before, if so, it ignores the alert. Otherwise, it sends a request (id, t_j) to each node j, which demands each node to report its incoming subspace, where id is the identifier of the source space, and t_j is the tag computed for id using a HMAC with key k_j.

4. Reporting Subspaces. Upon receiving the request (id, t_i) from the controller, each node i verifies t_i using k_i and drops the request if t_i is invalid. Otherwise, let $(\mathbf{y}_1, t_{1,1}, \cdots, t_{1,\lambda}), \cdots, (\mathbf{y}_l, t_{l,1}, \cdots, t_{l,\lambda})$ be vectors of source space id and their tags that node i received from node j. For each parent node j of i, node i sends $(i, j, \mathbf{y}_r, t_1, \cdots, t_\lambda, t)$ to the controller, where $\alpha_i \overset{R}{\leftarrow} \mathbb{F}_q$ $(i \in [l])$; $\mathbf{y}_r = \sum_{i=1}^{l} \alpha_i \mathbf{y}_i$; $t_i = \sum_{j=1}^{l} \alpha_j t_{j,i}$ $(i \in [\lambda])$; and t is a tag of $(i, j, \mathbf{y}_r, t_1, \cdots, t_\lambda)$ computed using a HMAC with key k_i.

5. Identifying the Attackers. After sending out the requests, the controller waits for the reports. After Δt seconds, it starts identifying the pollution attackers. First, it drops reports that have invalid HMAC tags. Then, it classifies any node that does not report all of its incoming spaces as a malicious node. Subsequently, it accepts reports with at least θ valid SpaceMac tags, where the validation uses keys in $\overline{\mathcal{P}_{ji}}$'s. Then, it identifies the polluted edges in the network based on the reported spaces and the source space. We note that checking if a reported space is polluted can be done quickly and efficiently in $O(n)$ in terms of multiplication operations by leveraging the global coding coefficients of the reported vector and the basis vectors of the source space. Finally, any node that does not have a polluted incoming edge but has a polluted outgoing edge is classified as malicious.

6. Releasing the Result. After identifying the set of attackers A, the controller sends (A, t_i) to each benign node i, where t_i is a tag computed for A using HMAC with key k_i. Upon receiving the tuple (A, t_i), each node i in the network verifies t_i using k_i, then if valid, it adds nodes in A into its blacklist. Every node in the network will neither send nor receive traffic from nodes in its blacklist in subsequent communication.

VI. Security Analysis

1. Single Adversary. We consider two different cases, where an attacker injects erroneous vectors into one or more than one of its downstream links. Jafarisiavoshani *et al.* [7] have shown that in a general network, the location of the adversary can be narrowed down up to a set of at most two nodes in both cases by partitioning the edges into two set: the set of polluted edges, E_p, and non-polluted edges, E_s, then analyzing the nodes with respect to the identified E_p and E_s. They also note that the partitioning of E_p and E_s is not unique since the adversary might lie. Using our scheme, the probability that the attacker lies about its incoming spaces is very small (theorem 2.) Furthermore, the probability that the attacker can prevent its children from reporting the subspaces polluted by itself is very small, too (theorem 3.) As a result, with high probability (depending on q, λ, δ, and θ), our scheme can produce an unambiguous partitioning of E_p and E_s, which helps to uniquely identify the attacker.

2. Multiple Adversaries. In the presence of multiple adversaries, an attacker might be "in the shadow" of some other attackers, which means that it might pollute only already polluted data and thus does not produce any detectable effect.

Definition 3. *An attacker is* shadowed *if it has at least one polluted incoming edge and is* exposed *otherwise.*

a) Independent Adversaries. In this case, we note that with high probability, our approach is already able to identify all exposed attackers. We utilize the following observation to identify all shadowed and exposed attackers.

Lemma 3. *For any directed acyclic graph with pollution attack in presence, there is at least one exposed attacker.*

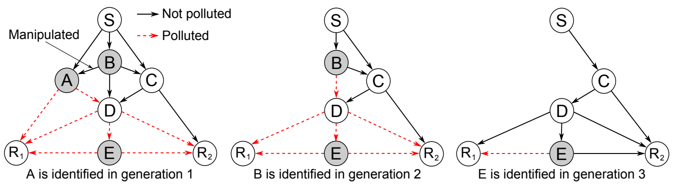
Proof: Consider a topological ordering of the graph, the first malicious node in the ordering is an exposed attacker. ■

Exploiting this, we can use multiple generations, *i.e.*, transmissions of (different) source spaces, to identify all attackers.

