On the Practicality of Cryptographic Defenses against Pollution Attacks in Wireless Network Coding

ANDREW NEWELL and JING DONG and CRISTINA NITA-ROTARU
Purdue University

Numerous practical systems based on network coding have been proposed in recent years demonstrating the wide range of benefits of network coding such as increased throughput, reliability, and energy efficiency. However, network coding systems are inherently vulnerable to a severe attack, known as *packet pollution*, which presents a key obstacle to the deployment of such systems. Several cryptographic schemes have been proposed to defend against pollution attacks.

We conduct a detailed analysis and an experimental evaluation in a realistic wireless network coding setting of a set of representative cryptographic defenses against pollution attacks. Our analysis identifies severe limitations of asymmetric based schemes which impose high communication overhead by placing constraints on the basic network coding parameters and high computation overhead by relying on numerous operations over large fields. Our analysis also shows that symmetric cryptographic schemes, while having better performance than asymmetric cryptographic based schemes, impose prohibitive overhead in the presence of multiple byzantine adversaries. We further evaluate these schemes by using a set of typical network coding system parameters on a realistic topology. Our experimental evaluation shows that all the schemes we compare induce a throughput degradation that negates the performance benefits of network coding in the presence of multiple colluding adversaries.

Categories and Subject Descriptors: C.2.1 [**Computer-communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms: Experimentation, Performance, Security

Additional Key Words and Phrases: Network Coding, Pollution Attacks, Wireless Mesh Networks

1. INTRODUCTION

Network coding has emerged as a new paradigm for designing network protocols. In network coding systems, nodes mix several buffered received packets to generate a coded packet, then forward the coded packet. The benefits of network coding include increased throughput [Ahlswede et al. 2000; Li et al. 2003; Jin et al. 2006], increased reliability [Widmer and Le Boudec 2005; Lun et al. 2005], and reduced energy cost [Wu et al. 2005]. Numerous practical systems [Chachulski et al. 2007; Zhang and Li 2008b; 2008a; Katti et al. 2006; Le et al. 2008; Das et al. 2008] have been proposed for wireless networks.

Due to a fundamental change in the trust model, network coding creates opportunities for new attacks. The most severe is the *pollution attack* in which an attacker injects polluted (or invalid) packets. Pollution attacks are devastating to a network coding sys-

This research is supported by the NSF NETS Program.
Authors' addresses: Department of Computer Science, Purdue University, 305 N. University St., West Lafayette, IN 47907 USA; email: {newella,dongj,crisn}@cs.purdue.edu

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

tem because they propagate epidemically as honest nodes use polluted packets to code new packets and thus unknowingly behave as polluters themselves.

While the receivers can detect polluted packets by using end-to-end integrity mechanisms or recover the packets when redundant coding is used, this approach is not practical for wireless networks that have limited resources. Instead, polluted packets must be identified and dropped at intermediate nodes. Such an approach requires cryptographic mechanisms that have homomorphic properties, allowing any intermediate node to verify that the coded packets it combines ultimately originated at the source.

Several cryptographic defenses against pollution attacks have been proposed [Boneh et al. 2009; Gkantsidis and Rodriguez Rodriguez 2006; Charles et al. 2006; Krohn et al. 2004; Yu et al. 2008; Zhao et al. 2007; Li et al. 2006; Agrawal and Boneh 2009; Zhang et al. 2011]. The security of such schemes relies on the security of the basic cryptographic primitive the scheme uses, which often relies on the intractability of a mathematical problem such as the Discrete Log Problem (DLP). However, both the cryptographic and network coding operations are performed in the same field, which may trigger significant changes to the network coding parameters in order to meet security requirements. Previous work did not study the impact of security parameters on the network coding parameters that control system performance.

Other work proposed defenses which utilize mechanisms other than just cryptography such as time-asymmetry [Dong et al. 2009; Li et al. 2010], information theory [Ho et al. 2004; Jaggi et al. 2007; Wang et al. 2007; Silva et al. 2008; Yeung and Cai 2006; Cai and Yeung 2006], and monitoring [Kim et al. 2010; Kehdi and Li 2009] to defend against pollution attacks. These works potentially escape the problems of pure cryptographic approaches, but due to their reliance on other mechanisms, these defenses place additional constraints on the network, modify the underlying routing protocol, and provide a lesser defense as some polluted packets will be forwarded and adversary affect the network performance. We discuss these other defenses in more detail in Section 6.

Cryptographic defense schemes have been proven to prevent pollution attacks. A defense scheme must also be practical to ensure adoption. To be practical, the defense scheme must scale with the number of (possibly colluding) attackers and retain as much performance of the insecure system as possible. The schemes proposed to defend against pollution attacks [Boneh et al. 2009; Gkantsidis and Rodriguez Rodriguez 2006; Charles et al. 2006; Krohn et al. 2004; Yu et al. 2008; Zhao et al. 2007; Li et al. 2006; Agrawal and Boneh 2009; Zhang et al. 2011] have not been evaluated to see if they are practical in wireless network coding systems.

In this paper, we perform the first systematic analytical and experimental evaluations of several representative cryptographic defenses against pollution attacks in wireless networks. We compare the KFM [Krohn et al. 2004], YWRG [Yu et al. 2008], NCS [Boneh et al. 2009], ZKMH [Zhao et al. 2007] asymmetric cryptographic schemes and the HOMOMAC [Agrawal and Boneh 2009] symmetric cryptographic scheme. Our simulations are conducted using the Glomosim [GloMoSim 2000] simulator, the well-known MORE [Chachulski et al. 2007] wireless network coding system and the link quality dataset from the Roofnet [MIT 2006] testbed. Our main contributions include:

— We define a unifying metric framework that captures the different performance and security impact of all schemes. We classify the cryptographic schemes based on their underlying cryptographic primitives of signature-based, hash-based, and MAC-based (Message Authentication Code). Our framework provides network coding and security parameters settings that allows us to compare the schemes while ensuring they have comparable security.

— We propose a definition of practicality for a secure network coding scheme. For a majority of cases where network coding outperforms shortest path routing, secure network coding should also outperform secure shortest path routing. We consider as a baseline ARAN [Sanzgiri et al. 2002], a secure version of the well-known AODV [Perkins et al. 2003] routing protocol for wireless networks.

— We identify strong dependencies between network coding parameters, which dictate network coding gains, and security parameters which ensure security of basic cryptographic primitives. Such dependencies cause defense schemes relying on asymmetric cryptographic primitives to impose symbol sizes of 20-32 bytes in comparison to symbol sizes of 1-2 bytes that were shown to work well for wireless network coding systems. As a result, such schemes make the coding overhead prohibitively large for wireless settings.

— We show through analysis and benchmarking that asymmetric cryptographic defense schemes require costly computation operations per plain packet symbol, per coded packet symbol, and per coded packet, resulting in a high computational overhead.

— We show through simulations that each scheme relying on an asymmetric cryptographic primitive induces a throughput degradation that negates the performance benefits of network coding. Even more, such a degradation of performance is incurred regardless of the presence of attackers in the network. We also show that the HOMO-MAC symmetric cryptographic scheme is the only one that is practical for a very small number of adversaries (2 or less).

— We find that schemes based on a symmetric cryptographic primitive for data integrity, a MAC, do not scale well when tolerating multiple byzantine adversaries. To be resilient against multiple byzantine adversaries a MAC-based scheme must append many MACs per packet and distribute unique sets of keys to each forwarder while the redundancy of MACs and keys ensures that no collection of byzantine nodes can create enough MACs to allow a polluted packet to pass a legitimate node's verification test, this same redundancy imposes prohibitive overhead when multiple attackers exist in the system.

Roadmap. We provide an overview of wireless network coding in Section 2. In Section 3 we classify existing cryptographic defenses against pollution attacks and describe representative schemes for each class. We present our analytical metrics and a comparison of a set of representative schemes in Section 4. We define the performance metrics and evaluate the schemes in Section 5. We present related work in Section 6 and conclude the paper in Section 7.

2. WIRELESS NETWORK CODING

We overview network coding, outlining the specific constraints of network coding in wireless networks.

2.1. System model

We consider a general *intra-flow* network coding system that mixes packets within the same flow. The network consists of a source node, a subset R of receiver nodes, and a subset F of *forwarder nodes*. The source delivers to receivers a sequence of plain packets which are divided into sub-sequences called *generations*. A generation consists of n packets. Each packet consists of m symbols. Symbols are elements of $\mathbb{F}_q$, and packets $\mathbf{b}_i$ are m-dimensional vectors of symbols:

$$\mathbf{b}_i = (b_{i,1}, b_{i,2}, ..., b_{i,m}), b_{i,j} \in \mathbb{F}_q.$$

A generation is represented by a matrix B of symbols $b_{i,j}$ where there are nm total symbols.

The source forms random linear combinations of the plain packets $\mathbf{x} = \sum_{i=1}^{n} v_i \mathbf{b}_i$ where $\mathbf{v}$ is an n-dimensional vector of random symbols such that $\mathbf{v} = (v_1, v_2, ..., v_n)$. The vector $\mathbf{v}$ is the *coding vector* which consists of the coding coefficients, and the vector $\mathbf{x}$ is the *coded data*. A *coded packet* $\langle \mathbf{v}, \mathbf{x} \rangle$ consists of both the coding vector and the coded data. A forwarder node buffers received coded packets and creates new coded packets by computing random linear combinations of the coded packets within its buffer (i.e., this can be a number between 2 and n). Once a receiver obtains n linearly independent coded packets it can obtain the plain packets by solving a system of linear equations.

2.2. Constraints for network coding

Coded packet size. One constraint when using network coding to design practical wireless networks is that the coded packet size is limited by the transmission frame size. This design decision allows a coded packet to be immediately available for use by a forwarder or receiver node.

