Entropy Attacks and Countermeasures in Wireless Network Coding

Andrew Newell
Purdue University
newella@cs.purdue.edu

Reza Curtmola
New Jersey Institute of Technology
crix@njit.edu

Cristina Nita-Rotaru
Purdue University
crisn@cs.purdue.edu

ABSTRACT

Multihop wireless networks gain higher performance by using network coding. However, using network coding also introduces new attacks such as the well-studied *pollution* attacks and less-studied *entropy* attacks. Unlike in pollution attacks where an attacker injects polluted packets (i.e., packets that are not linear combinations of the packets sent by the source), in entropy attacks an attacker creates non-innovative packets (i.e., packets that contain information already known by the system). In both cases the result is a severe degradation of the system performance. In this paper, we identify two variants of entropy attacks (*local* and *global*) and show that while they share some characteristics with pollution attacks and selective forwarding, none of the techniques proposed to defend against such attacks are applicable to entropy attacks because the packets look legitimate and the packet forwarding is stealthy in nature. We propose and evaluate several defenses that vary in detection capabilities and overhead.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.2.m [**Computer-Communication Networks**]: Miscellaneous—*Security*

General Terms

Security, Performance

Keywords

Network coding, Wireless, Security, Entropy attacks

1. INTRODUCTION

Traditional, wireless, multi-hop systems consist of forwarders that are constrained to only store and forward packets. Wireless, network coding systems remove this constraint and allow forwarders to use an encoding scheme that creates new packets based on stored packets. Encoding at the forwarders both maximizes the information carried by each packet and reduces the coordination necessary to deliver packets. Theoretical results [4] have shown that network coding can achieve higher network capacity than traditional networks with little coordination [15], and several practical systems [7, 9, 22, 27, 37, 38] have demonstrated significant performance increases due to network coding.

A node creates *correct* coded packets by computing a random linear combination of packets stored in its *coding buffer*. The coding buffer is the set of coded packets received correctly. A receiver is able to eventually recover the original packets if it obtains n linearly-independent coded packets generated based on original plain packets from the source. A malicious node can deviate from the standard coding procedure and conduct two types of attacks that are specific to network coding systems. The first, well-known attack is called a pollution attack [11, 13] where a node creates an *invalid* coded packet which is not a valid combination of coded packets. The second, less studied attack is called an entropy attack [13,20] where a node creates a *non-innovative* coded packet which is a non-random linear combination of coded packets such that the coded packet is *linearly dependent* with the coded packets stored at a downstream node. A linearly dependent coded packet wastes resources since it adds no useful information to help the receivers decode the original packets. We classify entropy attacks into two categories which require different capabilities from an attacker:

- A *local entropy attack* corresponds to an attacker that produces coded packets that are non-innovative to local neighboring nodes.
- A *global entropy attack* corresponds to a more capable attacker that produces coded packets that are seemingly innovative to local neighboring nodes but are non-innovative to at least one distant downstream node.

Many defenses for pollution attacks were proposed [3, 5, 8, 10, 13, 14, 19, 23, 25, 26, 28, 29, 33, 39], and they are all designed to defend against invalid coded packets but they provide no defense against attacks that use valid, but non-innovative coded packets. Cryptographic-based defenses [3, 5, 8, 10, 23, 26, 28, 29, 39] use homomorphic cryptography to detect and drop coded packets that are not valid linear combinations of the source data. Non-innovative packets will pass such verifications since they are valid combinations of the source data. Information theoretic defenses [14, 19, 33] rely on sending additional redundant information to correct invalid coded packets at the receiver. This additional information does not provide any benefit against an attacker that creates non-innovative packets because the added redundancy will only

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WiSecâĂŹ12, April 16–18, 2012, Tucson, Arizona, USA.
Copyright 2012 ACM 978-1-4503-1265-3/12/04 ...$10.00.

help recover the non-innovative packets. Lastly, existing monitoring defenses for pollution attacks [24, 25] focus on detecting invalid coded packets by comparing the packets received and sent by a node. While such schemes can detect simple types of entropy attacks in which the attacker for example sends the same valid packet repeatedly, they can not detect global entropy attacks.

