Split Null Keys: A Null Space Based Defense for Pollution Attacks in Wireless Network Coding

Andrew Newell Cristina Nita-Rotaru
Department of Computer Science, Purdue University
305 N. University St., West Lafayette, IN 47907 USA
{newella,crisn}@cs.purdue.edu

***Abstract*—Recent work in defending against pollution attacks for intra-flow network coding systems proposed a null spaces based algebraic approach which has a smaller computation cost than previous pollution defenses. The approach requires the source to distribute keys periodically, but in order to scale involves forwarder nodes in the creation of new keys and their distribution. As a result the key distribution is secure only in specific network topologies such as those created by large-scale peer to peer systems, and is not secure in wireless networks where such topologies do not exist. We propose Split Null Keys, which splits the keys such that only a small portion of the key is updated periodically. The small updates allow for a scalable key distribution scheme that does not involve forwarder nodes in creating keys and thus does not rely its security on constraints imposed on the network topology. We prove that our scheme is secure despite splitting the key and we show that when compared with existing defenses our scheme imposes lower communication and computation overhead, is resilient to colluding adversaries, and does not require time synchronization.**

I. Introduction

Network coding routing deviates from traditional store-and-forward routing by allowing intermediate nodes to code packets together. Network coding is particularly applicable in wireless networks where the broadcast nature and opportunistic reception of the wireless medium allows network coding to surpass traditional routing by taking advantage of any overheard packets. Numerous practical systems [1], [2], [3], [4], [5], [6] have been proposed for wireless networks.

Network coding systems are vulnerable to pollution attacks [7] in which adversaries acting as intermediate nodes inject bogus packets into the network. The injection of polluted packets can also occur in traditional store-and-forward routing protocols. In this case, since intermediate nodes just forward packets, any scheme that provides data source authentication (such as digital signatures) is an effective defense in detecting packets that were not created by the source. In random network coding, intermediate nodes code new packets by computing random linear combinations of the packets received from upstream nodes. In this case, traditional data source authentication mechanisms are not applicable. Such authentication schemes need to have homomorphic properties in order to allow intermediate nodes to verify that the packets they are coding are in turn linear combinations of packets that originated at the source. Even more, while pollution attacks require little resources from the attacker they have an epidemic effect in network coding systems as honest intermediate nodes unknowingly amplify the attack by creating new packets based on the received bogus packets and forwarding the resulting new malformed packets in the network.

Several defenses exist for pollution attacks relying on cryptographic, information theory, or algebraic mechanisms. *Cryptographic-based schemes* [8], [9], [10], [11], [12] create homomorphic digital signatures and hashes. While such schemes are effective in peer-to-peer systems, they impose prohibitive communication and computation overhead in wireless networks. A cryptographic solution based on MACs [13] imposes prohibitive overhead in the presence of multiple byzantine adversaries. *Information theoretical-based schemes* [14], [15] code redundant information into packets, allowing receivers to recover correct packets in spite of polluted packets. Such approaches hinder the system performance as they limit the throughput of the network coding system based on the adversary's available bandwidth or impose restrictions on broadcasts of intermediate nodes.

Algebraic-based schemes verify that packets received by forwarders belong to the space defined by the original packets sent by the source. Two representative approaches are the schemes in [16] and [17]. The scheme in [16] creates non-cryptographic checksums and relies its security on the difference between the time when a packet was received and the time when the checksum used to verify the packet was created. The scheme is effective but requires time synchronization and delays packets before forwarding them. The scheme in [17] uses null space properties to provide nodes with vectors (referred to as null keys) belonging to the null space of source packets that are used to algebraically verify that the packets belong to the same space as the source packets. Because the defined null keys are large and would impose high load on the source and high communication in the network, the source distributes null keys only to first hop neighbors and relies on the homomorphic property of the null keys to have intermediate nodes create null keys for their downstream nodes. Thus, the scheme relies its security on path diversity, to ensure that each node will have a null key that spans a space much larger than any one adversary can know about. Path diversity is possible in peer-to-peer networks because links can be inserted and deleted easily. However, this assumption is not valid in wireless networks where there is less path diversity and where the topology is optimized based on the wireless link qualities, making the scheme insecure in wireless networks.

We propose a new defense against pollution attacks based on the null space properties and without relying on any assumptions about the network topology or time synchronization. Specifically:

- We propose Split Null Keys (SNK), a new defense against pollution attacks that splits a null key in two components, a small generation dependent one and a larger generation independent one chosen randomly[1]. As a result, after an initialization phase when the generation independent component is distributed, only a small portion of a null key (160 bytes) that is dependent on the data from each generation must be updated for each generation. The small communication overhead allows the source to securely distribute the update individually to each forwarder. Since each forwarder receives its own update securely an attacker cannot exploit the knowledge of the null key, and thus no path diversity is required. SNK has a smaller communication cost per generation than previous work and a very small computation cost which consists of inexpensive matrix multiplications. Our scheme also does not delay packets for verification and scales with the number of colluding adversaries in the network.
- We formally prove that a probabilistic polynomial time adversary that can control any set of byzantine forwarder nodes and overhear all communication in the network, cannot pollute a target victim node. The intuition is that the large, generation independent portion of the null key serves as a secret between the source and forwarder, so keeping this portion constant across multiple generations does not deteriorate security as long as it remains secret. Even if forwarders collude the adversary cannot know how the null key of the target victim node and the null keys known by the adversary overlap due to the fact that all null keys are generated independently and randomly.
- We validate the performance and overhead of our scheme with extensive simulations using a well-known network coding system for wireless networks (MORE [1]) and realistic link quality measurements from the Roofnet [18] experimental testbed. Our results show that SNK imposes little communication overhead with an average of 25 kbps to distribute null keys. SNK outperforms previous defenses against pollution attacks in both benign and adversarial networks achieving better throughput and latency. Finally, SNK retains the benefits of network coding by performing better than a traditional, secure, store-and-forward routing protocol ARAN (a secure version of the well-known shortest path routing protocol AODV).

II. Related Work

Detecting polluted packets at intermediate nodes. Several homomorphic signature schemes proposed to provide a verification function for intermediate nodes in the network. The works in [8], [9], [10], [11], [12] utilize cryptographic primitives that rely on the discrete logarithm problem for security, which causes two major performance issues. A lower bound is enforced on the symbol size, and a high computational overhead is imposed by numerous modular exponentiations or elliptical curve operations. Work has been done to address the computational overhead of cryptographic pollution defenses at intermediate nodes. Zhao *et al.* [19] proposes to speed up computations by utilizing graphical processing units while Gkantsidis *et al.* [20] proposes to probabilistically verify received blocks in peer-to-peer networks. Gennaro *et al.* [21] show that the scheme [8] could use smaller coefficients near the source to reduce computational and communication overhead. The coefficients become larger with each hop and eventually approach the overhead imposed by previous cryptographic schemes, so topologies with many hops will still suffer.