Network coding overhead. An important characteristic of network coding systems is the communication overhead introduced in the system by the coding vector. Given a coded packet $\langle \mathbf{v}, \mathbf{x} \rangle$, this overhead is measured by the ratio $\rho = \frac{n}{m}$ where n is the number of symbols in the coding vector $\mathbf{v}$ (i.e., the size of the generation) and m is the number of symbols in the coded data $\mathbf{x}$. The smaller ρ is, the less communication overhead incurred by network coding. When designing a network coding system for wireless networks, the selection of parameters for network coding is chosen such that $n \ll m$ to ensure a small network coding overhead. However, the upper bound on the coded packet size has consequences on the lower bound on ρ. Specifically, reducing n is lower bounded because it reduces the potential gains of network coding (e.g., if $n = 1$ no network coding is present), and increasing m is upper bounded due to the maximum transmission frame size. A larger m may result in a packet larger than the transmission size and thus have a negative impact on system performance.

Transmission frame and symbol size. Increasing the coded packet size beyond the standard transmission frame would reduce the network coding overhead. However, the typical wireless transmission frame size offers a good balance between mitigating the overhead of sending header information and reducing the probability of a single bit-error in a packet [Lettieri and Srivastava 1998]. Increasing the coded packet size increases the probability of a bit-error which potentially reduces the overall throughput for a wireless link.

The symbol size has a direct impact on the number of coded data symbols m, and thus, on system performance. In order to meet the constraint that $(n + m)\lambda_q \leq \Omega$ (where λ_q is the symbol size and Ω is the maximum transmission frame size) and ensure network coding gains, several of the proposed practical systems use symbol sizes of 1-2 bytes.

3. CRYPTOGRAPHIC DEFENSES AGAINST POLLUTION ATTACKS

Several cryptographic schemes [Krohn et al. 2004; Yu et al. 2008; Zhao et al. 2007; Li et al. 2006; Agrawal and Boneh 2009; Charles et al. 2006; Boneh et al. 2009; Zhang et al. 2011] were proposed in recent years to defend against pollution attacks in network coding. We first present an overview of the general procedure these schemes follow, then present a detailed description for each scheme.

3.1. Overview

Table I presents a list of representative schemes guarding against pollution attacks. Each scheme provides protection against such attacks by supplying forwarders with a

Table I: Taxonomy of schemes

Scheme	Category	Security	Steps
KFM [Krohn et al. 2004]	Hash	DLP over a multiplicative group	Sign/Verify
YWRG [Yu et al. 2008]	Signature	DLP over a multiplicative group	Sign/Verify/Combine
ZKMH [Zhao et al. 2007]	Signature	DLP over a multiplicative group	Sign/Verify
LCL [Li et al. 2006]	Signature	DLP over a multiplicative group	Sign/Verify/Combine
NCS_1 [Boneh et al. 2009]	Signature	DLP using ECC	Sign/Verify/Combine
NCS_2 [Boneh et al. 2009]	Signature	DLP over a multiplicative group	Sign/Verify/Combine
CJL [Charles et al. 2006]	Signature	DLP using ECC	Sign/Verify/Combine
HOMOMAC [Agrawal and Boneh 2009]	MAC	PRF	Sign/Verify/Combine
HSM [Zhang et al. 2011]	MAC	DLP over a multiplicative group	Sign/Verify/Combine

verification mechanism that allows them to detect invalid (polluted) packets and cease propagating these invalid packets. An invalid packet is any packet $\mathbf{c} = \langle \mathbf{v}, \mathbf{x} \rangle$ that does not satisfy the following:

$$\mathbf{x} = \mathbf{v}B$$

B is a matrix of plain packet symbols. The verification scheme uses hashes, message authentication codes (MACs), or digital signatures as cryptographic primitves. The security of each scheme relies on the DLP over a multiplicative group, DLP using Elliptical Curve Cryptography (ECC), or Pseudo-Random Functions (PRF).

In each scheme the validation of packets ultimately relies on the source generating the verification information (hashes, MACs, or digital signatures). In some schemes the source distributes this information before sending the generation, while in other schemes the verification information is carried by each individual packet. In the latter case, forwarder nodes also are involved in creating the verification information by combining the verification information carried by each of the coded packets used to code a new packet. We refer to security related data sent either separate or attached to a coded packet as the *security payload*. Each scheme follows the same general procedure:

• **Initialize**: the source distributes generation independent public security parameters to forwarder nodes. This can be performed off-line.

• **Sign**: the source calculates a security payload either for each plain packet and appends it to each coded packet, or for the entire generation and distributes it to the forwarder nodes.

• **Verify**: each forwarder node verifies the received coded packets based on the generation independent security parameters and the security payload received either at the beginning of the generation or carried on each coded packet, depending on the scheme.

• **Combine**: the source or forwarder nodes combine the security payloads to create a security payload for a newly formed coded packet. Only schemes where each coded packet carries a security payload have a combine step.

Next, we describe each scheme considering only the sign, verify, and combine steps as the initialize step can be performed off-line and is generation independent. Table II presents the notation we use in the description of the schemes.

Table II: Parameters

Name	Description
λ_i	size of parameter i in bytes
n	number of packets in a generation
m	number of symbols per packet
q	maximum value of a symbol (λ_q size of a symbol)
p	security parameter (controls modulus size)
$\mathbf{v}$	coding vector (n symbols)
$\mathbf{x}$	coded data (m symbols)
Ω	maximum size of a frame in bytes
s	security payload appended to each coded packet
S	security payload distributed initially to forwarders
B	matrix n by m that represents the plain packets
$\mathbf{b}_i$	row i of B, i.e., the i^{th} block
$b_{i,j}$	a single symbol within row i and column j of B
$\mathbf{c}$	coded packet $\langle \mathbf{v}, \mathbf{x} \rangle$
$\mathbf{c}_i$	i^{th} coded packet within a node's buffer
$\mathbf{c}_i^*$	plain packets in the form of coded packets $\mathbf{c}_i^* = \langle \mathbf{v}, \mathbf{x} \rangle$ where $\mathbf{v} = \langle 0, ..., 0, 1, 0, ..., 0 \rangle$ with 1 in the i^{th} position and $\mathbf{x} = \mathbf{b}_i$
l	number of keys in a key-pool
ω	number of keys in a key-chain

3.2. Hash-based schemes

We first present schemes relying on homomorphic hashes [Gkantsidis and Rodriguez Rodriguez 2006; Krohn et al. 2004]. The main characteristic of a homomorphic hash is that the hash of a linear combination of the coded packets is equivalent to the linear combination of the hashes of coded packets. We classify the hash-based schemes as asymmetric cryptographic schemes because the homomorphic hashes are generated and then signed by conventional non-homomorphic signatures.

KFM[Krohn et al. 2004] is a representative example of a homomorphic hash construction. Algorithm 1 presents pseudo-code describing the main steps of the scheme. KFM consists only of a sign step [1] and a verify step. The source generates a hash for each plain packet, then distributes the n hashes to all forwarders in an authenticated manner. In order to verify a received coded packet, a forwarder compares the hash computed on the received coded packet with the hash constructed by linearly combining the hashes that were distributed by the source in the sign step. Due to the homomorphic properties of the hash function, if the coded packet is correct, the two compared hash values will be equal.

The benefits of this approach are that no security payload is carried by any coded packet and no extra computations are needed when forwarders generate new coded packets. The drawbacks are that the security payload sent at the beginning of each generation is significant, and that the verification process requires a large number of exponentiations. The security payload has a total size of $n\lambda_p$ bytes and the computation overhead has nm modular exponentiations performed by the source to generate the hashes. The verifying step requires $n + m$ modular exponentiations. The scheme requires that the source distributes hashes in an authenticated manner, otherwise an attacker can inject them and break the data authentication property.

[1]We use the same name for the step for consistency, even if in this case the operation is hashing.

Algorithm 1: KFM

```
*Parameters*
  $p$ and $q$ are primes of size $\lambda_p$ and $\lambda_q$ respectively
  $g_1, ..., g_m$ are generators of the group $\mathbb{F}_p$ of order $q$
*Signing plain packets* $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$
  1: Calculate hash $h(\mathbf{b}_i) = \prod_{j=1}^{m} g_j^{b_{i,j}} \bmod p$ for $\mathbf{b}_1, ..., \mathbf{b}_n$
*Verifying* $\boldsymbol{c} = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$
  1: Verify that $\prod_{i=1}^{m} g_i^{x_i} \bmod p = \prod_{i=1}^{n} h(\mathbf{b}_i)^{v_i} \bmod p$
```

3.3. Signature-based schemes

The majority of cryptographic defenses against pollution attacks rely on homomorphic signatures [Yu et al. 2008; Zhao et al. 2007; Li et al. 2006; Charles et al. 2006; Boneh et al. 2009; Zhang et al. 2011]. The main characteristic of a homomorphic signature is that the signature of a linear combination of coded packets is equivalent to the linear combination of signatures. We describe three representative schemes: YWRG [Yu et al. 2008], a scheme based on DLP over a multiplicative group in which coded packets are signed individually and carry with them the corresponding signature as a security payload; NCS$_1$[Boneh et al. 2009], a scheme based on DLP using ECC in which, as in YWRG, coded packets are signed individually and carry the signature; ZKMH[Zhao et al. 2007], a scheme based on DLP over a multiplicative group in which the source signs the entire generation instead of individual coded packets.

YWRG[Yu et al. 2008] is shown in Algorithm 2. First, the source signs each plain packet, then for every coded packet it combines the corresponding signatures to generate the signature of the coded packet and attaches this signature to the packet. When receiving a coded packet, a forwarder node verifies the signature included on the packet by using the security parameters distributed during the initialize step. Before sending out a new coded packet, the forwarder node computes the corresponding signature by combining the signatures of the coded packets it used to create the new coded packet and appends the signature to the new coded packet.