Previous work [13, 20] showed that receivers waste resources to process non-innovative packets. However, entropy attacks cause more damage to a network than just occupying these resources. An entropy attack disrupts routing the same way selective forwarding attacks [21] disrupt routing in a network. In both cases, the routing algorithm chooses an optimal route or multiple routes to send data on, but the attacking node refuses to participate correctly in the forwarding of packets which prevents information transfer along one or more paths. While the effect of the entropy and selective forwarding attacks are similar, defenses against selective forwarding attacks [21, 30, 35] can not be directly applied to entropy attacks because entropy attackers actually send packets and, in the case of global entropy attacks, the attack can not be locally detected and global information is needed. Multi-path defenses [21] rely on sending redundant information along multiple paths. Network coding routing inherently sends on multiple paths, but performance can still significantly be degraded by entropy attacks since a compromised node can still deny flow on a fraction of the paths. ACK-based defenses [35] are not applicable for entropy attacks since they require nodes to acknowledge that they have received packets and do not provide mechanisms to detect if those packets were innovative or not. Existing monitoring defenses [30] require that watchdog nodes receive a fraction of traffic in and out of a suspected node to make an accurate decision of misbehavior. The approach will not work for entropy attacks since the watchdog must receive all packets that the watched node receives, otherwise it cannot determine if the newly created packets are innovative.

In this paper we study entropy attacks and their impact on wireless network coding systems. Specifically:

- We classify entropy attacks based on attacker's capabilities into *local entropy attacks* and *global entropy attacks*. We introduce a new attack, the *global entropy attack* in which the attacker generates coded packets that seem to be innovative to immediate neighbors, but are non-innovative for nodes that are further downstream. We demonstrate via simulation the negative impact of entropy attacks on network performance.
- We propose a defense scheme against local entropy attacks. The scheme, Non-innovative Link Adjustment (NLA), routes around attackers by adjusting the link quality for each link based on the percentage of non-innovative packets they carry. We show that while NLA works well for local entropy attacks, is not effective for global entropy attacks.
- We propose two defenses to address global entropy attacks. In the first defense, Upstream Buffer Propagation (UBP), downstream nodes share information about received coded packets with upstream nodes such that immediate downstream neighbors of an attacker can detect the attack. In the second defense, Buffer Monitoring (BM), watchdog nodes monitor forwarder nodes to ensure that broadcast coded packets are random linear combinations of all received coded packets, and the coefficients of this linear combination are chosen according to a publicly known pseudo-random function. BM is essentially different from typical monitoring techniques since watchdogs need to know every packet received by a forwarder. The defenses differ in terms of detection efficacy and overhead cost.
- We analytically compare the security strength of the UBP and BM defense schemes. We are able to quantify the capabilities of an attacker under each global entropy defense.
- We use a real network topology to analyze the applicability of the BM monitoring-based global entropy defense since not every flow in a topology may have sufficient wireless links to monitor traffic. We find that for the real network topology we used the monitoring-based defense can be applied to only 84.7% of flows due to constraints of the topology.

2. RELATED WORK

Entropy attacks. Entropy attacks have been considered for network coding systems, but defenses have been proposed only to mitigate the overhead of transferring non-innovative coded packets [13, 20]. These works do not consider neither the impact on routing nor the possibility of a global entropy attack. In [13], the authors propose additional local coordination in a peer-to-peer network coding system prior to obtaining a coded packet to ensure it is innovative. The authors of [20] are concerned with the additional computation required at a node to determine whether a received coded packet is innovative or not. Their solution is to probabilistically check the linear independence of a coded packet which ensures that non-innovative coded packets are dropped immediately using minor computation.

Selective forwarding attacks. Wireless mesh network security have considered the effects of a Byzantine adversary conducting a selective forwarding attack [21], where a malicious node refuses to forward some packets it receives. Such an attack can cause significant damage to network performance. Monitoring is a suitable solution to detect selective forwarding in a wireless network [30]. Nodes that neighbor an attacker can detect when the attacker has received a packet and not forwarded the packet. One of the defense schemes we propose against global entropy attacks also relies on monitoring. Unlike in defenses against selective forwarding attacks, using monitoring to defend against entropy attacks is more challenging because the attacker in an entropy attack is still forwarding packets, but the neighboring nodes need more information to determine that the attacker is coding non-innovative coded packets.

Wormhole attacks. The global entropy attack that we introduce in this paper may use an out-of-band communication channel between nodes just as in a wormhole attack [16, 17, 34]: An upstream and downstream node collude by using coded packets received at the downstream node to create new coded packets at the upstream node. Existing defenses against wormhole attacks focus on individual packets which cannot be applied to network coding as packets are combined by forwarders. In [17], temporal and geographical leashes are placed on packets to ensure they are correctly forwarded through the network. Such a technique cannot be applied to network coding since packets are coded together and each forwarder creates new packets.

Pollution attacks. Many other works also consider security of network coding systems that contain Byzantine adversaries conducting pollution attacks [3,5,8,19,26,28,33,36,39]. In a pollution attack, a node creates coded packets that are not valid linear combinations of the source's data. All of the defenses against pollution attacks rely on the fact that an invalid coded packet is not a valid linear combination. Thus, a pollution defense is not helpful against an entropy attack because entropy attackers are still creating coded packets that are valid linear combinations of the source's data.

3. SYSTEM AND ADVERSARIAL MODEL

In this section, we describe intra-flow network coding based on random linear network coding and specifically define entropy attacks in such network coding systems.