Boneh *et al.* [22] propose the first homomorphic signature scheme over binary field sizes by using lattice-based techniques. Their construction limits the number of times signatures can be combined, and the key sizes are much larger than traditional cryptographic constructions.

Agrawal *et al.* [13] present a homomorphic MAC that relies on pseudo-random functions to overcome performance limitations. A work [23] shows an efficient way to overcome a problem unique to homomorphic MACs known as *tag pollution*. However, the underlying scheme still does not scale with the number of attackers in the network.

Dong *et al.* [16] propose a protocol for wireless networks that relies on checksums being disseminated periodically throughout the network. Attackers cannot conduct a forgery attack by observing a checksum because intermediate nodes verify received packets against checksums that were created at the source at a later time than the time when packets were received by intermediate nodes. The scheme has a lower overhead than cryptographic defenses but causes coded packets to be delayed before being verified.

Kehdi *et al.* [17] propose an algebraic based approach that uses null space properties to defend against pollution attacks. The scheme is suitable for large peer-to-peer networks with path diversity but not applicable in wireless networks where such diversity cannot be guaranteed. Our scheme is also based on null space properties but does not rely on path diversity and has a small communication overhead per generation.

Ensuring reconstruction of correct packets at receivers. The work [14] encodes redundant information to reconstruct the valid coded packets at the receiver in the presence of a byzantine adversary. However, the throughput of the network is dependent on the adversary's network capacity to the receiver, so an adversary with high network capacity can potentially reduce the throughput of the network to zero. The work [15] proposes to limit nodes' network capacity by limiting the broadcasts of each node to ensure the scheme [14] retains high throughput in the presence of adversaries. Limiting broadcasts is inconsistent with practical wireless network coding systems.

Identifying polluting attackers. The work [24] uses the homomorphic MACs [13] to determine the subspaces a node has received and the subspaces a node has forwarded. This is sufficient information to determine if a node is a pollution

[1]In a network coding scheme the source disseminates the entire sequence of packets in sub-sequences called *generations*.

TABLE I
NOTATION

Name	Description
n	Number of plain packets per generation
m	Number of symbols per plain packet
q	Field for a symbol, the symbol size is $\log_2(q)$
$\mathbf{X}$	Data matrix of plain packets, size n by m
$\mathbf{I}$	Identity matrix
$\mathbf{A}$	Augmented data matrix of size n by $n+m$ $\mathbf{A} = [\mathbf{I}\|\mathbf{X}]$
$\mathbf{c}$	Coded packet, it is an element of the row space of $\mathbf{A}$
$\mathbf{V}$	Coding header at a destination used for decoding
$\mathbf{B}$	Null space matrix of $\mathbf{A}$ which has size $n+m$ by m
$\mathbf{0}$	Matrix of all zeros
$\mathbf{K}_i$	Null key for forwarder i which is subspace of the column space of $\mathbf{B}$
ω	Security parameter for the number of null keys a forwarder uses for verification
$\tilde{\mathbf{K}}_i$	Generation dependent null key, first n rows of $\mathbf{K}_i$
$\bar{\mathbf{K}}_i$	Generation independent null key, last m columns of $\mathbf{K}_i$
$\mathbf{S}$	First n rows of $\mathbf{B}$
$\mathbf{T}$	Last m rows of $\mathbf{B}$
$\mathbf{G}_i$	Null key generator for null key $\mathbf{K}_i$, $\mathbf{B} * \mathbf{G}_i = \mathbf{K}_i$
θ	Number of possible forwarders for a source
β	Number of forwarders for a flow

attacker. A work based on monitoring [25] is able to detect whether a node is polluting with high probability given that nodes protect the headers of coded packets with error correcting codes and that multiple honest watchdogs exist per node in the network.The work [26] proposes a monitoring technique that requires source encoding where the amount of overhead is dependent on the channel qualities in the network. The adversary has a higher probability of being detected because it has to pollute many packets to overcome the source encoding.

III. System model

We describe the network coding system and adversarial model. The notation we use is presented in Table I.

A. Network Coding System

We assume an intra-flow network coding system with one *source* that sends data via *forwarders* to one or more *destinations*. The source sends data in *generations*. A generation represents a subsequence of packets from the total number of packets and consists of n *plain packets*. A plain packet consists of m *symbols* which are elements of the finite field $\mathbb{F}_q$ (each symbol is of size $\log_2(q)$ bits). The plain packets are encoded in a *data matrix* $\mathbf{X}$ of size n by m such that each row is a plain packet. The matrix $\mathbf{X}$ is augmented by the identity matrix $\mathbf{I}$ to form an *augmented data matrix* $\mathbf{A} = [\mathbf{I}|\mathbf{X}]$. The identity matrix is inserted to serve as a *coding header* for decoding the coded packets at a destination.

The source creates *coded packets* $\mathbf{c}$ by generating random vectors that belong to the row space of $\mathbf{A}$ and sends these coded packets to forwarders and destinations that store them in a *coding buffer*. Forwarders create coded packets by generating random vectors from their coding buffer. Destinations eventually obtain a coding buffer spanning the same space as the row space of $\mathbf{A}$, i.e., each destination has $[\mathbf{V}|\mathbf{V}*\mathbf{X}]$ where $\mathbf{V}$ is the coding header and has full rank. The destination decodes the packets by computing $\mathbf{V}^{-1} * [\mathbf{V}|\mathbf{V}*\mathbf{X}] = [\mathbf{V}^{-1}*\mathbf{V}|\mathbf{V}^{-1}*\mathbf{V}*\mathbf{X}] = [\mathbf{I}|\mathbf{X}]$ to obtain the original data matrix $\mathbf{X}$.

Parameter selection. The selection of parameters n and m impacts performance of a network coding system. The parameter n must be set to ensure a sufficient number of packets are coded together to obtain network coding gains. However, n affects *coding overhead* which is the overhead for distributing the coding header. Each generation contains a data matrix, $\mathbf{X}$, that is $n*m*\log_2(q)$ bits, and each generation the source distributes a larger, augmented data matrix, $\mathbf{A}$, that is $n^2*\log_2(q)+n*m*\log_2(q)$ bits. The extra $n^2*\log_2(q)$ bits distributed are coding overhead. Thus, the selection of n and m must ensure that $n << m$ to minimize coding overhead.

B. Adversarial Model

We assume that an attacker mounts pollution attacks by injecting *polluted coded packets* in the network. A polluted packet is a coded packet that is not an element of the row space of $\mathbf{A}$. Nodes downstream from the attacker accept this packet as valid and store it in their coding buffer. Forwarders with polluted coded packets in their coding buffers will create new coded packets that are also polluted. Thus, the forwarders unknowingly act as pollution attackers themselves, and the attack propagates epidemically throughout the network. Destinations with polluted coded packets in their coding buffers will not obtain the data sent by the source upon decoding. An attacker can be any node in the network, a rogue node without the credentials to be part of the network or a node with the credentials to be in the network but was compromised and controlled by an adversary. We assume that multiple attackers exist and they can collude.