Algorithm 2: YWRG

```
*Parameters*
  $p$ and $q$ are primes of size $\lambda_p$ and $\lambda_q$ respectively
  $g_1, ..., g_{m+n}$ are generators of $\mathbb{F}_p$ of order $q$
  $r$ (modulus), $e$ (public key), and $d$ (private key) of RSA
*Signing plain packets formed as coded packets* $\boldsymbol{c}_1^*, ..., \boldsymbol{c}_n^*$
  1: Calculate signature $\sigma(\mathbf{c}) = (\prod_{j=1}^{m+n} g_j^{c_j} \bmod p)^d \bmod r$ for $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$
*Verifying* $\boldsymbol{c}$ *with signature* $\sigma(\boldsymbol{c})$
  1: Verify that $\sigma(\mathbf{c})^e \bmod r = (\prod_{j=1}^{m+n} g_j^{c_j} \bmod p) \bmod r$
*Combining* $\sigma(\boldsymbol{c}_i)$ *for* $i = 1, ..., n$ *to produce* $\sigma(\boldsymbol{c})$ *where* $\boldsymbol{c} = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$
  1: Calculate $\sigma(\mathbf{c}) = \prod_{i=1}^{n} \sigma(\mathbf{c}_i)^{v_i} \bmod r$
```

YWRG has the benefit of not requiring any security payload to be distributed to forwarder nodes each generation. Instead, a security payload is appended to each coded packet. The drawbacks are the additional computational overhead of the combining step and the communication overhead of a security payload appended to each coded packet. The combining step requires an exponentiation for each coded packet being combined, and $n(n+m+1)$ exponentiations to generate signatures for an entire generation. The verifying step requires $n+m+1$ exponentiations.

NCS$_1$[Boneh et al. 2009] (presented in Algorithm 3) is a scheme similar to YWRG in that each coded packet carries its corresponding signature. However, the signature construction relies on ECC. As in YWRG, the source signs the plain packets and when coding a packet, the source combines the signatures of the coded packets to generate the signature of the newly formed coded packet. When verifying a coded packet, a forwarder computes the product of bilinear maps based upon the coded packet contents and checks the equivalence of this value with a bilinear map of the signature. When combining coded packets, a forwarder computes a new signature by performing ECC group operations on the coded packets' signatures.

```
Algorithm 3: NCS$_1$
Parameters
    $q$ is a prime of size $\lambda_q$
    $\alpha$ is an element of $\mathbb{F}_q$
    $P_1, ..., P_m$, and $Q$ are random elliptical curve points
    $U$ is an elliptical curve point such that $U = \alpha Q$
    $H(*, *)$ hash that outputs an elliptical curve point
    $e(*, *)$ is a bilinear pairing function
    id is a unique identifier for a generation
Signing plain packets formed as coded packets $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$
  1: Calculate the signature $\sigma(\mathbf{c}) = \alpha * (\sum_{i=1}^{n} v_i H(\text{id}, i) + \sum_{i=1}^{m} x_i P_i)$ for each $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$
Verifying $\boldsymbol{c}$ with signature $\sigma(\boldsymbol{c})$
  1: Verify that $e(\sigma(\mathbf{c}), Q) = e(\sum_{i=1}^{n} v_i H(\text{id}, i) + \sum_{i=1}^{m} x_i P_i, U)$
Combining coded packet signatures $\sigma(\boldsymbol{c}_i)$ for $i = 1, ..., n$ to produce $\sigma(\boldsymbol{c})$ where $\boldsymbol{c} = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$
  1: Calculate $\sigma(\mathbf{c})$ where $\sigma(\mathbf{c}) = \sum_{i=1}^{n} v_i \sigma(\mathbf{c}_i)$
```

The main benefits of NCS$_1$ are that the sizes of the signature and a symbol (λ_q) are smaller given the use of ECC. The drawbacks are the increased computational times for ECC operations and bilinear mappings over modular exponentiations. Even though NCS$_1$ has a comparable number of costly operations as other schemes, each ECC or bilinear mapping operation requires extra time.

ZKMH[Zhao et al. 2007] is a scheme that signs the entire generation instead of individual coded packets. Algorithm 4 presents pseudo-code describing the main steps of the scheme, signing and verifying. The source signs the entire generation and distributes this security payload to the forwarder nodes. The signature is generated over the linear subspace formed by treating $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$ as basis vectors. In order to verify a coded packet, a forwarder node checks whether the coded packet belongs to the linear subspace, i.e., a coded packet is valid if and only if the coded packet is a linear combination of $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$.

ZKMH shares the main benefits of KFM in that there is no security payload appended to coded packets and no combining of security payloads. In addition, because

Algorithm 4: ZKMH

```
Parameters
  $p$ and $q$ are primes of size $\lambda_p$ and $\lambda_q$ respectively
  $g$ is a generator of $\mathbb{F}_p$
  $\alpha_1, ..., \alpha_{n+m}$ are elements of $\mathbb{F}_q$ of order $q$
  $t_1, ..., t_{n+m}$ are elements of $\mathbb{F}_p$ where $t_i = g^{\alpha_i} \bmod p$
Signing plain packets formed as coded packets $\boldsymbol{c}_1^*, ..., \boldsymbol{c}_n^*$
  1: Generate a new $\alpha_i$ and calculate $t_i = g^{\alpha_i} \bmod p$ for some $i = 1, ..., n+m$
  2: Generate $\mathbf{u}$ where $\mathbf{c}_i^* \cdot \mathbf{u} = 0$ for $i = 1, ..., n$
  3: Calculate the signature $\sigma$ where $\sigma = (u_1/\alpha_1, u_2/\alpha_2, ..., u_{n+m}/\alpha_{n+m})$
Verifying $\boldsymbol{c}$
  1: Verify that $\prod_{i=1}^{n+m} t_i^{\sigma_i c_i} \bmod p = 1$
```

the ZKHM scheme is based on a signature construction, the security payload distributed to forwarders does not have to be authenticated as the source is the only node capable of generating valid signatures. The drawback of computational overhead still exists as $n + m$ exponentiations are required to verify a coded packet.

3.4. MAC-based schemes

A third type of cryptographic defense against pollution attacks relies on homomorphic MACs [Agrawal and Boneh 2009; Zhang et al. 2011]. We describe HOMOMAC [Agrawal and Boneh 2009] as a representative scheme for MACs. The main characteristic of a homomorphic MAC is that the MAC of a linear combination of coded packets is equivalent to the linear combination of MACs of each coded packet. MAC-based schemes rely on a symmetric cryptographic primitive which requires the verifier to have the same key that the source used to create the MAC. Several approaches are possible: (1) have the source share symmetric keys with each forwarder such that if one node is compromised the security of the network is not affected, (2) have the source and all forwarders sharing only one key in which case the compromise of one node compromises the security of the entire network, or (3) have the source share a set of keys from a key-pool, and use a set of MACs one for each key. In the third case the scheme trades off security for performance. It can tolerate a certain number of compromised nodes without requiring secret keys between the source and each forwarder. When the number of compromised nodes is higher than what the scheme can tolerate, the compromised nodes can collude to forge a coded packet that will pass as valid. HOMOMAC, the representative scheme that we choose, utilizes the third method of distributing keys to ensure a trade-off between resilience to byzantine nodes and number of keys maintained by each node.

HOMOMAC[Agrawal and Boneh 2009] uses homomorphic MACs to defend against pollution attacks. Algorithm 5 contains pseudo-code describing the main steps of the scheme. The scheme is similar with the YWRG and NCS$_1$ schemes in the sense that each coded packet carries a security payload, with the difference that this security payload is not a digital signature but a set of l MACs. First, the source generates l MACs for each plain packet, and when generating a coded packet combines the MACs of the plain packets to generate the MACs for the coded packet. When receiving a coded packet, a forwarder checks that ω MACs are all correct by using the ω MACs from the forwarder's key-chain. When creating new coded packets, a forwarder node computes a linear combination of each of the l MACs to form l new MACs for the new coded packet.

Algorithm 5: HOMOMAC

```
*Parameters*
    $q$ is a power of a prime where $q$ is of size $\lambda_q$
    $k_1, ..., k_l$ are keys generated at the source
    id is a unique identifier for a generation
    $PRG(*)$ is a pseudo-random generator that outputs a vector $\mathbb{F}_q^{n+m}$
    $PRF(*, *, *)$ is a pseudo-random function that outputs an element of $\mathbb{F}_q$
*Signing plain packets formed as coded packets $\boldsymbol{c}_1^*, ..., \boldsymbol{c}_n^*$*
  1: Calculate MACs for $i = 1, ..., l$ $MAC_i(\mathbf{c}_j) = (PRG(k_i) \cdot \mathbf{c}_j) + PRF(k_i, \text{id}, j) \bmod q$
     for each $\mathbf{c}_1^*, ..., \mathbf{c}_n^*$
*Verifying $\boldsymbol{c} = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$ with $MAC_1(\boldsymbol{c}), ..., MAC_l(\boldsymbol{c})$ using a key-chain of keys $k_{z_1}, ..., k_{z_\omega}$*
  1: Verify that $PRG(k_{z_i}) \cdot \mathbf{c} + \sum_{j=1}^{n} v_j PRF(k_{z_i}, \text{id}, j) \bmod q = MAC_{z_i}(\mathbf{c})$ for $i = 1, ..., \omega$
*Combining $MAC_i(\boldsymbol{c}_j)$ for $i = 1, ..., l$ and $j = 1, ..., n$ to produce $MAC_i(\boldsymbol{c})$ for
$i = 1, ..., l$ where $\boldsymbol{c} = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$*
  1: Calculate MACs for $i = 1, ..., l$ $MAC_i(\mathbf{c}) = \sum_{j=1}^{n} v_j MAC_i(\mathbf{c}_j) \bmod q$ for $i = 1, ..., l$
```

The approach has the benefits that it has no costly operations during the computational steps and no constraint on the symbol size. The majority of operations are modular multiplications and the security relies on pseudo-random functions (PRF) which places no restrictions on the symbol size. The major drawback is the weaker defense it provides in the presence of multiple byzantine adversaries due to the special key distribution. This is the only scheme where the maximum number of byzantine adversaries within the network must be known and bounded. The scheme provides a defense when the adversary has a limited number of key-chains such that only a limited number of MACs can be created for an invalid coded packet. This constraint is broken when enough colluding adversarial nodes combine several key-chains. The value of the maximum number of adversaries is an input to the initial key distribution. This value must also be bounded because as the value increases so does the number of MACs necessary to compute and append to coded packets. To tolerate a larger number of adversaries the communication cost of the size of the security payload appended to each coded packet is quadratic with the number of adversaries. See Section 4.5 for details.