Lemma 4. *In a network with M independent attackers. With high probability (depending on q, λ, δ, and θ), all attackers can be identified after N generations which experience pollution attack, where $N \leq M$.*

b) Colluding Adversaries. We note that each pair of parent j and child i uses distinct key sets $\mathcal{S}_{ji}$ and $\mathcal{P}_{ji}$; thus, the collusion of malicious nodes does not provide knowledge about the key sets of benign nodes. However, when the distance between any two attackers equals to one, where distance refers to the length of the shortest path connecting two nodes, these attackers can collude to report a false space.

Assume that in a network, there are colluding attackers j and i connected by a directed edge $e(j,i)$ and there is no other pair of attackers in the network having distance one. We ask the question: "What can j and i achieve by manipulating edge $e(j,i)$?" Consider a topological ordering $\mathcal{O}$ of the nodes. If i makes $e(j,i) \in E_p$ then j is exposed and identified after all malicious nodes that come before j in $\mathcal{O}$ are identified. After j is identified, i and the rest of the attackers will be eventually identified. Otherwise, if i makes $e(j,i) \in E_s$ then i is exposed and identified after all malicious nodes that come before i in $\mathcal{O}$ are identified. Analogous to the other case, after i is identified, j (if not already identified) and the rest of the attackers will be eventually identified. Consequently, by manipulating the status of edge $e(j,i)$, the attackers can, at best, change the order in which j and i are identified. The above analysis can be extended to the general case where there are multiple pairs having distances one by considering the pair (j,i), where i has a polluted outgoing edge, that appears first in $\mathcal{O}$ first. Fig. 2 shows an example. As a result, we can generalize lemma 4:



Not polluted
Polluted
Manipulated
S
A
B
C
D
E
R_1
R_2
A is identified in generation 1
B is identified in generation 2
E is identified in generation 3

Fig. 2. An example where there are three attackers B, A, and E. Attackers A and B collude to make BA, which is polluted, non-polluted. Nevertheless, all are identified after 3 generations.

Number of attackers	4	8	12	16	20
Average number of generations	1.89	2.91	3.85	4.69	4.89
Average delay (ms)	219	334	439	532	549

TABLE I
The average number of generations and delay required to detect all attackers.

Lemma 5. *In a network with M attackers. With high probability (depending on q, λ, δ, and θ), all attackers can be identified after N generations which experience pollution attack, where $N \leq M$.*

VII. Evaluation

1. Colluding Adversaries. We generate between a pair of source and receiver nodes a random network of 50 nodes. The ratio of edges to nodes is a random number in $[1, 5]$. All edges have a random propagation delay between 10 and 100 ms. In a single generation, 5 augmented vectors in $\mathbb{F}_{2^8}^{1024}$ are generated and sent by the source node. The attackers in the network are chosen randomly from the population of the 50 network nodes in a way that the removal of them does not disconnect the receiver from the sender, and each attacker can still pollute when the rest attackers are removed. The attackers pollute all of their outgoing edges. When asked by the controller, most of them honestly report their incoming subspaces; however, some of them, whose have malicious parents, lie about their received subspaces from those parents. We evaluate the average number of generations to identify all M attackers in the network, where M varies from 4 to 20. For each M, we perform the simulation for 100 rounds to get the average value. We also evaluate the average delay it takes to identify all attackers, where the delay refers to the time between when the source starts sending and when all the attackers are identified. The results shown in table I indicate that we succeed in identifying all attackers after much smaller than M generations.

2. Computation Overhead. The main computation overhead is attributed to the operations Mac, Combine, and Verify performed at each intermediate node. As discussed in [1], one can achieve low computation overhead by implementing multiplication using offline table look-up, addition using simple XOR, and PRFs using AES from OpenSSL. Table II shows the estimated computational delays based on results from [1]. We note that for each new generation, the first Mac and Verify operations introduce larger delays (as shown in table II) because they have to use PRFs; however, results of the PRF calls can be precomputed (if id's are known) as well as cached to use for the whole generation, which makes their subsequent delays as small as Combine's.

3. Communication Overhead. The major communication overhead of our scheme comes from the tags accompanying each data block. Table III shows that with an overhead of about 20 bytes per data block, the probability that a malicious parent succeeds in preventing its child to report, Pr[j], and the probability that a malicious child succeeds in disparaging its

Parameters			Computation Time per Tag (μs)		
q	m	$n+m$	Mac	Verify	Combine
2^8	5	1024	< 1000	< 1000	< 1

TABLE II
Estimated computational delays based on [1]

q	λ	δ	θ	Pr$[j]$	Pr$[i]$	Space Overhead
2^8	19	9	3	2^{-10}	2^{-17}	20 bytes
2^8	24	12	3	2^{-14}	2^{-16}	25 bytes
2^8	29	14	4	2^{-16}	2^{-21}	30 bytes

TABLE III
The probability a malicious parent succeeds in preventing its child to report, Pr$[j]$, the probability a malicious child succeeds in disparaging its parent, Pr$[i]$, and the space overhead correspond to different parameter sets.

parent, Pr$[i]$, are both very small.