IV. Split Null Keys (SNK)

We first overview the null space properties that our scheme relies on and then describe our scheme. In the following, the term *forwarders* also refers to destinations.

A. Null Space Properties

Let the null space of the row space of the matrix $\mathbf{A}$ (of size n by $n+m$) be the column space of $\mathbf{B}$, then we have:

$$\mathbf{A} * \mathbf{B} = \mathbf{0}$$

and $\mathbf{B}$ is a basis for the null space of the row space of $\mathbf{A}$.

According to the rank nullity theorem

$$r(\mathbf{A}) + r(\mathbf{B}) = n + m \Rightarrow r(\mathbf{B}) = m$$

so the rank of the column space of $\mathbf{B}$, $r(\mathbf{B})$, is m. Thus, $\mathbf{B}$ is a matrix of size $(n+m)$ by m.

Definition A *null key* is a matrix that spans a subspace of the column space of $\mathbf{B}$. We denote a null key by $\mathbf{K}$.

We now show three properties for null keys in relation to valid coded packets and polluted coded packets.

Lemma 1: A valid coded packet multiplied by a null key always equals a zero vector.

Proof: By definition, any vector of the row space of $\mathbf{A}$ multiplied with any vector of the column space of $\mathbf{B}$ results in a zero. ■

Lemma 2: A randomly generated coded packet $\mathbf{c}$ has a probability of $(\frac{1}{q})^{\omega}$ to satisfy $\mathbf{c} * \mathbf{K} = \mathbf{0}$ where $\mathbf{K}$ is a null key with rank ω and q is the symbol size.

Proof: Let K be the column space of $\mathbf{K}$. The probability that a randomly chosen coded packet $\mathbf{c}$ yields $\mathbf{c} * \mathbf{K} = \mathbf{0}$ is equivalent to the probability that a randomly chosen coded packet is a vector that is in the null space K. The null space of K is the space of all vectors $\mathbf{c}'$ such that $\mathbf{c}' * \mathbf{K} = \mathbf{0}$. The rank of K is ω, so the rank of the null space of K is $n+m-\omega$ according to the rank nullity theorem. The number of vectors in the null space of K is $q^{n+m-\omega}$, and the number of possible coded packets is q^{n+m}. Thus, the probability that a randomly chosen coded packet is a vector that is in the null space K is $\frac{q^{n+m-\omega}}{q^{n+m}} = (\frac{1}{q})^{\omega}$. ■

Lemma 3: Let $\mathbf{K}'$ be a matrix that represents a subspace of the column space of the null key $\mathbf{K}$ where $\mathbf{K}'$ has rank ω' and $\mathbf{K}$ has rank ω ($\omega' \leq \omega$). Then, a randomly selected coded packet $\mathbf{c}$ from the set of coded packets that satisfy $\mathbf{c} * \mathbf{K}' = \mathbf{0}$ has a probability of $(\frac{1}{q})^{(\omega-\omega')}$ to satisfy $\mathbf{c} * \mathbf{K} = \mathbf{0}$.

Proof: Let the column space of $\mathbf{K}$ and $\mathbf{K}'$ be denoted by K and K' respectively. The ranks of the null spaces of K and K' are $n+m-\omega$ and $n+m-\omega'$ respectively. The number of vectors in the null spaces of K and K' are $q^{n+m-\omega}$ and $q^{n+m-\omega'}$ respectively. Given that $\mathbf{K}'$ is a linear combination of the vectors of $\mathbf{K}$ we have that any coded packet $\mathbf{c}'$ that satisfies $\mathbf{c}' * \mathbf{K} = \mathbf{0}$ also satisfies $\mathbf{c}' * \mathbf{K}' = \mathbf{0}$, so null space of K is a subset of the null space of K'. A randomly selected coded packet $\mathbf{c}$ from the null space of K' has a probability of $\frac{q^{n+m-\omega}}{q^{n+m-\omega'}} = (\frac{1}{q})^{(\omega-\omega')}$ to satisfy $\mathbf{c} * \mathbf{K} = \mathbf{0}$. ■

Using null keys to detect pollution Based on Lemma 1 and Lemma 2, polluted packets can be identified as follows. A forwarder i having the null key $\mathbf{K}_i$ and receiving a coded packet $\mathbf{c}$ will compute $\mathbf{c} * \mathbf{K}_i$. If the result is a zero vector then the coded packet is accepted, otherwise the coded packet is dropped. In the case the packet is accepted, there is low probability that the packet may still be polluted, $(\frac{1}{q})^{\omega}$ if the packet is chosen randomly according to Lemma 2 which is controlled by the column rank of the null key, ω. We show in Section V that an attacker cannot do better than generating polluted coded packets randomly when attempting to pass a victim node's verification test.

Impact on security when dimensions of null keys overlap. If the dimensions of the null keys at two forwarders overlap, and a malicious forwarder knows the dimensions that overlap, then the malicious forwarder can pollute the other forwarder with a high probability given in Lemma 3. The higher the overlap, the higher the success of crafting a polluted packet. Thus, it is essential that an attacker does not know the dimensions that overlap between their null key and the null keys at other honest forwarders in the network.

B. SNK Overview

As a null key is a matrix that is a subspace of the column space of $\mathbf{B}$ (which is an $(n+m)$ by m matrix), the size of a null key for a forwarder is $(n+m)$ by ω, where ω is the column rank of $\mathbf{K}_i$. As stated in Lemma 3 the dimensions of null keys should not overlap, so if the source distributes all the null keys, this results in a very high communication overhead. Typical settings for wireless networks are $q = 256$ (1 byte symbols), $n = 32$, and $m = 1468$ ($n + m = 1500$ typical wireless packet size), $w = 5$ to prevent random guessing (Lemma 2) so even with few forwarders in the network, the source will spend more time sending null keys than data.

In previous work [17] it was proposed to reduce this communication overhead by having the source send null keys only to the first hop nodes and rely on forwarder nodes to generate null keys for downstream nodes by combining null keys from upstream nodes. Such an approach scales well with large networks, but it makes a critical assumption, that there is *enough path diversity* such that a malicious forwarder cannot know the dimensions that overlap with null keys at other forwarders. An attacker that knows which dimensions overlap can easily craft a polluted packet that passes a legitimate forwarder's verification test with high probability or even 1 according to Lemma 3. Such path diversity cannot be guaranteed in wireless networks.