4. ANALYTICAL COMPARISON

In this section we compare the schemes that we presented in the previous section. We define metrics that reflect the overhead imposed by each scheme and summarize these metrics using realistic system parameters for a typical wireless network coding system. Table II presents the variables used by our metric definitions.

4.1. Analysis metrics

Communication overhead. We quantify the communication overhead using two metrics: *relative security overhead* and *relative coding overhead*. Relative security overhead reflects the overhead imposed by distributing the security payload. Relative coding overhead reflects the overhead to send coding coefficients. Different schemes may result in higher network coding overhead due to the relationship that exists between the symbol size used by the network coding and the constraints imposed by the cryptographic scheme on the symbol size in order to preserve its security properties.

We define *relative security overhead* as the fraction between the total security payload (data sent initially by the source and security payload for all the packets in a generation), and the total data sent for a generation (security payload and all the coded packets in a generation). We define relative security overhead for a generation and not for a packet to capture all the additional security overhead imposed by a security scheme.

$$Rel\_Security\_Overhead = \frac{n\lambda_s + \lambda_S}{n(\lambda_{\mathbf{v}} + \lambda_{\mathbf{x}} + \lambda_s) + \lambda_S} \tag{1}$$

We define the *relative coding overhead* as the fraction between the coding vector for a packet and the total data for that secure coded packet (coding vector, coded data, and security payload for that packet). We define relative coding overhead for a packet to capture the cost per packet imposed on coding by a security scheme.

$$Rel\_Coding\_Overhead = \frac{\lambda_{\mathbf{v}}}{\lambda_{\mathbf{v}} + \lambda_{\mathbf{x}} + \lambda_s} \tag{2}$$

Computation overhead. We breakdown the computational overhead imposed by each security scheme during the different steps: signing which is performed only at the source, verifying which is performed only at a forwarder node, and combining which is performed at either the source or a forwarder node. We use two sets of metrics: one that accounts for the number of expensive operations involved in each step, and a more precise one, that accounts for the actual time required to perform each step.

The first set of metrics consists of *signing operations*, *verifying operations*, and *combining operations* which count the following expensive operations: modular exponentiations, ECC scalar operations, and bilinear mappings. Benchmarking tests show that these three operations require relatively similar time; therefore, we group them together to provide a comparison among different schemes. *Signing operations* counts the number of cryptographic operations performed by the source to sign a generation. *Verifying operations* counts the number of operations performed by a forwarder node to verify a coded packet. The number of expensive operations for the signing and verifying steps remains constant for each scheme. The number of operations varies for the combining step based upon the number of coded packets used to create a new coded packet. As this number may vary from 1 to n, we assume that on average a forwarder node combines $\frac{n}{2}$ packets to generate a new coded packet and define *combining operations* as the number of operations performed by a forwarder node to generate the security payload needed for a new coded packet based on $\frac{n}{2}$ buffered packets.[2]

The second set of metrics consists of *signing time*, *verifying time*, and *combining time* which are calculated from benchmarking tests of these operations using a cryptographic library and averaged over 100 runs.

4.2. Parameter settings

We assign values to the parameters Ω, λ_q, λ_p, n, m, l, and ω from Table II matching realistic wireless settings. The values assigned ensure that network coding gains are possible, security holds for each scheme, and the level of security is comparable between different schemes. We give priority to security for parameters that have implications on both security and network coding gains. Our assigned parameter values are shown in Table III, and these choices are justified below.

The values for the security parameters λ_q, λ_p, l, and ω shown in Table III are assigned such that security holds for each scheme and the level of security is similar for

[2]Adjusting the number of coded packets being combined does not affect our results significantly. The overhead of combining is correlated linearly with the number of coded packets in a buffer.

Table III: Parameter values

Scheme	λ_q	λ_p	l	ω	n	m	Ω
KFM	32	128	N/A	N/A	16	30	1500
YWRG	32	128	N/A	N/A	16	26	1500
ZKMH	32	128	N/A	N/A	16	30	1500
NCS$_1$	20	N/A	N/A	N/A	16	57	1500
HOMOMAC	1	N/A	121	11	16	1363	1500

Table IV: Communication overhead analysis

Scheme	Relative security overhead	Relative coding overhead
KFM	$\frac{\lambda_p}{(n+m)\lambda_q+\lambda_p}$ **	$\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$
YWRG	$\frac{\lambda_p}{(n+m)\lambda_q+\lambda_p}$ *	$\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$
ZKMH	$\frac{n+m+1}{(n+1)(n+m)+1}$ **	$\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$
NCS$_1$	$\frac{2}{n+m+2}$ *	$\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$
HOMOMAC	$\frac{l}{n+m+l}$ *	$\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$

* appended to coded packets (s)
** distributed to forwarders (S)

each scheme. Security refers to the inability of an adversary to bypass the verification scheme by forging hashes, digital-signatures, or MACs. KFM, YWRG, and ZKMH each rely on the DLP over the group $\mathbb{F}_p$ for security, so the field size λ_p is set to 1024 bits, and the symbol size λ_q is set to 256 bits. NCS$_1$ uses ECC, so its group size (the group size is also the symbol size for this scheme) λ_q is set to 160 bits which is comparable to the DLP over the group $\mathbb{F}_p$ that has a modulus of 1024 bits. HOMOMAC relies on pseudo-random functions for its security, and in [Agrawal and Boneh 2009] the authors show that a key-pool size l of 121, a key-chain size ω of 11, and a symbol size λ_q of 1 gives a $(\frac{1}{2})^{40}$ probability of successfully forging a coded packet in the presence of two byzantine adversaries. HOMOMAC has a lower level of security than the other schemes due to the fact that this is the only scheme whose security depends on the number of colluding adversaries.

The values for the network coding parameters Ω, n, and m are assigned to ensure that network coding gains are possible, these choices are shown in Table III. The number of plain packets per generation, n, is set to 16. Smaller values of n reduce the relative coding overhead, and $n = 16$ was the smallest value for n shown to still provide network coding gains in [Chachulski et al. 2007]. We chose to use for Ω the standard maximum transmission frame size of 1500 bytes. Transmission frames of this size were used to measure packet loss rates for wireless links of Roofnet [MIT 2006], which we use for our topology later in the evaluation section.[3] With all of these parameters set, m is maximized for each scheme such that the network coding coefficients, any security payload appended to a coded packet, and the coded data fit within the maximum transmission frame size.

[3]Higher transmission size reduces relative coding overhead, but results in increased packet loss [Lettieri and Srivastava 1998].

Table V: Computation overhead analysis

Scheme	Sign	Verify	Combine
KFM	nm	$n+m$	0
YWRG	$n(n+m+1)$	$n+m+1$	$\frac{n}{2}$
ZKMH	1	$n+m$	0
NCS_1	$n(n+m+1)$	$n+m+2$	$\frac{n}{2}$
HOMOMAC	0	0	0

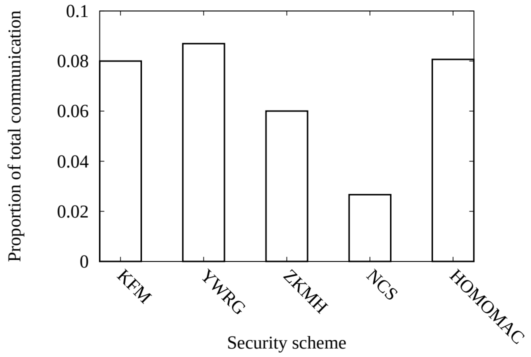


Fig. 1: Relative security overhead; values generated with parameters from Table III

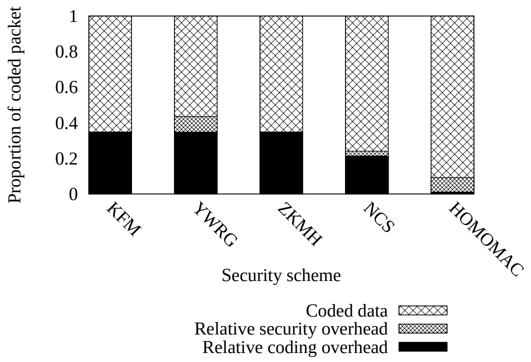


Fig. 2: Coded packet contents which show the relative coding overhead for all schemes and relative security overhead for YWRG, NCS_1, and HOMOMAC; values generated with parameters from Table III

4.3. Communication overhead

We summarize the communication overhead in Table IV. Figures 1 and 2 show concrete values for the communication overhead metrics by plugging in the values of Table III into the metrics from Table IV.

Relative security overhead. YWRG, NCS_1, and HOMOMAC induce communication overhead by appending security payloads to packets. As seen in Table IV, HOMOMAC has a potentially larger relative security overhead of $\frac{l}{n+m+l}$ as opposed to $\frac{\lambda_p}{(n+m)\lambda_q+\lambda_p}$ and $\frac{2}{n+m+2}$ for YWRG and NCS_1, respectively. However, the values of a typical scenario shown in Figure 1 demonstrate that HOMOMAC's relative security overhead is quite small due to the large size of m for HOMOMAC used in typical scenarios. KFM and YWRG induce communication overhead by distributing the security payload to forwarders at the start of each generation. The values of relative security overhead are $\frac{\lambda_p}{(n+m)\lambda_q+\lambda_p}$ and $\frac{n+m+1}{(n+1)(n+m)+1}$ for KFM and ZKMH, respectively. Both of these values result in a low relative security overhead for each scheme which is reaffirmed by results from a typical scenario shown in Figure 1.