VIII. Conclusion

In this study, we proposed a defense scheme to combat pollution attacks for intra-session network coding. Our scheme can identify and eliminate all attackers. The key element of our approach is the novel homomorphic MAC scheme for expanding subspaces (SpaceMac) that we utilized together with subspace properties to allow the exact identification of all attackers, even when a large number of them collude. Moreover, our scheme has both low computation and communication overhead. The simulation results show that when there are multiple colluding attackers, all of them can be successfully identified in a very short time.

IX. Acknowledgements

We would like to thank Stanislaw Jarecki at UC Irvine for insightful comments on the SpaceMac scheme. This work was partially supported by NSF CAREER award 0747110.

References

[1] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-Based Integrity for Network Coding," in *Proc. ACNS*, 2009.
[2] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a Linear Subspace: Signature Schemes for Network Coding," in *Proc. PKC*, 2009.
[3] C. Gkantsidis and P. R. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proc. IEEE INFOCOM*, 2005.
[4] ——, "Cooperative Security for Network Coding File Distribution," in *Proc. IEEE INFOCOM*, 2007.
[5] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. R. Karger, "Byzantine modification detection in multicast networks with randomized network coding," *IEEE Transactions on Information Theory, Special Issue on Information Theoretic Security*, vol. 54, pp. 2798–2803, June 2008.
[6] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "Subspace Properties of Randomized Network Coding," in *Proc. Info. Theory Workshop*, 2007.
[7] ——, "On Locating Byzantine Attackers," in *Proc. Network Coding Workshop: Theory and Applications*, 2008.
[8] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient Network Coding in the Presence of Byzantine Adversaries," in *Proc. IEEE INFOCOM*, 2007.
[9] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC Press, 2007.
[10] E. Kehdi and B. Li, "Null Keys: Limiting Malicious Attacks Via Null Space Properties of Network Coding," in *Proc. IEEE INFOCOM*, 2009.
[11] R. Koetter and F. R. Kschischang, "Coding for Errors and Erasures in Random Network Coding," in *Proc. IEEE SIT*, 2007.
[12] A. Le, "Technical report: On identifying pollution attackers for randomized network coding," http://www.ics.uci.edu/ anhml/publications.html.
[13] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE Authentication for Network Coding," in *Proc. IEEE INFOCOM*, 2010.
[14] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.
[15] M. Wang and B. Li, "R2: Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 1655–1666, 2007.
[16] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, "Identifying Malicious Nodes in Network-Coding-Based Peer-to-Peer Streaming Networks," in *Proc. IEEE INFOCOM*, 2010.

Appendix

The proof of theorem 1 is by using a sequence of games denoted Game 0 and Game 1. Let W_0 and W_1 denote the events that $\mathcal{A}$ wins the homomorphic MAC security in Game 0 and Game 1, respectively. Game 0 is identical to Attack Game 1 applied to the scheme SpaceMac. Hence,

$$\Pr[W_0] = \text{Adv}[\mathcal{A}, \mathsf{SpaceMac}] \tag{1}$$

Game 1 is identical to Game 0 except that the challenger $\mathcal{C}$ computes $\mathbf{r} \leftarrow (r_1, \cdots, r_{n+m})$, where $r_i \overset{R}{\leftarrow} \mathbb{F}_q$ instead of $r_i \leftarrow F(k, \text{id}, i)$, and everything else remains the same. Then, there exists a PRF adversary $\mathcal{B}$ such that

$$|\Pr[W_0] - \Pr[W_1]| = \text{PRF-Adv}[\mathcal{B}, F] \tag{2}$$

The complete challenger in Game 1 works as follows:

Queries. $\mathcal{A}$ adaptively queries $\mathcal{C}$, where each query is of the form $(\text{id}, \mathbf{y})$. $\mathcal{C}$ replies to query i of $\mathcal{A}$ as follows:

```
if id is never used in any of the previous queries:
    $\mathbf{r}_i := (r^i_1, \cdots, r^i_{n+m})$, where $r^i_j \overset{R}{\leftarrow} \mathbb{F}_q, j \in [n+m]$
else:
    $\mathbf{r}_i$ := the one used in the previous response
send $t := \mathbf{y}_i \cdot \mathbf{r}_i$ to $\mathcal{A}$
```