Given the lack of path diversity in wireless network, we cannot rely on forwarders to create new null keys for downstream nodes. At the same time, the size of a null key is large preventing a source from sending individual keys to each forwarder. Our scheme, Split Null Keys (SNK), is based on the observation that only a small portion of a null key for a generation is dependent on the data for that generation while the remaining, larger portion of the null key is chosen randomly. The large, random portion serves as a secret between the source and forwarder, so keeping this portion constant across multiple generations does not deteriorate security as long as it remains secret. SNK splits a null key in two components: a component that is generation independent and sent only at system initialization, and a component that is generation dependent and sent every generation. Specifically, for each null key, the generation dependent component has a size of $\omega * n * \log_2(q)$ bits, and the generation independent has a size of $\omega * m * \log_2(q)$ bits. Thus, every generation, for a forwarder, our scheme needs to send only $\omega * n * \log_2(q)$ bits, while if the scheme [17] is used, the entire null key of size at least $(n+m) * \log_2(q)$ bits needs to be updated (null keys for this scheme are sometimes larger based on the topology). The source generates and distributes the null keys for each forwarder, in a secure manner[2], so SNK does not rely on path diversity of the network topology.

At a high level, SNK works as follows (see also Algorithm 1). In the initialization step which is performed only once, the source creates and distributes the generation independent null keys to forwarders. In the update step which is performed every generation, the source calculates and distributes the generation dependent null keys for a new generation represented by the data matrix $\mathbf{X}$ for each forwarder in the flow. In the verifying step which is performed every time

[2]Note that nodes should share symmetric keys with the source in order to have end-to-end data integrity and confidentiality; a basic service for any communication protocol.

Algorithm 1 SNK

```
Initialization (generation independent): Source initializes a network with forwarders $f_1,...,f_\theta$
1: Randomly select null key generators $\mathbf{G}_1,...,\mathbf{G}_\theta$
2: Calculate $\bar{\mathbf{K}}_i = \mathbf{G}_i$ for $i = 1,...,\theta$
3: Distribute $\bar{\mathbf{K}}_i$ to forwarder $i$ for $i = 1,...,\theta$
Null key update (generation dependent): Source generates update keys for a generation consisting of data matrix X for a flow with forwarders $f_1,...,f_\beta$
1: Calculate $\tilde{\mathbf{K}}_i = -\mathbf{X} * \mathbf{G}_i$ for $i = f_1,...,f_\beta$
2: Distribute $\tilde{\mathbf{K}}_i$ to forwarder $i$ for $i = f_1,...,f_\beta$
Verification (per packet): Forwarder $f$ verifies a coded packet $\mathbf{c}$
1: Form null key $\mathbf{K}_f$ from $\tilde{\mathbf{K}}_f$ and $\bar{\mathbf{K}}_f$, $\mathbf{K}_f = \begin{bmatrix} \tilde{\mathbf{K}}_f \\ \bar{\mathbf{K}}_f \end{bmatrix}$
2: Verify that $\mathbf{c} * \mathbf{K}_f = \mathbf{0}$
```

a packet is received, a forwarder forms its null key from the received null key parts and verifies a received coded packet.

C. Null Keys Splitting Procedure

We split each null key into two parts $\mathbf{K}_i = \begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix}$, a *generation dependent null key* ($\tilde{\mathbf{K}}_i$) and a *generation independent null key* ($\bar{\mathbf{K}}_i$). The first n rows of the null key are the generation dependent portion, while the remaining m rows are the generation independent portion. Generation independent null keys are updated once for multiple generations while generation dependent null keys are updated every generation. As in a typical network coding system $n << m$, ensuring that the generation dependent portion of a null key has n rows reduces the overhead significantly.

In order to ensure that the generation independent null key component remains constant across multiple generations we split $\mathbf{B}$ as follows. Let $\mathbf{S}$ be an n by m matrix and $\mathbf{T}$ be an m by m matrix such that $\mathbf{B} = \begin{bmatrix} \mathbf{S} \\ \mathbf{T} \end{bmatrix}$. The source keeps $\mathbf{T}$ constant for each generation and computes a new $\mathbf{S}$ in order to satisfy the null space property that $\mathbf{A} * \mathbf{B} = \mathbf{0}$. Let $\mathbf{T} = \mathbf{I}$ for each generation:

$$\begin{aligned} \mathbf{A} * \mathbf{B} = \mathbf{0} \quad &\Rightarrow \quad [\mathbf{I}|\mathbf{X}] * \begin{bmatrix} \mathbf{S} \\ \mathbf{T} \end{bmatrix} = \mathbf{0} \quad \Rightarrow \quad \mathbf{I} * \mathbf{S} + \mathbf{X} * \mathbf{T} = \mathbf{0} \\ &\Rightarrow \quad \mathbf{S} + \mathbf{X} * \mathbf{T} = \mathbf{0} \qquad \Rightarrow \quad \mathbf{S} = -\mathbf{X} * \mathbf{T} \\ &\Rightarrow \quad \mathbf{S} = -\mathbf{X} * \mathbf{I} \qquad\quad \Rightarrow \quad \mathbf{S} = -\mathbf{X} \end{aligned}$$

Thus, by choosing $\mathbf{T} = \mathbf{I}$, we obtain $\mathbf{B} = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix}$.

A null key is a random subspace of the column space of $\mathbf{B}$. To ensure that a generation independent null key remains constant for multiple generations, a *null key generator* is selected for each null key, and the null key generator remains constant for multiple generations. The null key generator is a random matrix $\mathbf{G}_i$ of size m by ω with full column rank. A null key is computed as $\mathbf{K}_i = \mathbf{B} * \mathbf{G}_i$.

We show that if we choose $\mathbf{T} = \mathbf{I}$, which in turn means $\mathbf{B} = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix}$, then only the generation dependent null key is dependent on $\mathbf{X}$:

$$\begin{aligned} \begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix} \quad &= \quad \mathbf{K}_i = \mathbf{B} * \mathbf{G}_i = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix} * \mathbf{G}_i = \begin{bmatrix} -\mathbf{X} * \mathbf{G}_i \\ \mathbf{G}_i \end{bmatrix} \\ &\Rightarrow \quad \tilde{\mathbf{K}}_i = -\mathbf{X} * \mathbf{G}_i \qquad \bar{\mathbf{K}}_i = \mathbf{G}_i \end{aligned}$$

We summarize the splitting algorithm. For each forwarder i the source generates a random matrix $\mathbf{G}_i$ of size m by ω with full column rank. Then, the source computes the generation independent null key $\bar{\mathbf{K}}_i$ as $\bar{\mathbf{K}}_i = \mathbf{G}_i$ and the generation dependent null key $\tilde{\mathbf{K}}_i$ computed for each generation with data $\mathbf{X}$ as $\tilde{\mathbf{K}}_i = -(\mathbf{X} * \mathbf{G}_i)$. Thus, the null key remains constant for multiple generations and the generation dependent null key depends on the data matrix of each generation. Each forwarder recreates its null key as $\begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix} = \mathbf{K}_i$.