Relative coding overhead. It may appear from Table IV that all schemes have a similar relative coding overhead $\frac{n\lambda_q}{(n+m)\lambda_q+\lambda_s}$, and $(n+m)\lambda_q+\lambda_s=\Omega$. However, the value λ_q varies for each scheme and has a major impact on relative coding overhead. Figure 2 shows that roughly one third of a coded packet is required for the coding vector in KFM, YWRG, and ZKMH, and roughly one fourth of a coded packet is required for the coding vector in NCS_1.

Overall, relative coding overhead is the dominating factor of communication overhead for KFM, YWRG, ZKMH, and NCS_1. These schemes suffer from large relative coding overhead due to the large values imposed on symbol size by security requirements.

4.4. Computation overhead

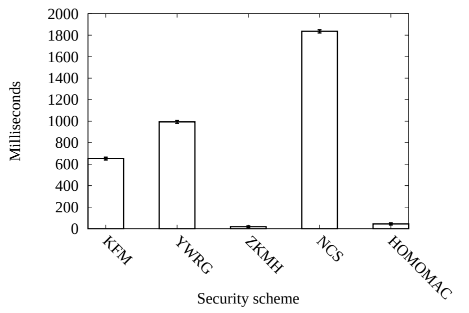
We present the overhead of signing, verifying, and combining both in terms of the number of expensive operations from Table V and in terms of time required from Figure 3 deduced from benchmarking results.

For benchmarking we implemented the operations of the signing, verifying, and combining steps for each scheme by using OpenSSL [OpenSSL 2010] and pbc [PBC 2010] libraries. The parameter values from Table III are used in our benchmarking. Figure 3 shows the results which are averaged over 100 runs on an Intel 2.2 GHz Linux machine.

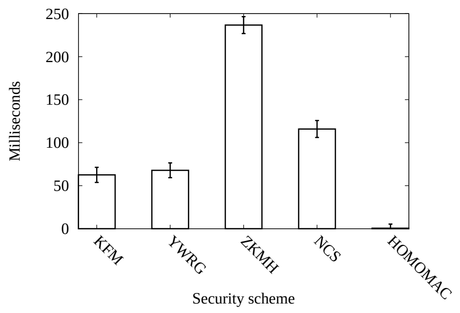
Signing. As seen in Table V, YWRG and NCS_1 require the most signing operations $n(n+m+1)$ with KFM following them at nm operations. These are significantly larger when comparing with 1 and 0 for ZKMH and HOMOMAC respectively. The benchmarking results of a typical scenario for signing are shown in Figure 3a. These results reveal a similar trend to our equations derived analytically; however, they do show that HOMOMAC has a larger signing time than ZKMH even though only modular multiplications are required for HOMOMAC. The creation of multiple MACs per plain packet results in a significant number of modular multiplications. These benchmarking results show that KFM, YWRG, and NCS_1 require time on the order of seconds to sign a generation. In our simulations, we observe time on the order of hundreds of milliseconds for the insecure system to deliver an entire generation. Thus, KFM, YWRG, and NCS_1 sign data at a slower rate than MORE distributes data which will result in a loss in throughput.

Verifying. From Table V, we can see there are roughly $n+m$ verifying operations for each scheme except HOMOMAC. This is intuitive as there are $n+m$ symbols within a coded packet. The benchmarking results in Figure 3b show that ZKMH requires much more verifying time. This is due to the exponent size in the modular exponentiations for this step. From the verification step in Algorithm 4, the exponent is a series of modular multiplications which results in a value of size λ_p not λ_q. Given that verifying is performed multiple times throughout the network in any generation, the verifying time required for ZKMH will have a severe impact on the system. Note that verifying time can potentially be reduced by probabilistically verifying which sacrifices some defense as polluted packets may be forwarded.

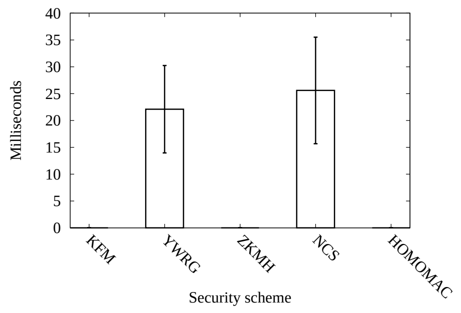
Combining. Table V shows that only YWRG and NCS_1 require costly operations for combining, an average of $\frac{n}{2}$ costly operations. The benchmarking results of Fig-



(a) Signing time



(b) Verifying time



(c) Combining time

Fig. 3: Signing time, verifying time, and combining time of each security scheme with a 95% confidence interval; values generated from benchmarking tests

ure 3c show that in a typical scenario, NCS_1 performs better than YWRG due to the nature of ECC operations compared to modular exponentiations. Combining packets occurs more often than verifying a coded packet since verification can be done in batches. Therefore, every packet sent must be delayed for the combining time which significantly penalizes YWRG and NCS_1. Overall, HOMOMAC achieves an acceptable amount of computational overhead because it does not base its security on a DLP.

HOMOMAC has another advantage in terms of computation; the symbol size of HOMOMAC is so small that the modular multiplications can be stored in a lookup table, avoiding arithmetic operations for network coding. When comparing the schemes that require a combining operation (YWRG, NCS_1, and HOMOMAC) versus those that do not (KFM and ZKMH), the schemes that do not require a combining step still incur extra computation to create packets due to network coding operations requiring several addition/multiplication operations to be computed over the large finite fields.

4.5. Resilience to multiple adversaries

HOMOMAC is the only defense scheme that is sensitive to the number of adversaries in the network. The tolerance to colluding adversaries of HOMOMAC is based on a cover-free family construction [Kumar et al. 1999; Stinson and Wei 2004]. Cover-free families are utilized to define how to distribute key-chains to forwarders such that no collusion of adversarial forwarders' key-chains can cover a certain number of keys in any legitimate node's key-chain. The number of keys not covered in an honest node's key-chain controls the probability that an adversary can successfully forge a coded packet that passes this nodes verification test. Increasing the tolerance to more byzan-

Table VI: HOMOMAC's overhead vs Byzantine adversaries tolerated

Byzantine adversaries tolerated	ω	l	Relative security overhead
2	11	121	.08
4	17	306	.20
6	23	552	.37
8	29	870	.58
10	37	1406	.94
11	39	1560	N/A

tine adversaries increases the communication and computational overhead of HOMOMAC because the number of keys increases.

We use a combinatorial based cover-free family construction [Stinson and Wei 2004] to keep the false verification rate constant at $(\frac{1}{2})^{40}$ while increasing the tolerance to byzantine adversaries to reveal the increased communication overhead. The results in Table VI show that the size of the key-chain ω grows linearly with respect to the number of tolerated byzantine adversaries, while the size of the key-pool l grows quadratically with respect to the number of byzantine adversaries tolerated.

As shown in Table VI the scheme cannot tolerate more than 11 byzantine adversaries due to the limited size of the wireless transmission frame (1500 bytes), and tolerating more than 2 byzantine adversaries becomes infeasible due to the relative security overhead. The drastic increase in relative security overhead is due to the quadratic increase in the size of the key-pool with respect to the number of tolerated byzantine adversaries, and the key-pool corresponds to the number of MACs per coded packet.

5. EXPERIMENTAL EVALUATION

In the previous section we analyzed KFM, YWRG, ZKMH, NCS_1, and HOMOMAC based on each type of overhead. In this section we evaluate the schemes through simulations. We show the overall performance and overhead for a network which implements each of these schemes. First, we detail our simulation methodology which involves the routing protocol, simulator, topology, and metrics used to produce results. Then, we present the results of our simulations. Lastly, we summarize and rationalize the results.

5.1. Simulation methodology

Our experiments are based on the Glomosim [GloMoSim 2000] simulator. We use a raw link bandwidth of 5.5 Mbps and 802.11 as the MAC layer protocol. For a realistic network topology and link quality, we use the link quality measurements from Roofnet [MIT 2006] which is a 38-node 802.11b/g mesh network at MIT.

We select a well-known, wireless, intra-flow network coding protocol, MORE [Chachulski et al. 2007], as a representative network coding system for multi-hop wireless networks. In the high level, the MORE protocol works as follows. Once coded packets are initialized for a generation, The source node continuously broadcasts coded packets for a generation until it receives an acknowledgement from each destination. Each forwarder node has a broadcast rate, i.e., the number of coded packets to broadcast per received coded packet. The selection of the broadcast rate is based on the position of the node and the overall topology of the network. Once the destination node receives enough coded packets for decoding, it sends an acknowledgement message back to the source. The acknowledgement alerts the source to initialize and start broadcasting the next generation of packets.

For each simulation, we setup a random flow in the network by designating two random nodes as the source node and the destination node. We simulate each flow without interference from other flows. Simulating flows independently creates the best possible scenario for the defense schemes as the communication and computation overhead of a given flow does not impact other flows. For a given simulation, the source begins transmission at 100 seconds until 400 seconds. We observed no significant difference in results when varying the amount of transmission time. For each defense scheme, we repeat the simulation for 200 random flows.

We use throughput as our metric for performance. Throughput is measured as the rate (in kbps) of data being decoded at the receiver. More precisely, the throughput is the total amount of data decoded at the receiver (r bits) divided by the transfer time (T seconds):

$$Throughput = \frac{r}{1000 * T} \tag{3}$$

We use latency as our metric for overhead of time taken to receive the first data at the destination. Latency is measured as the difference in time between the start of the first generation at the source and the decoding of the first generation at the destination. More precisely, latency is the time when the first generation is decoded at the destination (t_d) minus the time when the source starts transferring (t_s):

$$Latency = t_d - t_s \tag{4}$$

We use communication overhead as our metric for overhead of additional communication in the network. Communication overhead is measured as the rate (in kbps) summed over all nodes of overhead data being broadcasted where overhead data includes both security payloads and coding vectors. More precisely, communication overhead is the security payload data s_i and coding vector data v_i broadcasted by each node i is summed and then divided by the transfer time (T seconds):

$$Communication\_Overhead = \frac{\sum_i s_i + v_i}{1000 * T} \tag{5}$$

We simulate the computation overhead of a security scheme as a packet processing delay on the node as determined by the benchmarking results shown in Figure 3. The communication overhead is simulated by setting up the contents of a coded packet as specified in Figure 2. HOMOMAC is the only scheme where the security parameters differ based on the number of byzantine adversaries in the network. Thus, we represent HOMOMAC as HOMOMAC-X where X stands for the number of byzantine adversaries HOMOMAC is resilient to.