3.1 Random Intra-flow Network Coding

A general intra-flow network coding system consists of a *source* and multiple *forwarders* (destinations can be thought of as a special case of a forwarder). The source represents data to be sent as a matrix $\mathbf{B}$ of n linearly independent rows. The matrix $\mathbf{B}$ is created such that any collection of n linear combinations that are linearly independent can be transformed into $\mathbf{B}$ with gaussian elimination. Such a creation of $\mathbf{B}$ is possible by appending coding headers to each row. The source continuously broadcasts *coded packets* $\mathbf{c}(t)$ that are random linear combinations of $\mathbf{B}$:

$$\mathbf{c}(t) = \mathbf{r}(t) * \mathbf{B} \tag{1}$$

The vector $\mathbf{r}(t)$ is the random vector used to create $\mathbf{c}(t)$ at time t at the source. The source aims to send n coded packets to the destination such that the n coded packets are linearly independent, and this is called a generation.

Each forwarder i has a coding buffer $\mathbf{B}_i(t)$ which is a matrix such that the first rows are the set of all overheard coded packets at node i that are linearly independent when the forwarder i broadcasts a coded packet at time t, and the rest of the rows are zero such that the total number of rows is n. The t component is necessary because the number of non-zero rows in the coding buffer typically grows over time as more coded packets are received. Forwarders have a condition that defines when to forward a packet. E.g., in MORE[7] a forwarder will forward when it has received a sufficient (depending on the topology) number of coded packets. When this condition is met at time t for forwarder i, the forwarder generates broadcasts the coded packet $\mathbf{c}_i(t)$:

$$\mathbf{c}_i(t) = \mathbf{r}_i(t) * \mathbf{B}_i(t) \tag{2}$$

The vector $\mathbf{r}_i(t)$ is the random vector used to create the coded packet $\mathbf{c}_i(t)$ at time t at node i.

The destinations eventually receive n linearly indendent coded packets. The destinations can decode by performing gaussian elimination on their coding buffers which will result in the original $\mathbf{B}$ matrix from the source.

3.2 Entropy Attacks

We define two classes of entropy attacks, local and global. A global entropy attacker is capable of overhearing traffic on a link that is located several hops downstream. Such overhearing is possible if the attacker has a more advanced antenna for reception or cooperates with another wireless device that is located near the link that must be eavesdropped. A global entropy attacker requires a much more sophisticated defense to deal with. We will motivate via simulation in Section 6 the need to create sophisticated defenses for detecting the most capable entropy attackers.

Local Entropy Attacks. A local entropy attacker creates coded packets that are non-innovative to neighboring nodes. Such an attacker creates non-innovative coded packets by refusing to code optimally as the optimal is creating a coded packets that is a random linear combinations of all received coded packets. Specifically, we define a local entropy attacker as a forwarder i that deviates from the protocol at some time t by creating a coded packet $\bar{\mathbf{c}}_i(t)$:

$$\bar{\mathbf{c}}_i(t) = \bar{\mathbf{r}}_i(t) * \mathbf{B}_i(t) \tag{3}$$

The vector $\bar{\mathbf{r}}_i(t)$ is an arbitrary vector chosen by the attacker. A random linear network coding protocol dictates that coded packets are combined randomly, but the attacker deviates by choosing a non-random vector $\bar{\mathbf{r}}_i(t)$. Specifically, a non-random vector is a vector such that at least one element is not chosen randomly, while a random vector has every element chosen randomly.

Global Entropy Attacks. A global entropy attacker uses global information about what coded packets have been sent in the network to create coded packets that are seemingly innovative to local nodes but are non-innovative to some distant downstream node. These coded packets are also created by refusing to create random linear combinations of received coded packets but also by including a combination of coded packets from some other portion of the network to deceptively cause local neighbors to believe the coded packet is innovative. Specifically we define a global entropy attacker as a forwarder i that deviates from the protocol at time t by creating coded packets $\bar{\mathbf{c}}_i(t)$:

$$\bar{\mathbf{c}}_i(t) = \bar{\mathbf{r}}_i(t) * \mathbf{B}_i(t) + \mathbf{d}_i(t) \tag{4}$$

The vector $\bar{\mathbf{r}}_i(t)$ is not a random vector. The vector $\mathbf{d}_i(t)$ is not an element of the row space of $\mathbf{B}_i(t)$ but is an element of the row space of $\mathbf{B}$. That is, $\mathbf{d}_i(t)$ is a linear combination of some coded packets in the network, but it is not a linear combination of the coded packets that have been received by node i. The $\mathbf{d}_i(t)$ component of $\bar{\mathbf{c}}_i(t)$ is coded information being replayed from some other portion of the network.