D. Null Key Distribution and Verification

Distribution. An adversary that knows the null keys for a legitimate node can form coded packets that pass the legitimate node's verification test. Thus, the source distributes null keys to each forwarder over confidential and authenticated channels. We justify the use of these secure channels as we will show in Section VI that our approach incurs significantly less overhead than previous cryptographic approaches which do not require secure channels with forwarders. Each forwarder shares a unique symmetric key with the source which can be setup before distributing generation independent null keys. For each generation the source generates a *null key packet* with the contents $\langle i||GID||Enc(\tilde{\mathbf{K}}_i)||MAC(i||GID||Enc(\tilde{\mathbf{K}}_i))\rangle$ where GID is an identifier for the generation, $Enc()$ is a block cipher encryption such as AES in CBC mode, and $MAC()$ is a message authentication code such as HMAC with SHA-1. A null key packet is sent on a multi-hop best path from the source to each forwarder of the flow for each generation.

Motivation for encrypting $\tilde{\mathbf{K}}_i$. When distributing the generation dependent null key $\tilde{\mathbf{K}}_i$ from the source to a forwarder it is necessary to encrypt it such that no other forwarder can decrypt the value. If the generation dependent null key were not encrypted, then a subtle attack exists where an attacker can exploit the knowledge of $\tilde{\mathbf{K}}_i$ over many generations. Given that $\tilde{\mathbf{K}}_i$ is sent in the clear, the attacker obtains n unique equations in a system of m unknowns, $\tilde{\mathbf{K}}_i * \bar{\mathbf{K}}_i = -\mathbf{X}$, each generation. After $\lceil \frac{m}{n} \rceil$ generations, the attacker obtains enough equations to compute the value $\bar{\mathbf{K}}_i$. With the entire contents of a victim node's null key, the attacker can craft polluted coded packets that pass a victim node's verification test. However, it is easy to prevent this subtle attack by encrypting each $\tilde{\mathbf{K}}_i$, and we do this as part of our protocol.

Verification. Given that a forwarder i has null key $\mathbf{K}_i$, the forwarder verifies packet $\mathbf{c}$ by checking if $\mathbf{c} * \mathbf{K}_i = \mathbf{0}$. According to Lemma 1, a valid packet will pass verification. Without the knowledge of $\mathbf{K}_i$ or any dimensions of $\mathbf{K}_i$, an attacker can conduct an attack only by randomly generating polluted coded packets. The probability that such a random polluted packet does pass the verification test is negligible for typical wireless network coding settings and is given in Lemma 2. In our scheme, because the source (which is trusted by all nodes) is the only one generating null keys and because these null key components are disseminated in a secure manner, the attacker cannot gain any knowledge of $\mathbf{K}_i$ or any dimensions of $\mathbf{K}_i$ and thus cannot improve their probability of polluting node i.

V. Security Analysis

We show security in the extreme case where an adversary overhears all communication in the network for any number of previous generations and compromises all forwarders in the network except a node that is the target of the attack. [3] We formalize the information available to the attack along with the attacker's goal of crafting a polluted coded packet that passes a victim node's verification test in Game 1. We are able to show that the attacker can only wins this game with negligible probability. According to our attack game, the scheme in [17] is insecure since the adversary will win Game 1 because the knowledge of null keys upstream of the target node allows those nodes to pollute the target node.

Theorem 1 states that if both the cipher used to encrypt the null keys and the PRG (Pseudo Random Generator) used to generate the null keys cannot be broken then the probability that an attacker will win the game is the probability that an attacker guesses and passes verification which is negligible for typical wireless network coding systems. Note that both the generation independent and dependent portions of the null key are encrypted such that an attacker must break the cipher to obtain any knoweldge of null keys sent to the victim. The proof of this theorem uses Lemma 4 (proof omitted due to space) along with an algebraic argument to show that the best strategy available to an adversary in Game 1 is to randomly guess coded packets.

Game 1 Pollution attack game for SNK

Game between a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. Parameters are (E, R, q, n, m, ω) E is a cipher, R is a PRG, and the other parameters are the same as in SNK. **Setup:**

```
1: $\mathscr{C}$ generates a random key $k$ and a random seed $s$.
2: $\mathscr{C}$ computes $\mathbf{G}_l$ for each forwarder $l$ using $R(s)$.
3: $\mathscr{C}$ computes $\bar{\mathbf{K}}_l = \mathbf{G}_l$ for each forwarder $l$.
4: $\mathscr{C}$ computes $\tilde{\mathbf{K}}_l(j) = -\mathbf{X}(j) * \mathbf{G}_l$ for each forwarder $l$ and each generation $j$.
5: $\mathscr{C}$ chooses some target forwarder $i$.
```

Queries:

```
1: $\mathscr{A}$ can request the target forwarder $i$. $\mathscr{C}$ responds with $i$.
2: $\mathscr{A}$ can request the encrypted generation dependent key for a given generation $j$. $\mathscr{C}$ responds with $E_k(\tilde{\mathbf{K}}_i(j))$.
3: $\mathscr{A}$ can request the encrypted generation independent key. $\mathscr{C}$ responds with $E_k(\bar{\mathbf{K}}_i)$.
4: $\mathscr{A}$ can request the null key $\mathbf{K}_l(j)$ of any node $l$ s.t. $l \neq i$ and any generation $j$. $\mathscr{C}$ responds with $\mathbf{K}_l(j)$.
5: $\mathscr{A}$ can request the data for generation $j$. $\mathscr{C}$ responds with $\mathbf{X}(j)$.
```

Output:

```
1: $\mathscr{A}$ must output a generation identifier $j$ and coded packet $\mathbf{c} = [\mathbf{v}|\mathbf{x}]$. $\mathscr{A}$ wins the game if $\mathbf{v} * \mathbf{X}(j) \neq \mathbf{x}$ and $\mathbf{c} * \mathbf{K}_i(j) = \mathbf{0}$.
```

Lemma 4: Consider a Probabilistic Polynomial-Time (PPT) adversary $\mathscr{A}_1$ that knows the data for a generation $\mathbf{X}$ and must produce a polluted coded packet $\mathbf{c}$ such that $\mathbf{K}_i * \mathbf{c} = \mathbf{0}$. Given that $\mathbf{G}_i$ are chosen truly randomly for each forwarder, $\mathscr{A}_1$ gains no advantage if it has knowledge of z null keys $\mathbf{K}_j$ where $j \neq i$ and z is bounded by a polynomial.

[3]We do not consider an adversary that attempts to modify null key packets sent to the target node from the source. We assume that the MAC that is attached to each null key packet is sufficient to protect against such modification. This can be formally shown with additional attack games and theorems, but we omit this for space considerations.

Theorem 1: SNK with parameters (E, R, q, n, m, ω) is secure as long as the cipher and PRG are secure. More specifically, for an adversary $\mathscr{A}$ that cannot break the cipher or PRG, the adversary has a probability of $(\frac{1}{q})^\omega$ to win Game 1.