We define practicality of a secure network coding protocol (pollution defense) as the ability to outperform a secure shortest path routing protocol. For this definition, we let security be data integrity at forwarder nodes against byzantine adversaries. The purpose for switching from a shortest path routing system to a network coding system is the performance gains of network coding. Thus, a network that may have byzantine adversaries should have the same goal; that is, the purpose of switching from a secure shortest path routing system to a secure network coding system is also the performance gains of network coding. If the secure network coding system imposes high overhead and all network coding gains are negated as a consequence of the overhead, then the secure network coding system is impractical.

We determine the flows that offer possible network coding gains. Network coding gains are not possible on all flows, because network coding is most advantageous when the source and destination are far apart and many paths exist from the source to the destination. We select source and destination pairs randomly, so some sources may

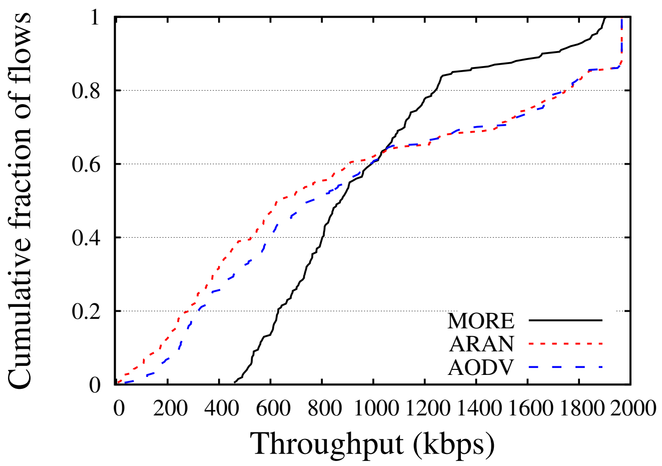


Fig. 4: Throughput CDF of MORE, AODV, and ARAN. The flows below the horizontal, dotted-dashed line represent flows in which network coding outperforms shortest path routing, and these are the only flows considered in the remainder of our experiments.

end up close to the destination. We first verify that flows in our network offer network coding gains by comparing MORE with a shortest path routing protocol, and we choose AODV [Perkins et al. 2003] as a representative shortest path routing protocol. As a shortest path routing protocol with data integrity to compare with pollution defenses, we select ARAN [Sanzgiri et al. 2002] as a secure variant of AODV. For our comparison with the pollution defenses, we exclude flows in which MORE without defense performs worse than ARAN. The flows are excluded because a pollution defense degrades the performance of MORE, and a pollution defense on top of MORE has no chance of outperforming ARAN in these cases. The schemes we compare are:

— KFM, YWRG, ZKMH, NCS_1, and HOMOMAC are the representative cryptographic pollution defenses that we are comparing in this work.
— MORE is the wireless network coding protocol that we implement pollution defenses on top of.
— AODV is the shortest path routing protocol that we use as a reference to determine that our network allows network coding gains.
— ARAN is a secure shortest path routing protocol that we use to compare the performance of representative cryptographic pollution defenses with.

5.2. Simulation results

We present the performance of three different experiments in this section. First, we show that our network does allow for network coding gains. Second, we show the performance of each scheme versus ARAN. Third, we show the performance of HOMOMAC in the presence of multiple byzantine adversaries.

Network coding performance. Our network setup does permit network coding gains in the majority of flows as the MORE protocol outperforms the best path routing protocols in many flows as shown in Figure 4. Specifically, MORE is able to outperform AODV in 106 of 200 flows. The overhead of the ARAN protocol does not reduce throughput by more than 50 kbps compared to AODV in all flows. Thus, as shown in Figure 5 the MORE protocol outperforms ARAN in 119 of 200 flows which is slightly more than compared with AODV. Upon further investigation, the flows where ARAN outperforms MORE are the more trivial flows where the source and destination are close or even

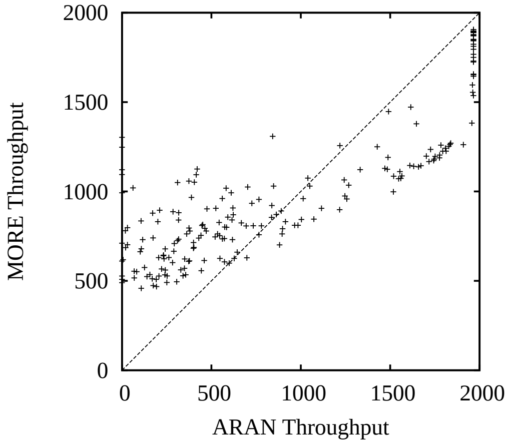


Fig. 5: Throughput scatter plot of throughput for each flow with MORE and ARAN. The values lying above the dashed line indicate a flow in which MORE outperforms ARAN, and below the line indicates that ARAN outperforms MORE.

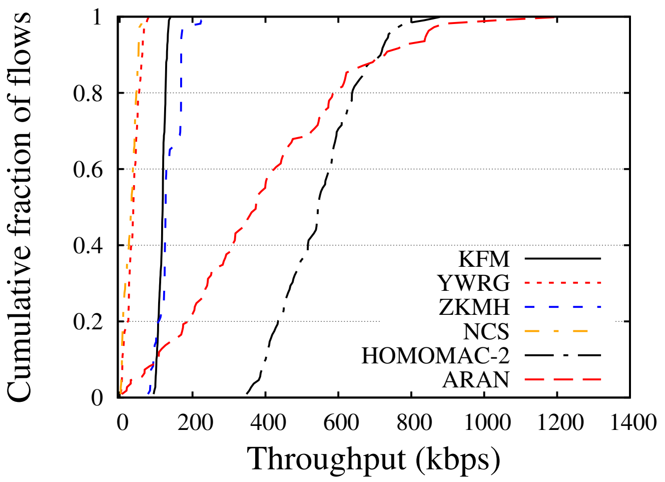


Fig. 6: Throughput CDF of the pollution defense schemes, KFM, YWRG, ZKMH, NCS$_1$, and HOMOMAC, and secure shortest path routing, ARAN. A scheme must outperform ARAN in a significant portion of flows to be considered practical; otherwise, ARAN provides better performance with the same defense.

neighbors in the network topology. In these flows, a network coding protocol has little opportunity to leverage network coding gains. In the rest of our experiments we only consider the 119 flows which MORE outperforms ARAN as these are the only flows where it is possible for MORE with the overhead of a pollution defense to outperform ARAN.

Defense scheme performance. Figure 6 shows the performance of the network coding system under different defense schemes compared with ARAN, the secure shortest path routing protocol. We observe that out of defense schemes based on asymmetric primitives, KFM, YWRG, ZKMH, and NCS$_1$, only ZKMH and KFM outperform ARAN in any flow. ZKMH and KFM perform slightly better than ARAN in only 10% of

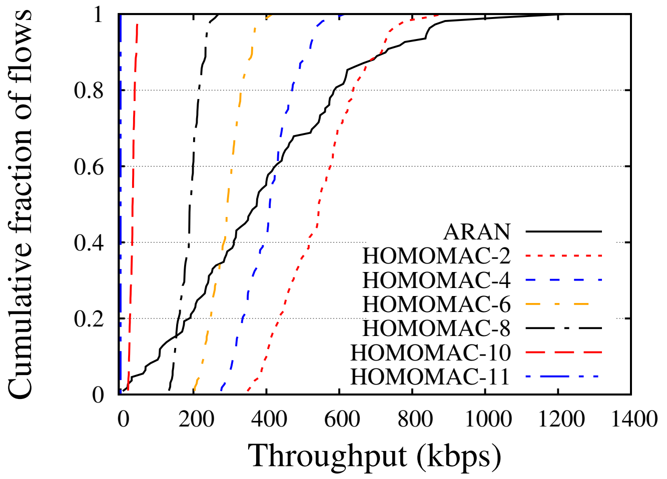


Fig. 7: Throughput CDF of HOMOMAC with resilience to multiple byzantine adversaries and ARAN.

flows. The MORE protocol with no defense had higher throughput than ARAN in every flow of this experiment; thus, these defense schemes negate the performance benefits of network coding.

HOMOMAC has an acceptable performance when resilient to two adversaries. HOMOMAC maintains the performance benefits of network coding in over 90% of flows. HOMOMAC has a drastically lower overhead than the other schemes due to the differences in computation and communication overhead seen in Figures 2 and 3.

Multiple byzantine adversaries. Only HOMOMAC's security parameters are sensitive to the number of byzantine adversaries in the network. The security parameters directly affect the performance of the system. Our experimental results in Figure 7 show that HOMOMAC performance degrades with resilience to increasing numbers of byzantine adversaries. The fraction of flows where HOMOMAC performs better than ARAN with resilience to 2, 4, 6, 8, and 10 adversaries is 90%, 60%, 35%, 15%, and 2% respectively. For 11 adversaries, the security parameters do not fit into a transmission frame anymore.

Defense scheme latency overhead. The latency results are shown in Figure 8. The schemes KFM, YWRG, ZKMH, and NCS_1 incur latency overheads that are at least twice the amount of latency overhead of MORE for the majority of flows. The latencies of the scheme ZKMH differ significantly based on the flow which can be seen by the sharp spikes in the figure. This pattern is a result of the flows varying in the number of hops and the ZKMH scheme requires higher computational overhead at each forwarder as opposed to the source. The additional latency of HOMOMAC over MORE is between 50-70 milliseconds for all flows which is quite low considering the latency of MORE.