4. SIMULATION METHODOLOGY

We aim to motivate the need for defenses against entropy attacks by showing the impact of entropy attacks through simulations. We conduct simulation experiments to measure the performance of a realistic system under various attack and defense scenarios.

We select MORE [7] as our wireless intra-flow network coding protocol that is based on random linear network coding. The source continuously broadcasts coded packets. A node is selected as a forwarder if it lies on a path or multiple paths from the source to destination, and these paths have sufficient link qualities. Based on global link state information, each forwarder is assigned a rebroadcast ratio which determines the number of coded packets received before creating and broadcasting a new coded packet. These ratios reflect how much each node contributes to a flow. Once the destination has received a sufficient number of coded packets for a generation, the destination sends an ACK back to the source. Upon receiving the ACK, the source starts sending a new generation.

Our experiments are conducted using the Glomosim [1] simulator. We use a raw link bandwidth of 5.5 Mbps and 802.11 [18] as the MAC layer protocol. For a realistic network topology and link qualities, we use the link quality measurements from the Roofnet [2] network which is a 38-node 802.11b/g mesh network deployed on MIT campus.

An experiment consists of 200 simulations that each have a random flow. A random flow consists of a randomly chosen source and destination pair. In a given simulation, the source transfers data to the destination for 400 seconds. We measure performance for a simulation and display all 200 simulations as a Cumulative Distribution Function (CDF).

For performance, we measure throughput of the system. Throughput is the rate (in kbps) of data being decoded at the destination. More specifically, throughput is the total amount of data decoded at the destination (r bits) divided by the transfer time (T seconds):

$$Throughput = \frac{r}{1000 * T} \tag{5}$$

We select the network coding parameters to match the default settings for MORE as described in [7]. The number of rows in the matrix $\mathbf{B}$ is $n = 32$, a symbol size of 1 byte $q = 2^8$, and the size of a coded packet is 1500 bytes.

To simulate attackers, we define the specific coding behavior of attackers. Random nodes from the set of forwarders for a flow are selected as attackers, and those attackers follow the specified attack behavior.

5. LOCAL ENTROPY ATTACKS

Using the methodology described in Section 4, we show the impact of a local entropy attack on a typical network, and then we present a defense strategy.

5.1 Threat

The damage caused by a local entropy attack is similar to selective forwarding. A broadcast coded packet that is non-innovative to all neighboring nodes is equivalent to a node not broadcasting a coded packet at all. The local entropy attack does cause some additional damage compared to selective forwarding since bandwidth and computation of neighboring nodes is wasted to receive the non-innovative coded packets and determine that they are non-innovative.

A local entropy attack is effective because the system trusts each node to utilize the link capacities fully to deliver innovative coded packets downstream. Thus, when a local entropy attacker is located at an important position on a path or multiple paths between the source and destination, the system delivers many coded packets to the attacker under the false assumption that new innovative coded packets are delivered further downstream from this node.

To show the effectiveness of a local entropy attack, we conduct an experiment using the simulation methodology from Section 4 with zero, one, and two entropy attackers. These local entropy attackers choose $\bar{\mathbf{r}}_i(t) = \langle r_1, ..., r_{16}, 0, ..., 0\rangle$, so the first 16 symbols are random while the last 16 symbols are zero. Thus, the attacker codes normally the first 16 received coded packets, but any further coded packets received are never used for coding.

Figure 1(a) shows the results of the local entropy attack. Such a simple attack results in zero throughput for 43% and 70% of flows for one and two attackers, respectively. The zero throughput flows are flows where the attackers happened to cut the topology consisting of the forwarders. Even the throughput in non-zero flows for the attacking scenarios degrade throughput significantly. Roughly 15% of the flows in each case of attackers has non-zero throughput where the throughput is less than the lowest throughput for MORE without an attack. The non-zero flows are affected as well, as the median throughputs are 900, 400, and 0 kbps for 0, 1, and 2 attackers respectively.

5.2 Defense

The ideal defense against a local entropy attacker is to determine which nodes are performing the attack and remove them from the system. This is not straightforward based on local decisions since even honest nodes may unknowingly send some non-innovative coded packets. The reason is because a node knows what it has already sent, but it does not know what downstream nodes may have received from another path, and a downstream node may receive the same information along two upstream paths.

Figure 1(b) shows the proportion of received non-innovative coded packets in each flow for MORE. There is an obvious increase in non-innovative coded packet reception with attackers present, but even for the benign case some flows will contain a significant proportion of non-innovative coded packets. In these benign flows, there are multiple paths to the destination which have high packet reception probabilities, thus there is little diversity in the coded packets downstream when these paths coverage which accounts for the non-innovative coded packet receptions. For 30% of benign flows 10% of received coded packets were non-innovative. This is a total for all nodes in the network, so some links in the flow potentially carry an even larger proportion of non-innovative coded packets.