Output. $\mathcal{A}$ eventually outputs a tuple $(\text{id}^*, \mathbf{y}^*, t^*)$. When $\mathbf{y}^*$ does not equal $\mathbf{0}$, to determine if $\mathcal{A}$ wins the game, we compute

```
if $\text{id}^* = \text{id}_i$ (for some $i$) then        // case (i)
    set $\mathbf{r}^* := \mathbf{r}_i$
else                                         // case (ii)
    set $\mathbf{r}^* := (r^*_1, \cdots, r^*_{n+m})$, where $r^*_i \overset{R}{\leftarrow} \mathbb{F}_q, i \in [n+m]$
```

Let l denote the number of times $\mathcal{A}$ queried $\mathcal{C}$ using id* and get tags for l vectors, $\mathbf{y}^*_1, \cdots, \mathbf{y}^*_l$, of these queries. The adversary wins the game, *i.e.*, event W_1 happens, if and only if

$$t^* = \mathbf{y}^* \cdot \mathbf{r}^* \text{ , and} \tag{3}$$

$$\mathbf{y}^* \notin \mathsf{span}(\mathbf{y}^*_1, \cdots, \mathbf{y}^*_l) \,. \tag{4}$$

We will show that $\Pr[W_1] = \frac{1}{q}$. Let T be the event that $\mathcal{A}$ outputs a tuple with a completely new id*, *i.e.*, $\mathcal{A}$ never made queries using id* before.

• When T happens, *i.e.*, in case (ii), since r^*_i 's are indistinguishable from random values and $\mathbf{y}^* \neq \mathbf{0}$, the right hand side of equation (3) is a completely random value in $\mathbb{F}_q$. Thus,

$$\Pr[W_1 \wedge T] = \frac{1}{q} \Pr[T] \,. \tag{5}$$

• When T does not happen, *i.e.*, in case (i): $\mathbf{r}^*$ of equation (3) equals $\mathbf{r}_i$ for some i, and $\mathbf{r}^*$ has been used to generate tags for vectors $\mathbf{y}^*_1, \cdots, \mathbf{y}^*_l$. In this case, we proceed by showing that for a fixed $\mathbf{y}^*$, t^* looks indistinguishable from a random value in $\mathbb{F}_q$. To this end, let $\Pi = \mathsf{span}(\mathbf{y}^*_1, \cdots, \mathbf{y}^*_l)$ and d be the dimension of Π. $d < m+n$ because otherwise $\Pi = \mathbb{F}^{n+m}_q$, which implies $\mathbf{y}^* \in \Pi$. Let $\{\mathbf{b}_1, \cdots, \mathbf{b}_d\}$ be a basis of Π. Let $r^*_1, \cdots, r^*_{n+m}$ be the unknowns. Remark that $\mathbf{r}^* = (r^*_1, \cdots, r^*_{n+m})$. The queries and the output form the following system of linear equations:

$$\begin{aligned} \mathbf{y}^*_1 \cdot \mathbf{r}^* &= t_{\mathbf{y}^*_1} \\ &\cdots \\ \mathbf{y}^*_l \cdot \mathbf{r}^* &= t_{\mathbf{y}^*_l} \\ \mathbf{y}^* \cdot \mathbf{r}^* &= t^* \end{aligned}$$

This system is equivalent to the below system:

$$\begin{aligned} \mathbf{b}^*_1 \cdot \mathbf{r}^* &= t_{\mathbf{b}^*_1} \\ &\cdots \\ \mathbf{b}^*_d \cdot \mathbf{r}^* &= t_{\mathbf{b}^*_d} \\ \mathbf{y}^* \cdot \mathbf{r}^* &= t^* \end{aligned}$$

Note that $t_{\mathbf{b}^*_i}$ is a linear combination of some $t_{\mathbf{y}^*_i}$'s, and that $\mathbf{y}^*$ is linearly independent of $\mathbf{b}^*_i$'s . Now, the above system of equations is consistent regardless of the value of t^* because the coefficient matrix has rank $d+1$, which equals the number of equations. Furthermore, for any value t^*, the solution space always has the same size $q^{m+n-d-1}$. Thus, for a fixed $\mathbf{y}^*$, its valid tag t^* could be any value in $\mathbb{F}_q$ equally likely, given that r^*_i's are chosen uniformly at random from $\mathbb{F}_q$. As a result, the probability that the adversary chooses a correct t^* is $1/q$. Thus,

$$\Pr[W_1 \wedge \neg T] = \frac{1}{q} \Pr[\neg T] \,. \tag{6}$$

• From equations (5) and (6), we have

$$\Pr[W_1] = \Pr[W_1 \wedge T] + \Pr[W_1 \wedge \neg T] = \frac{1}{q} \,. \tag{7}$$

Equations (1), (2), and (7) together prove the theorem. ■