Proof: With a secure cipher, the adversary learns nothing by querying the challenger for encrypted versions of i's generation dependent or independent null keys, so queries 2 and 3 are not advantageous to the adversary. With a secure PRG and the result of Lemma 4, the adversary cannot infer the value of $\mathbf{G}_i$ from the values of $\mathbf{G}_l$ for $i \neq l$, so query 4 is not advantageous to the adversary. The remaining queries, 1 and 5, allow the adversary to learn i and $\mathbf{X}(j)$ to help the adversary choose a $\mathbf{c}$ and j such that:

$$\begin{aligned}
\mathbf{c} * \mathbf{K}_i(j) = \mathbf{0} \quad &\Rightarrow \quad [\mathbf{v}|\mathbf{x}] * \begin{bmatrix} \tilde{\mathbf{K}}_i(j) \\ \bar{\mathbf{K}}_i \end{bmatrix} = \mathbf{0} \\
&\Rightarrow \quad \mathbf{v} * \tilde{\mathbf{K}}_i(j) + \mathbf{x} * \bar{\mathbf{K}}_i = \mathbf{0} \\
&\Rightarrow \quad \mathbf{v} * (-\mathbf{X}(j) * \mathbf{G}_i) + \mathbf{x} * \mathbf{G}_i = \mathbf{0} \\
&\Rightarrow \quad \mathbf{x} * \mathbf{G}_i - \mathbf{v} * \mathbf{X}(j) * \mathbf{G}_i = \mathbf{0} \\
&\Rightarrow \quad (\mathbf{x} - \mathbf{v} * \mathbf{X}(j)) * \mathbf{G}_i = \mathbf{0}
\end{aligned}$$

Let $\mathbf{a} = \mathbf{x} - \mathbf{v} * \mathbf{X}(j)$. The adversary can freely choose any $\mathbf{a}$ by choosing an arbitrary $\mathbf{v}$, and then setting $\mathbf{x}$ to $\mathbf{a} + \mathbf{v} * \mathbf{X}(j)$. With this substitution, the adversaries goal is reduced to choosing an $\mathbf{a}$ such that $\mathbf{a} * \mathbf{G}_i = \mathbf{0}$.

The trivial solution for choosing $\mathbf{a}$ is to let $\mathbf{a} = \mathbf{0}$, but this violates the $\mathbf{v} * \mathbf{X} \neq \mathbf{x}$ condition of winning the game. Thus, the adversary must choose an $\mathbf{a}$ such that $\mathbf{a} \neq \mathbf{0}$ and $\mathbf{a} * \mathbf{G}_i = \mathbf{0}$. Given that $\mathbf{G}_i$ is unknown to the adversary and is completely random in Game 1, any choice of $\mathbf{a}$ by the adversary will result in $\mathbf{a} * \mathbf{G}_i$ being a random vector of ω elements. Each element of this vector takes a random element from a field of q elements. Thus, the probability that $\mathbf{a} * \mathbf{G}_i = \mathbf{0}$ is $(\frac{1}{q})^\omega$. ∎

VI. Evaluation

In this section, we compare the performance and overhead of SNK with other pollution defenses and a secure, store-and-forward routing protocol.

A. Simulation Methodology

Our experiments are conducted using the Glomosim [27] simulator with an implementation of the MORE [1] wireless network coding system. We use 802.11 [28] with a raw link bandwidth of 5.5 Mbps. For our topology, we use the link quality measurements from Roofnet [18] which is a 38-node 802.11b/g mesh network. For each simulation, we setup a random flow in the network by selecting two random nodes as the source and the destination; the source transmits for 400 seconds. We select 200 random flows and conduct the simulation once for each flow and protocol.

Metrics. We measure *throughput* as the rate (in kbps) of data being decoded at the destination. We measure *latency* as the time between the start of the source transferring the first generation to decoding of the generation at the destination. We measure *communication overhead* as the total summed rate (in kbps) of overhead data broadcasted by all nodes. Data that does not belong to a standard network coding system is overhead data which are checksums, MACs, and null keys.

To demonstrate the efficacy of our scheme, we compare it with previous defenses against pollution attacks, all implemented in the MORE system. We compare SNK with the insecure MORE system, two representative cryptographic schemes KFM [9] and HOMOMAC [13], and two algebraic schemes DART [16] and EDART [16]. To show that SNK is practical, we also compare it with a secure traditional routing protocol, ARAN [29]. We do not compare with the scheme in [17] since as described in Section IV such a scheme will not be secure in wireless networks.

We consider communication and computation overhead of each scheme. SNK sends generation dependent null key packets each generation, DART and EDART send checksum packets during generations, HOMOMAC appends MACs to coded packets, and KFM requires heavy computations.

Parameter selection. We select the network coding parameters to match the default settings for MORE in [1]. The size of a generation is $n = 32$, a symbol size of 1 byte $q = 2^8$, and the size of a coded packet is 1500 bytes. These parameters are the same for each scheme with the exception of KFM which requires a larger symbol size to ensure the intractability of the discrete logarithm problem. DART and EDART are configured to ensure their best performance, as in [16].

Attack settings. We select defense parameters for each defense scheme to ensure the same strength of $(\frac{1}{2})^{40}$ where the strength corresponds to the probability that the verification mechanism accepts a polluted coded packet. The rank of null keys, size of checksums, and number of MACs are selected appropriately for SNK, DART/EDART, and HOMOMAC respectively. We cannot ensure such strength for the adaptive defense scheme EDART as it purposely forwards some coded packets without verifying to reduce the delay imposed by DART. A pollution attacker broadcasts a polluted coded packet for every 5 coded packets it receives. The polluted coded packets are generated randomly as there is no better strategy for selecting polluted coded packets for these schemes given that the underlying security assumptions hold.

B. Performance Evaluation

We compare with two other proactive defenses KFM and DART. The proactive schemes verify every coded packet independent of the number of forwarders, so their overhead is the same in adversarial and benign networks. We include the insecure system MORE as a baseline for comparison.

From the results shown in Figure 1(a), SNK outperforms DART by over 100 kbps in the lowest (according to throughput) 50% of flows due to the lowest flows having larger number of hops from the source to destination. This pattern shows that DART's performance diminishes more than SNK as more hops exist between the source and destination due to the delaying of each coded packet for verification. Despite the similar throughput of SNK and DART, Figure 1(b) shows that DART imposes 5 times the latency compared to SNK. The increased latency of DART is due to the pipelining of 5 generations that is necessary to mitigate the delayed verification of packets and achieve high throughput. KFM, only maintains roughly 50 kbps for all flows since it suffers from large computational overhead like many homomorphic signature schemes.

C. Scalability with Multiple Adversaries

We compare SNK with EDART and HOMOMAC whose performance depends on the number of adversaries.

Comparison with EDART. DART delays every packet to wait for the checksum that verifies that packet, and EDART differs by forwarding packets before they are verified. As a result, some polluted packets travel multiple hops causing more damage, and EDART responds by forcing affected nodes to delay packets for verification. Unlike EDART, our scheme, SNK, verifies all packets without delaying them. So, the performance of SNK relative to the performance of EDART improves when attackers are present.