Non-innovative Link Adjustment (NLA). We propose NLA as a defense against local entropy attacks. Because honest nodes also create some non-innovative packets, we do not adopt a strict node removal strategy. Instead, we punish each link proportionally with the amount of non-innovative coded packets sent on the link. The modified link qualities are obtained by multiplying the original link quality with the proportion of received innovative coded packets to total coded packets on a link. A forwarder notifies the network if it notices a significant change in the modified link qualities. Thus, when a local entropy attacker sends non-innovative coded packets, the routing layer is alerted that specific links are not carrying innovative coded packets. For MORE, nodes recalculate their rebroadcast ratios based on the modified link states which will route data on paths that avoid the attacker node. Performing this securely requires mechanisms that limit the ability of an attacker to falsely accuse other nodes. We assume that such mechanisms are in place and they are out of the scope of this work.

Figure 1(c) demonstrates how the local entropy attack is mitigated by NLA. As a baseline for the defense, we include an ideal defense where the entropy attacker is removed from the network. We removed flows (31 of 200) where the entropy attacker partitions the network from source to destination as there does not exist any set of forwarders that provides positive throughput. Without a defense, 30% of flows result in zero throughput, and these are cases where the entropy attacker partitions the set of forwarders chosen by the routing logic. With NLA, only 5% of flows result in zero throughput because in the majority of cases where the

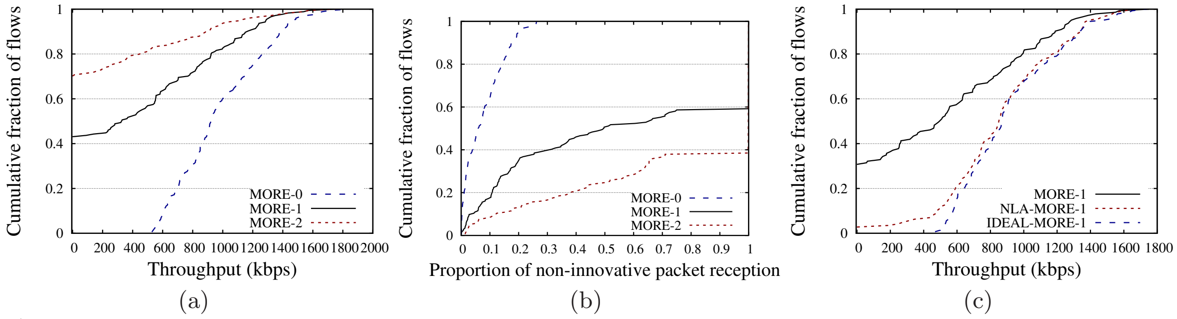

Figure 1: MORE-x is the standard wireless network coding protocol with x local entropy attackers randomly selected on each flow. NLA-MORE-x is MORE-x with the defense NLA. IDEAL-MORE-x is MORE-X with the x local entropy attackers removed from the system which represents the ideal defense against an entropy attack. (a) Throughput CDF of network coding with varying number of entropy attackers. (b) Proportion of non-innovative coded packet reception (summed over all nodes) CDFs for varying numbers of local entropy attackers. (c) Throughput CDF of network coding with one local entropy attacker for no defense, the NLA defense, and the ideal defense.

Table 1: Throughput results of various attack and defense scenarios with the topology in Figure 2

Defense	Throughput (kbps)
None	233
NLA	214
Ideal	345

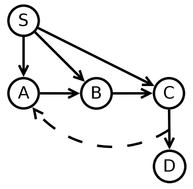

Figure 2: S is the source, D is the destination, A is the global entropy attacker. The solid lines indicate standard wireless links, while the dashed line indicates that node A can overhear the coded packets on the link between C and D.

entropy attacker partitions the initial set of forwarders the NLA defense severely punishes the links outgoing from the malicious node such that the routing logic chooses a new set of forwarders which can route data around the malicious node. With the exception of the lowest performing 20% of flows, the NLA is capable of performing within 50 kbps of the cases where the local entropy attacker is removed.

6. GLOBAL ENTROPY ATTACKS

In this section we first show how a global entropy attack impacts performance and argue that the NLA defense cannot mitigate such an attack. We then propose two defenses that can defend against global entropy attacks.