Defense scheme communication overhead. The communication overhead results are shown in Figure 9. It is important to note that communication overhead is impacted by both the communication overhead analysis of Table IV and the throughput of a scheme. Throughput has an impact on the communication overhead because higher throughput implies more traffic in the network. Thus, it is significant that the schemes KFM, YWRG, ZKMH, and NCS_1 impose higher communication overhead than HOMOMAC and MORE given that HOMOMAC and MORE have much higher throughput as seen in Figure 6. The scheme HOMOMAC has higher communication overhead than

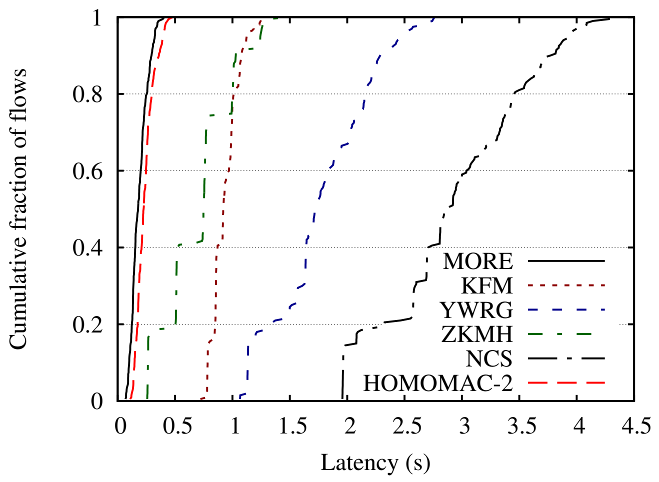


Fig. 8: Latency CDF of MORE, KFM, YWRG, ZKMH, NCS$_1$, and HOMOMAC.

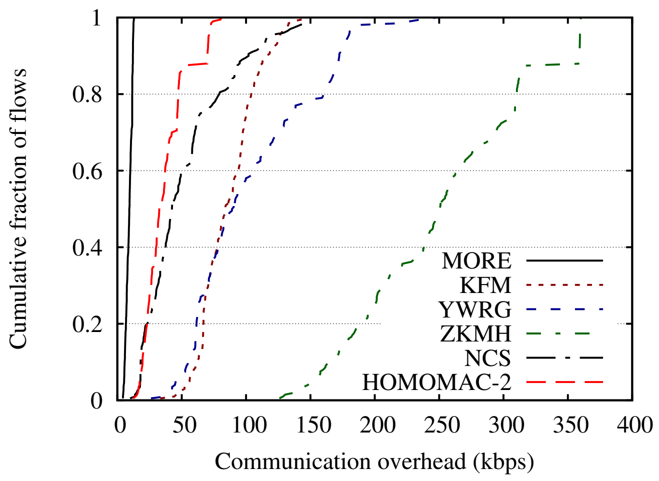


Fig. 9: Communication overhead CDF of MORE, KFM, YWRG, ZKMH, NCS$_1$, and HOMOMAC.

MORE by a factor of 5 which is due to the overhead imposed by the numerous homomorphic MACs being appended to each coded packet. The communication overhead rate of HOMOMAC is 5% of the throughput rate in the median flow which is low for many applications.

5.3. Summary of experimental evaluation

Our experiments show that the schemes KFM, YWRG, ZKMH, and NCS$_1$ are impractical due to the performance degradation they impose. The throughput is affected mostly by the computational overhead required by each scheme. Requiring seconds for signing a generation, hundreds of milliseconds for verifying coded packets that are received, and tens of milliseconds for combining coded packets results in significant throughput loss. The relative coding overhead also contributes to the poor performance by reducing overall throughput by a proportion similar to the relative coding overhead shown in Figure 2.

These same schemes KFM, YWRG, ZKMH, and NCS$_1$ also impose significantly higher overhead than MORE in terms of latency and communication. The latency overhead is mainly a result of the computationally expensive signing, verifying, and combining that must be completed before the destination can decode the first generation. The communication overhead is mainly a product of the high relative coding overhead which imposes a significant communication overhead from the coding header of each coded packet being broadcast.

HOMOMAC has better performance and lower overhead than other schemes when configured to tolerate a small number of attackers. The reasons are that HOMOMAC has a smaller lower-bound on symbol size and requires no costly operations for security steps, which results in low communication overhead and computational overhead respectively. We conclude that when a very small number of attackers exist in the network, HOMOMAC can offer a viable solution to defend against pollution attacks.

Finally, we note that HOMOMAC has a weaker security property than the other four schemes. The other schemes supply a secure verification mechanism for honest forwarder nodes in the presence of an arbitrary number of adversaries. Part of HOMOMAC's key-distribution establishes a threshold for the number of adversaries within the network, so if the number of adversaries increases beyond this threshold then the scheme does not supply a secure verification mechanism to forwarders. Increasing this threshold value imposes more communication overhead on the network. This special aspect of HOMOMAC also requires more delicate mechanisms for handling node dynamics in the network because of the need of key-distribution. For example, an adversary can mount a Sybil attack by repetitively joining the network and receive key-chains until the adversary gains the knowledge of the entire key-pool. Then, the adversary has the ability to break the security scheme and mount a successful pollution attack. We showed that in the presence of multiple attackers, HOMOMAC's configuration makes it impractical for wireless network coding systems.

6. RELATED WORK

In this section we review related work. We classify related work into information-theoretic defenses against pollution, monitoring-based pollution detection, timing-based defenses against pollution, null-space based defenses against pollution, and lattice-based defenses against pollution, and optimizations of cryptographic pollution defenses.

Information-theoretic defenses against pollution. The work [Ho et al. 2004] aims to detect that modification by a byzantine adversary has occurred after decoding at the receiver. However, detection after decoding does not prevent the loss of network resources caused by pollution attacks. Another approach [Jaggi et al. 2007] is able to reconstruct the valid coded packets at the destination in the presence of a byzantine adversary that injects invalid coded packets. The reconstruction requires sending redundant information that enforces an upper-bound on the throughput. This upper-bound is reduced by the adversary's network capacity to the receiver, that can potentially degrade the throughput to zero when the adversary has high bandwidth. The work [Wang et al. 2007] proposes to ensure high throughput with the approach of [Jaggi et al. 2007] by reducing the potential network capacity of an adversary. The solution enforces a limit of one coded packet to be broadcast per generation by each forwarder node which is inconsistent with practical wireless network coding systems.

Network error correction codes are another type of pollution defense. Work in this area [Silva et al. 2008; Yeung and Cai 2006; Cai and Yeung 2006] encodes extra information into each packet that allows for the recovery of the original packets even in the presence of invalid packets. The amount of extra encoding is dependent on the error correction ability. Thus, it is more appropriate to utilize error correction techniques to fix errors that occur rarely, such as transmission errors. It is impractical to utilize these techniques against a pollution attack because an adversary is capable of injecting many polluted packets resulting in far too many errors to correct with reasonable-sized encoding.

Work on information-theoretic defenses against byzantine nodes mainly consider linear network coding. The work [Kosut et al. 2009] shows that there are cases where nonlinear network coding achieves higher network capacity than linear network coding when some nodes are malicious. This work defines specific classes of nonlinear codes that are necessary to achieve higher network capacity than linear codes in malicious networks. Further work on this topic [Liang et al. 2009] shows that connectivity information is necessary to define a tight bound on network capacity in the presence of byzantine nodes when using nonlinear network coding.

Monitoring-based pollution detection. In a wireless setting, by monitoring the traffic in and out of a given node, it is possible to determine whether it is misbehaving in terms of its coding. The work [Kim et al. 2010] achieves high accuracy in detecting misbehavior given certain conditions. The nodes must protect their headers with error correcting codes such that multiple receivers always receive the header, and each untrusted node must have multiple trusted nodes monitoring it. Such a solution is capable of detecting the source of pollution instead of only defending against pollution.

Timing-based defenses against pollution. Broadcast authentication techniques that rely on timing have been applied for pollution defense. The works in this area both require accurate time synchronization per node and delay coded packets for verification. The work [Dong et al. 2009] proposes an alternative to cryptographic-based schemes by using an inexpensive checksum and time-based synchronization to ensure defense against pollution. The source continuously creates and disseminates a checksum appended with the time of creation (checksum messages are disseminated in an authenticated manner). Then, the forwarder nodes are able to verify all coded packets received prior to the time in the checksum message received. Because of the time synchronization, the expensive cryptographic computations are not necessary, and thus the computational overhead is reduced.

Another work based on timing [Li et al. 2010] proposes homomorphic MACs with tags based on keys that are disclosed at specified intervals. Such an approach prevents the problem of colluding attackers that the homomorphic MAC scheme [Agrawal and Boneh 2009] suffers from as well as preventing a subtle attack, tag pollution. In tag pollution, an adversary produces invalid tags that result in false positive verifications at downstream nodes which has the potential have the same detrimental effects as polluting packets. By disclosing keys such that all nodes verify the same tags prevents the invalid tag problem.

These defenses utilize computationally cheap homomorphic checksums and MACs. However, we do not consider their basis to be cryptography since without the timing aspects, it is trivial to pollute other nodes. The reliance on timing requires two major differences. First, the nodes must maintain accurate time synchronization. Second, coded packets cannot be verified immediately. To ensure a strong pollution defense, nodes must delay packets to wait for checksums or keys to be disseminated from the source. The work [Dong et al. 2009] proposed to distribute multiple generations simultaneously to mitigate these delays, but this has adverse effects on the latency. Alternatively, nodes can forward before checking packets and adaptively learn when an attack starts and only delay packets in that case as proposed in the work [Dong et al. 2009]. Timing-based solutions have additional constraints which purely cryptographic-based schemes do not have.

Null-space defenses against pollution. The work [Kehdi and Li 2009] proposes a solution to pollution attacks via null space properties. A null space is a set of null keys (vectors) such that any coded packet multiplied by a null key will result in a zero vector and multiplied by a randomly generated coded packet will result in a non-zero vector with high probability. By relying on the null space properties instead of cryptography, the scheme does not require larger symbol sizes nor does it require heavy computations. However, their security analysis does not apply to arbitrary network topologies. The scheme relies on specific topologies with path diversity so that malicious nodes do not obtain the same subspace of the null space of legitimate nodes. Constraining the topology is counter-intuitive for wireless network coding where opportunistic routing is essential for coding gains. Furthermore, the presence of byzantine adversaries further reduces the effects of path diversity.