The performance of SNK and EDART with varying numbers of attackers are shown in Figures 3(a), 3(b), and 3(c). EDART outperforms SNK slightly in the benign scenario because packets are forwarded without being delayed for verification (verification is done later when a valid checksum is received), while SNK always verifies every packet. When attackers are present, SNK outperforms EDART which is most visible in the top 15% of flows where the difference ranges from 100 kbps to 300 kbps. These two schemes perform similarly for the rest of the 85% of flows with SNK gaining relative throughput to EDART as the number of attackers increases. Averaged over all flows, the increase in throughput of SNK relative to EDART are 4.8% and 6.2% for cases for cases of 5 and 10 attackers respectively. As the number of attackers increases, the EDART scheme will have a lower performance because it may either delay coded packets to verify them or allow polluted coded packets to be forwarded.

Comparison with HOMOMAC. We use HOMOMAC-x to denote the HOMOMAC scheme configured to defend against x adversaries. HOMOMAC utilizes redundant MACs and a special key distribution to ensure that an adversary cannot forge a coded packet. To remain resilient, the number of MACs per coded packet must increase as the number of colluding adversaries increases. We configure HOMOMAC to be resilient to varying numbers of colluding adversaries. In Figure 2, we compare the different HOMOMAC variants with SNK which is resilient to any number of adversaries. The severe degradation in performance is due to the increased communication cost of appending MACs to each packet, and the number of MACs increases quadratically with respect to the number of colluding adversaries.

D. SNK vs. Traditional Secure Routing

We showed that SNK outperforms other pollution defenses in a network coding system. However, for a secure scheme to be practical it must preserve network coding benefits. In other words, the secure network coding scheme should still have better performance than a secure traditional store-and-forward routing protocol. To demonstrate that SNK is a practical

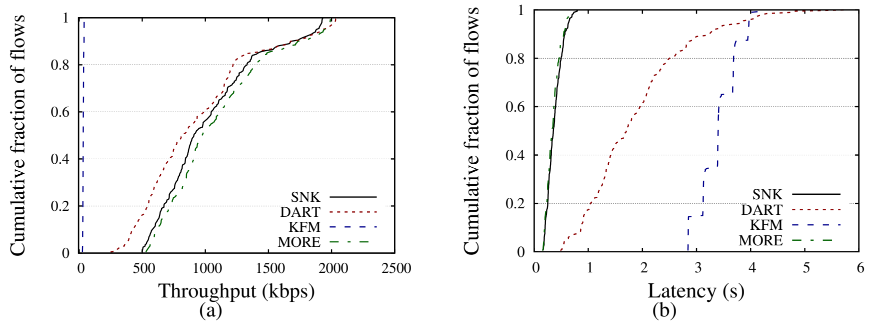


Fig. 1. Throughput and latency of SNK, KFM and DART.

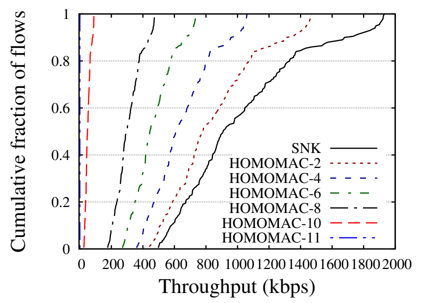


Fig. 2. Throughput of SNK which provides defense against any number of adversaries and HOMOMAC-x that is configured to provide defense against x adversaries.

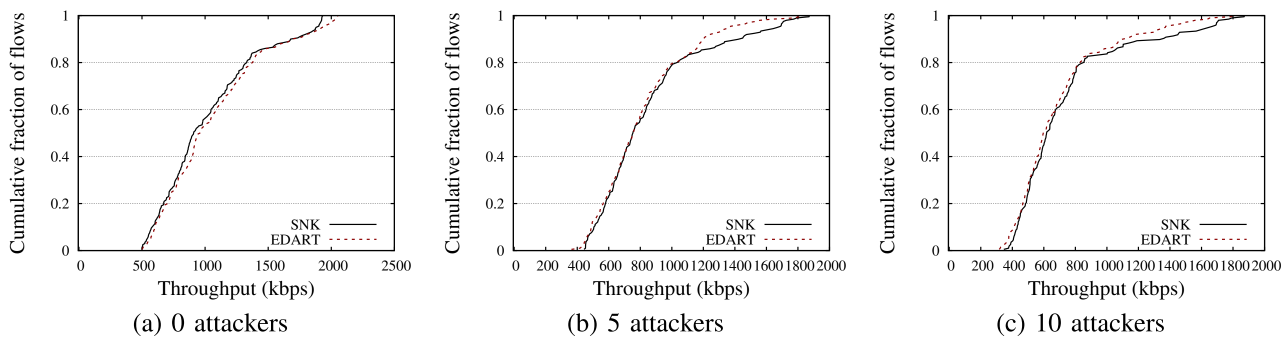


Fig. 3. Throughput of SNK and EDART for different numbers of attackers.

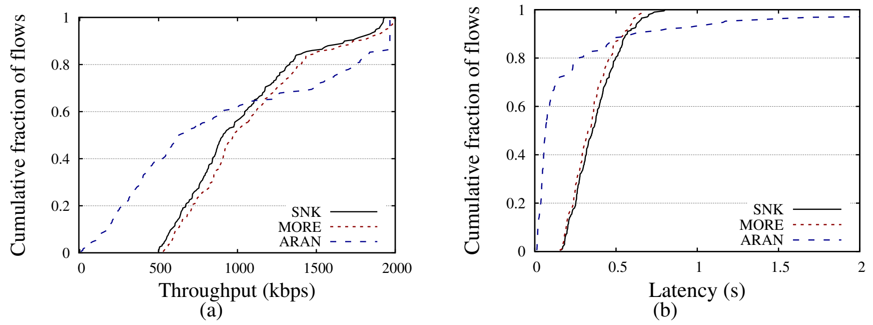


Fig. 4. Throughput and latency for SNK, MORE, and ARAN.

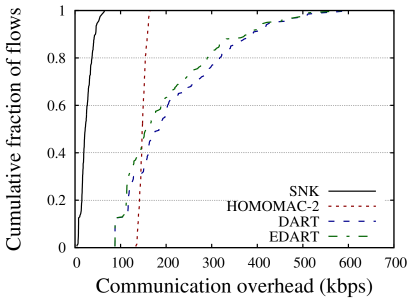


Fig. 5. Communication overhead of SNK, HOMOMAC-2, DART, and EDART.

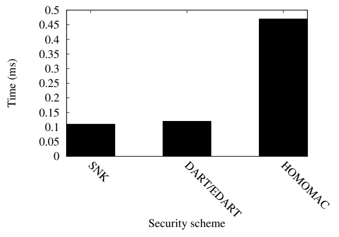
defense, we compare it with ARAN, a secure version of the well-known AODV wireless routing protocol which signs packets to ensure packets modified by routers are dropped.