6.1 Threat

We show in Figure 2 a specific global entropy attack example. A malicious node A is able to perform a global entropy attack that A's downstream neighbor B cannot detect. The malicious node A sets $\bar{\mathbf{r}}_A(t) = \mathbf{0}$ and $\mathbf{d}_A(t) = \mathbf{r}_A(t) * \mathbf{B}_Y(t)$ where $\mathbf{B}_Y(t)$ is a coded buffer created from packets that have been overhead from the link between C and D. With this setting, Equation 4 becomes:

$$\bar{\mathbf{c}}_A(t) \quad = \quad \bar{\mathbf{r}}_A(t) * \mathbf{B}_X(t) + \mathbf{d}_A(t) = \mathbf{r}_A(t) * \mathbf{B}_Y(t)$$

Node A has access to coded packets sent by C to D via some out-of-band channel as shown by the dashed line in the figure, which allows A to create $\mathbf{B}_Y(t)$ (this can be done as simple as overhearing, or by colluding with D). Coded packets broadcast by A are linear combinations of coded packets that are broadcast by C which include coded packets that C received directly from S. B has no knowledge of coded packets that S broadcasts, C receives, and B fails to receive. So, B will receive coded packets that are innovative to B's coding buffer from A, but these coded packets are not innovative to C's coding buffer. Thus, when C receives linearly dependent coded packets from B, C cannot be sure if B or A is the entropy attacker given only local information.

The global entropy attack focuses mainly on being stealthy, but it is still damaging and can reduce throughput significantly like the local entropy attack. We performed simulations with the topology from Figure 2 and used high link qualities for edges (S, A), (A, B), (B, C), (C, D), and we used lower link qualities for edges (S, B) and (S, C). Thus, the network assumes many packets are routed through the path S, A, B, C, D. We measured a throughput of 769 kbps when A is honest and 233 kbps when A is a global attacker by replaying combinations of coded packets from link (C, D). Thus, node A harms system performance, and the node still sends coded packets that are innovative to local neighbors but are not innovative to neighbors further downstream.

We apply NLA to the scenario in Figure 2 to show how it fails to prevent the entropy attack. As summarized in Table 1, the throughput is 214 kbps when NLA is applied and 345 kbps when the ideal defense is applied. The ideal defense is the case where the global entropy attacker is removed from the network. NLA actually has lower throughput compared to the case of no defense which has 233 kbps because the modified link quality of (B, C) is lowered to nearly zero and thus the network only utilizes the path S, C, D while the path S, B, C, D still provides some innovative coded packets even under a global entropy attack. This topology illustrates how a global entropy can reduce throughput by roughly 30%, but in other topologies a global entropy attack has the potential to cause greater damage. We conclude that NLA cannot defend against the global entropy attack.

6.2 Global Entropy Defenses Overview

We present two global entropy defenses. In the Upstream Buffer Propagation (UBP) defense, nodes propagate buffer information upstream to pinpoint the origin of the global entropy attack. In the Buffer Monitoring (BM) defense, nodes monitor incoming and outgoing traffic of untrusted, neighbor nodes to immediately detect any coded packets created in a non-random fashion. To contrast these two techniques, UBP is more reactive and thus has lower overhead while BM is more proactive and requires higher overhead and places constraints on the topology. These schemes require the following three wireless communication primitives:

- **Broadcast.** A message is broadcast once and neighboring nodes will receive the message probabilistically.
- **Reliable unicast.** A specific neighboring node is designated and the sender will repeatedly broadcast a message until the receiving node acknowledges the reception of the message. This primitive can be the standard 802.11 unicast.
- **Reliable multicast.** A set of neighboring nodes are designated and the sender will repeatedly broadcast a message until all receiving nodes acknowledge the reception of the message. This primitive does not exist naturally in the 802.11 protocol, but it can be realized efficiently by periodically broadcasting the message until an acknowledgement is received from each of the designated receivers. As this primitive is not standard, we analyze its overhead in Appendix B.

Both defenses propose a mechanism for nodes to make an accurate accusation that another node is a global entropy attacker. A complete solution requires an appropriate response to such an accusation which is not straightforward since an accuser may be malicious. This is a general problem for many security protocols, and it is out of the scope of this work as we can apply approaches from other work that resolve this issue. One approach from a work on a secure wireless multicast protocol [12] proposes to only remove accused nodes temporarily and limit the accusations of a node. Thus, a malicious accusation is not permanently damaging, and a malicious node cannot disrupt the network by accusing many nodes. Another approach is to use a reputation system [6,31] to lower the reputation of nodes that have been accused or have made invalid accusations. With these reputation values, a node with low reputation can be removed from the network and its accusations can be ignored as well.

6.3 Upstream Buffer Propagation

The defense is initiated by the reception of non-innovative coded packets which implies that a global entropy attack is upstream. The entropy attacker may reside along any upstream path. We can determine the entropy attacker by propagating buffer information upstream until it reaches the source of the global entropy attacker. Thus, a legitimate node i, in response to receiving non-innovative coded packets, creates a packet containing its buffer information and sends this packet upstream. When this buffer information reaches the entropy attacker, the entropy attacker has a choice to continue performing the entropy attack and be detected, or to start sending innovative coded packets. Either way, the entropy attack is mitigated.