Lattice-based defenses against pollution. Lattice-based cryptography presents many new and interesting properties [Ajtai 1996]. A work [Boneh and Freeman 2010] has leveraged the unique characteristics of lattice-based cryptography to provide a signature scheme for network coding that is secure over arbitrary sized fields and does not impose expensive computations. However, their proposed scheme suffers from both a bounded number of hops that a signature is valid for and prohibitively large signatures and public keys. If the prohibitive signature and key sizes are overcome, lattice-based cryptography offers a potential practical cryptographic solution that escapes the high relative network coding overhead and expensive computations.

Optimizations of cryptographic pollution defenses. The work [Gkantsidis and Rodriguez Rodriguez 2006] presents a cooperative security scheme that reduces the computational costs of the hashing scheme [Krohn et al. 2004] by probabilistically verifying the coded packets. In [Zhao et al. 2009], the authors show that GPUs are capable of reducing the computational costs of the cryptographic-based pollution defenses. By using GPUs, the computational steps of the homomorphic hash scheme [Krohn et al. 2004] are 38 times faster compared to the same computations on a CPU. This speedup would alleviate computational overhead of many defense schemes, but this does not address the problem of high communication overhead. Both schemes [Gkantsidis and Rodriguez Rodriguez 2006; Zhao et al. 2009] are optimizations specifically for the scheme [Krohn et al. 2004], but the techniques can be generalized to optimize other cryptographic-based schemes as well.

7. CONCLUSION

We define a framework for comparing cryptographic-based defenses against pollution attacks in a wireless environment. We use this framework to analytically compare a representative list of schemes. We then evaluate these schemes through simulations to quantify the performance of a wireless network coding system employing these schemes. Our detailed analysis and performance evaluation indicate that cryptographic defense schemes are impractical in wireless networks. Schemes utilizing an asymmetric primitive impose high communication overhead by changing the network coding parameters and high computation overhead from operations over large finite fields. The communication overhead is difficult to mitigate as increasing the size of coded packets significantly reduces the probability they are received. The computation overhead could be mitigated by probabilistically verifying which sacrifices security, but this may result in large drops in throughput when an attack occurs and it does not mitigate the computation overhead of signing and combining packets. The scheme that utilizes a symmetric primitive relies on redundancy to prevent forwarders from forging packets. The redundancy of security data is prohibitive when resilience to multiple byzantine adversaries is desired. There are possible practical solutions to pollution defense outside the area of cryptographic based defenses, but they impose one or more of the following: additional network constraints, changes to the underlying routing protocol, and decreased security.

REFERENCES

AGRAWAL, S. AND BONEH, D. 2009. Homomorphic MACs: Mac-based integrity for network coding. In *Proc. of ACNS*.

AHLSWEDE, R., CAI, N., YEN ROBERT LI, S., YEUNG, R. W., MEMBER, S., AND MEMBER, S. 2000. Network information flow. *IEEE Transactions on Information Theory*.

AJTAI, M. 1996. Generating hard instances of lattice problems. In *Proc. of ACM Symposium on Theory of Computing*.

BONEH, D., FREEMAN, D., KATZ, J., AND WATERS, B. 2009. Signing a linear subspace: Signature schemes for network coding. In *Proc. of PKC*.

BONEH, D. AND FREEMAN, D. M. 2010. Homomorphic signatures over binary fields: Secure network coding with small coefficients. `http://eprint.iacr.org/`.

CAI, N. AND YEUNG, R. W. 2006. Network error correction, part ii: Lower bounds. *Communications in Information and Systems 6*.

CHACHULSKI, S., JENNINGS, M., KATTI, S., AND KATABI, D. 2007. Trading structure for randomness in wireless opportunistic routing. In *Proc. of ACM SIGCOMM*.

CHARLES, D., JAIN, K., AND LAUTER, K. 2006. Signatures for network coding. *Proc. of 40th Annual Conference on Information Sciences and Systems*.

DAS, S., WU, Y., CHANDRA, R., AND HU, Y. C. 2008. Context-based routing: Technique, applications, and experience. In *Proc. of NSDI*.

DONG, J., CURTMOLA, R., AND NITA-ROTARU, C. 2009. Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In *Proc. of WiSec*.

GKANTSIDIS, C. AND RODRIGUEZ RODRIGUEZ, P. 2006. Cooperative security for network coding file distribution. *Proc. of INFOCOM*.

GLOMOSIM. 2000. Glomosim. `http://pcl.cs.ucla.edu/projects/glomosim/`.

HO, T., LEONG, B., KOETTER, R., MEDARD, M., EROS, M., AND KARGER, D. R. 2004. Byzantine modification detection in multicast networks using randomized network coding. In *Proc. of ISIT*.

JAGGI, S., LANGBERG, M., KATTI, S., HO, T., KATABI, D., AND MDARD, M. 2007. Resilient network coding in the presence of byzantine adversaries. In *Proc. of INFOCOM*.

JIN, J., HO, T., AND VISWANATHAN, H. 2006. Comparison of network coding and non-network coding schemes for multi-hop wireless networks. In *Proc. of ISIT*.

KATTI, S., RAHUL, H., HU, W., KATABI, D., MÉDARD, M., AND CROWCROFT, J. 2006. Xors in the air: practical wireless network coding. In *Proc. of ACM SIGCOMM*.

KEHDI, E. AND LI, B. 2009. Null keys: Limiting malicious attacks via null space properties of network coding. In *Proc. of INFOCOM*.

KIM, M., MÉDARD, M., AND BARROS, J. 2010. A multi-hop multi-source algebraic watchdog. *CoRR*.

KOSUT, O., TONG, L., AND TSE, D. 2009. Nonlinear network coding is necessary to combat general byzantine attacks. In *Proc. of Allerton conference on Communication, Control, and Computing*.

KROHN, M., FREEDMAN, M., AND MAZIÉRES, D. 2004. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proc. of S&P*.

KUMAR, R., RAJAGOPALAN, S., AND SAHAI, A. 1999. Coding constructions for blacklisting problems without computational assumptions. In *Proc. of CRYPTO*.

LE, J., LUI, J. C. S., AND CHIU, D. M. 2008. DCAR: Distributed coding-aware routing in wireless networks. In *Proc. of ICDCS*.

LETTIERI, P. AND SRIVASTAVA, M. B. 1998. Adaptive frame length control for improving wireless link throughput, range, and energy efficiency.

LI, Q., CHIU, D., AND LUI, J. 2006. On the practical and security issues of batch content distribution via network coding. In *Proc. of ICNP*.

LI, S.-Y., YEUNG, R., AND CAI, N. 2003. Linear network coding. *IEEE Transactions on Information Theory*.

LI, Y., YAO, H., CHEN, M., JAGGI, S., AND ROSEN, A. 2010. Ripple authentication for network coding. In *Proc. of INFOCOM*.

LIANG, G., AGARWAL, R., AND VAIDYA, N. 2009. Non-linear network coding against byzantine adversary: Part 1. Tech. rep., Univ. of Illinois at Urbana-Champaign.

LUN, D. S., MÉDARD, M., KOETTER, R., AND EFFROS, M. 2005. Further results on coding for reliable communication over packet networks. *CoRR*, 1848–1852,.

MIT. 2006. Roofnet. `http://pdos.csail.mit.edu/roofnet/doku.php`.

OPENSSL. 2010. Openssl. `http://www.openssl.org/`.

PBC. 2010. Pbc. http://crypto.stanford.edu/pbc/.

PERKINS, C., BELDING-ROYER, E., AND DAS, S. 2003. Ad hoc on-demand distance vector (AODV) routing.

SANZGIRI, K., DAHILL, B., LEVINE, B. N., SHIELDS, C., AND BELDING-ROYER, E. M. 2002. A secure routing protocol for ad hoc networks. *Network Protocols, IEEE International Conference on*.

SILVA, D., KSCHISCHANG, F. R., AND KOETTER, R. 2008. A rank-metric approach to error control in random network coding. *IEEE Transactions on Information Theory 54,* 9, 3951–3967.

STINSON, D. AND WEI, R. 2004. Generalized cover-free families. 463–477.

WANG, D., SILVA, D., AND KSCHISCHANG, F. R. 2007. Constricting the adversary: A broadcast transformation for network coding. In *Proc. of Allerton Conference on Communication, Control, and Computing*.

WIDMER, J. AND LE BOUDEC, J.-Y. 2005. Network coding for efficient communication in extreme networks. In *Proc. of WDTN*.

WU, Y., CHOU, P. A., AND YUAN KUNG, S. 2005. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Trans. Commun 53*, 1906–1918.

YEUNG, R. W. AND CAI, N. 2006. Network error correction, part i: Basic concepts and upper bounds. *Communications in Information and Systems 6*.

YU, Z., WEI, Y., RAMKUMAR, B., AND GUAN, Y. 2008. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proc. of INFOCOM*.

ZHANG, P., JIANG, Y., LIN, C., YAO, H., WASEF, A., AND SHEN, X. 2011. Padding for orthogonality: Efficient subspace authentication for network coding. In *Proc. of INFOCOM*.

ZHANG, X. AND LI, B. 2008a. DICE: a game theoretic framework for wireless multipath network coding. In *Proc. of MobiHoc*.

ZHANG, X. AND LI, B. 2008b. Optimized multipath network coding in lossy wireless networks. In *Proc. of ICDCS*.

ZHAO, F., KALKER, T., MÉDARD, M., AND HAN, K. 2007. Signatures of content distribution with network coding. In *Proc. of ISIT*.

ZHAO, K., CHU, X., WANG, M., AND JIANG, Y. 2009. Speeding up homomorphic hashing using gpus. In *Proc. of ICC*.