Figure 4(a) shows that SNK retains most of the throughput of MORE. The throughput of SNK is roughly 50 kbps lower than MORE in all flows, and the degradation is consistent among all flows due to consistent overhead in distributing null keys. SNK outperforms ARAN in nearly the same fraction of flows that MORE outperforms ARAN, 65%, and this is due to the advantages of network coding. The 35% of flows where ARAN outperforms MORE and SNK are flows that have few hops, and few network coding advantages exist. The latency of network coding systems is generally higher than traditional routing because an entire generation is transferred before the first byte of data is decoded at the destination. SNK only imposes up to 10 ms of additional latency over MORE as seen in Figure 4(b). For 90% of flows, network coding imposes higher latency on the network. However, at the highest 10% of flows, the latency of ARAN is significantly higher due to the fact that shortest path routing suffers in flows with long paths in wireless mesh networks.

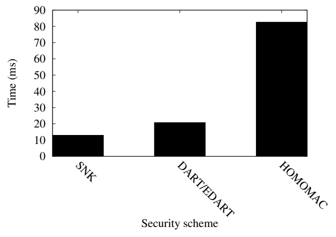
E. Overhead Results

We further evaluate the overhead of the protocols with better performance. We compare the overhead of SNK, with DART, EDART, and HOMOMAC. We did not include KFM in the overhead comparison because of its low performance (Figure 1) does not make it a good candidate for a pollution defense in wireless networks.

Communication overhead. Figure 5 presents the communication overhead for SNK, DART, EDART, and HOMOMAC. The median communication overhead of SNK is 25 kbps which is an insignificant amount given the median throughput rate of 900 kbps for SNK. The other pollution defenses have larger communication overheads. HOMOMAC has a consistent communication overhead between 130-170 kbps



(a) Forwarder computation



(b) Source computation

Fig. 6. Time for computational overhead of schemes SNK, DART/EDART, and HOMOMAC-2. Forwarder computation (a) is for verifying a coded packet. Source computation (b) is to generate null keys, checksums, and homomorphic MACs for SNK, DART/EDART, and HOMOMAC-2 respectively for a generation.

for all flows. DART and EDART have lower communication overhead than HOMOMAC in the lowest 40% of flows, but DART and EDART have communication overhead as high as 600 kbps in the highest flows. The large variations in overhead for DART and EDART are a result of the variations in the number of forwarders in each flow, and DART and EDART periodically disseminate checksums to all forwarders.

Computation overhead. We measure the computation overhead that takes place at the source to generate null keys, checksums, or MACs and at the forwarders to verify incoming coded packets. On average in our topology, a flow has 4.57 forwarders, so SNK's time is 4.57 multiplied by the time to create a null key packet. A null key packet requires the creation of one generation dependent null key, encrypting it with AES, and computing an HMAC of the packet with SHA-1. For DART/EDART at least one checksum packet is required per generation due to the checksum interval of 32 and $n = 32$, so we give DART/EDART an advantage by only benchmarking for one checksum per generation. A checksum packet requires the creation of 5 checksums for the 5 pipelined generations and 1 RSA signature. HOMOMAC's time is for creating the required homomorphic MACs for each generation. Benchmarking results are average times of 1000 runs of a computation on a 2.4 Ghz processor with cryptographic computations from the OpenSSL library [30].

Figure 6(a) and 6(b) present computational overhead at a forwarder and source respectively for each scheme. Note that the computational overhead at the destination is comparable to that at a forwarder. In both cases, SNK imposes the least computational overhead while DART/EDART imposes slightly more computational overhead. Due to the generation and verification of multiple MACs, HOMOMAC requires 4 times the overhead of both SNK and DART/EDART.

VII. Conclusion

We present the Split Null Keys (SNK) protocol, a pollution defense for wireless network coding that relies on null space properties. We show that our scheme is secure against a strong adversary that overhears all communication and compromises multiple forwarders. We evaluate our defense through simulations in a typical network coding system scenario. Our evaluation shows that SNK maintains the network coding gains of MORE and outperforms previous pollution defenses.

References

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

[2] X. Zhang and B. Li, "Optimized multipath network coding in lossy wireless networks," in *Proc. of ICDCS*, 2008.

[3] ——, "DICE: a game theoretic framework for wireless multipath network coding," in *Proc. of Mobihoc*, 2008.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *Proc. of SIGCOMM*, 2006.

[5] J. Le, J. C. S. Lui, and D. M. Chiu, "DCAR: Distributed coding-aware routing in wireless networks," in *Proc. of ICDCS*, 2008.

[6] S. Das, Y. Wu, R. Chandra, and Y. C. Hu, "Context-based routing: Technique, applications, and experience," in *Proc. of NSDI*, 2008.

[7] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Secure network coding for wireless mesh networks: Threats, challenges, and directions," *Computer Communications (Elsevier)*, vol. 32, November 2009.

[8] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Proc. of PKC*, 2009.

[9] M. Krohn, M. Freedman, and D. Maziéres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proc. of S&P*, 2004.

[10] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," *Proc. of CISS*, 2006.

[11] F. Zhao, T. Kalker, M. Médard, and K. Han, "Signatures of content distribution with network coding," in *Proc. of ISIT*, 2007.

[12] Q. Li, D. Chiu, and J. Lui, "On the practical and security issues of batch content distribution via network coding," in *Proc. of ICNP*, 2006.

[13] S. Agrawal and D. Boneh, "Homomorphic macs: Mac-based integrity for network coding," in *Proc. of ACNS*, 2009.

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

[15] D. Wang, D. Silva, and F. R. Kschischang, "Constricting the adversary: A broadcast transformation for network coding," in *Proc. of Allerton*, 2007.

[16] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proc. of WiSec*, 2009.

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

[18] "MIT roofnet." http://pdos.csail.mit.edu/roofnet/doku.php.

[19] K. Zhao, X. Chu, M. Wang, and Y. Jiang, "Speeding up homomorphic hashing using gpus," in *Proc. of ICC*, 2009.

[20] C. Gkantsidis and P. Rodriguez Rodriguez, "Cooperative security for network coding file distribution," 2006.

[21] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," *Proc. of PKC*, 2010.

[22] *Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures*. Springer, 2011.

[23] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. Shenz, "Padding for orthogonality: efficient subspace authentication for network coding," in *Proc. of INFOCOM*, 2011.

[24] A. Le and A. Markopoulou, "Locating byzantine attackers in intra-session network coding using spacemac," in *NetCod, IEEE International Symposium on*, 2010.

[25] M. Kim, M. Médard, and J. Barros, "A multi-hop multi-source algebraic watchdog," *Proc. of CoRR*, 2010.

[26] G. Liang, R. Agarwal, and N. Vaidya, "When watchdog meets coding," in *INFOCOM, Proceedings IEEE*, 2010.

[27] "Glomosim," http://pcl.cs.ucla.edu/projects/glomosim/.

[28] IEEE, *IEEE Std 802.11, 1999 Edition*, 1999, http://standards.ieee.org/catalog/olis/lanman.html.

[29] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding Royer, "A Secure Routing Protocol for Ad Hoc Networks," *Network Protocols, IEEE International Conference on*, 2002.

[30] "Openssl," http://www.openssl.org/.