There are two challenges to reducing the overhead of propagating buffer information upstream to make it practical. The buffer information consists of the coding headers at a node, and as a node receives more coded packets the total size of all coding headers at the node becomes large. Thus, our first challenge is to send a small constant-sized message that conveys to upstream nodes the contents of the buffer, and we do this with a special type of checksum. Our second challenge is to prevent flooding the message upstream and instead choose based on local information at each hop the upstream path that the attack is on.

6.3.1 *Null Space Checksums*

The buffer checksums utilize the *null space* of the vector space spanned by coding headers which is the space of all vectors that result in a zero when multiplied by a vector from the vector space spanned by the coding headers. We provide background on null spaces and then we explain how to use null spaces to provide a checksum for coding headers which is represented by a *coding header matrix* $\mathbf{V}_i(t)$ which is a matrix of the nonzero coding headers of $\mathbf{B}_i(t)$. We denote these checksums as *null space checksums*.

Consider a vector space A and a null space $N(A)$ of A. Given any two vectors $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{x} \in A$ and $\mathbf{y} \in N(A)$, we have:

$$\mathbf{x} * \mathbf{y}^T = 0 \tag{6}$$

The notation $\mathbf{w}^T$ is the transpose of the matrix or vector $\mathbf{w}$.

For our checksum, we consider A to be the row space of $\mathbf{V}_i(t)$ which has $R(\mathbf{V}_i(t))$ rows (we denote $R(\mathbf{X})$ as a function that returns the number of rows of the matrix $\mathbf{X}$). The vector of the null space is a vector $\mathbf{s}_i(t)^T$ such that:

$$\mathbf{V}_i(t) * \mathbf{s}_i(t)^T = \mathbf{0} \tag{7}$$

This corresponds to a linear system of $R(\mathbf{V}_i(t))$ equations and n unknowns. To find a valid $\mathbf{s}_i(t)^T$, we simply fill in $n - R(\mathbf{V}_i(t))$ symbols randomly, and we are left with $R(\mathbf{V}_i(t))$ equations and $R(\mathbf{V}_i(t))$ unknowns which is solved to fill in the remaining symbols. Thus, computing this vector is computationally inexpensive.

Given $\mathbf{s}_i(t)^T$ and the coding header matrix of another node j at a time t', $\mathbf{V}_j(t')$, we have the following probability:

$$Pr\left(\mathbf{V}_j(t') * \mathbf{s}_i(t)^T = \mathbf{0}\right) = \left(\frac{1}{q}\right)^d \tag{8}$$

The variable d is the number of rows of $\mathbf{V}_j(t')$ are linearly independent with all rows of $\mathbf{V}_i(t)$, and the variable q is the cardinality of the field that each symbols lies in (e.g., $q = 256$ for network coding over 1 byte symbols). Any row of $\mathbf{V}_j(t')$ that is linearly dependent with all rows of $\mathbf{V}_i(t)$ results in a zero if multiplied by the vector $\mathbf{s}_i(t)^T$. Any row of $\mathbf{V}_j(t')$ that is linearly independent with all rows of $\mathbf{V}_i(t)$ results in a random symbol when multiplied by the vector $\mathbf{s}_i(t)^T$. Thus, d symbols are random, and the probability that all d symbols are zero is $(\frac{1}{q})^d$.

Thus, a node j that is upstream from node i knows that if $\mathbf{V}_j(t') * \mathbf{s}_i(t) = \mathbf{0}$ then node j at time t' most likely has no innovative packets with respect to node i's coding buffer at time t. Also, if node j finds that $\mathbf{V}_j(t') * \mathbf{s}_i(t) \neq \mathbf{0}$ then j can definitely create coded packets that are innovative with respect to node i's coding buffer. The null space checksums are encapsulated in a Buffer Checksum Packet (BCP) which contains additional information required by the protocol. We show in Appendix A that computational overhead is low to generate and check these BCPs.

6.3.2 *Single Path Propagation*

Instead of sending the buffer information on all possible paths upstream, it only needs to be sent along one single path. The path can be determined by local decisions while ensuring with high probability that the global entropy attacker will be part of the path. A node attaches a sequence number to every coded packet it creates and forwards, and nodes maintain some state that allows them to determine which upstream neighbor sent the last coded packet that triggered the broadcast of a new coded packet.

To determine a single upstream path for buffer propagation, each node maintains both a sequence number and a Sequence Number Table (SNT). When a node j broadcasts a coded packet, it appends its sequence number u_j to the coded packet and then increments u_j by one. Upon receiving from upstream neighbor i a coded packet that has a sequence number u_i, node j adds an entry $\langle u_i, j, u_j \rangle$ to its SNT and removes any old entries with the same u_i.

A node j receives a BCP because it had broadcast a coded packet that was globally non-innovative which triggered the propagation of this BCP at a downstream node that may be several hops downstream. Based on its SNT, node j knows the upstream neighbor i that sent the last coded packet which was used to create the globally non-innovative coded packet at node j. Thus, the attacker is either node i or some node upstream of node i which caused i to send this packet that is globally non-innovative. The BCP is forwarded upstream to node i along with the sequence number of the coded packet that i created so that, if node i is honest, it can make an accurate decision about which upstream node it should send the BCP on.

6.3.3 Protocol Description

We now describe in Algorithm 1 the UBP defense in detail. These actions are all in addition to normal network coding actions and they are triggered by timer expiration or by packet reception. We assume each node has a public/private key pair, such that any node i can sign a message with its private key K_i which is denoted by $S_{K_i}(\cdot)$ and any other node in the network can verify this signature.

The protocol is initiated when a node receives a non-innovative coded packet and starts the propagation of a BCP upstream (lines 1-3 of receiving a coded packet). Then, an entry is created for the Upstream Accusation Table (UAT), and a timer is started for this entry (lines 4-5 of receiving a coded packet). The purpose of the UAT is to keep track of each usptream neighbor that should send an innovative coded packet since a BCP was sent to that upstream neighbor. The time that an upstream neighbor has to send an innovative coded packet is the estimate of the time taken for the BCP to propagate up to the source and then a coded packet to propagate down to j. Upon UAT expiration, the node accuses the upstream neighbor of being an entropy attacker if the upstream neighbor did not manage to send an innovative coded packet (lines 1-2 of UAT expiration).

A node that receives a BCP will first check the signature and then check whether it has innovative packets with respect to the null space checksum within the BCP (lines 1-2 of receiving a BCP). The actions taken by the node depend on whether it has innovative coded packets. If the node does have innovative coded packets with respect to the null space checksum, then the node will broadcast a coded packet and perform the appropriate updates to the SNT (lines 3-7 of receiving a BCP). Note that these same updates are applied to the SNT for every broadcast of coded packets despite it not being explicitly mentioned. In the other case, the node forwards the BCP upstream by selecting the most likely next hop that sent globally non-innovative coded packets (lines 8-14 of receiving a BCP). The forwarded BCP is modified to include the sequence number of the coded packet that is expected to have been globally non-innovative. This sequence number is known since the SNT maintains the sequence number of the coded packet received from an upstream node just before each broadcast.

The propagation of BCPs upstream continues along a path until either a malicious node refuses to keep forwarding it, a node has innovative coded packets and sends them downstream, or the BCP reaches the source. If the BCP reaches the source, then the source always has innovative coded packets and will send an innovative coded packet downstream. Thus, each node has a chance to broadcast and propagate innovative coded packets downstream before the UAT of its downstream neighbor expires. The timers for accusation should be set such that upstream nodes' timers expire first, and only the most upstream node that makes an accusation will count. So, a malicious node will be accused if it refuses to either forward the BCP upstream or forward innovative coded packets downstream.

Algorithm 1 Reactive upstream buffer propagation protocol for node j in addition to normal network coding actions

BCP: packet with contents ⟨originator, null space checksum, sequence number, originator signature⟩
UAT: table with entries ⟨upstream node, originator, null space checksum⟩
SNT: table with entries ⟨local sequence number, upstream node, upstream sequence number⟩

Received coded packet $\mathbf{c}$ from upstream neighbor k with sequence number u_k at time t

```
1: if c is non-innovative then
2:   $BCP \leftarrow \langle j, \mathbf{s}_j(t), u_k, S_{K_j}(\mathbf{s}_j(t)) \rangle$
3:   reliable_unicast($k, BCP$)
4:   add_entry_to_table($\langle k, j, \mathbf{s}_j(t) \rangle, UAT$)
5:   start_timer($\langle k, j, \mathbf{s}_j(t) \rangle$)
6: else
7:   remove_entry_from_table($\langle u_j, *, * \rangle, SNT$)
8:   add_entry_to_table($\langle u_j, k, u_k \rangle, SNT$)
9:   if $\exists \langle k', i, \mathbf{s}_i(t) \rangle \in UAT$ s.t. $k' = k$ then
10:    if c is innovative w.r.t. $\mathbf{s}_i(t)$ then
11:      remove_entry_from_table($\langle k, *, * \rangle, UAT$)
```

Received BCP $\langle i, \mathbf{s}_i(t'), u'_j, S_{K_i}(\mathbf{s}_i(t')) \rangle$ at time t from node l

```
1: if $S_{K_i}(\mathbf{s}_i(t'))$ is correct then
2:   if $\mathbf{V}_j(t')$ has innovative coded packets w.r.t $\mathbf{s}_i(t')$ then
3:     $c \leftarrow$ create_coded_packet()
4:     broadcast($\langle c, u_j \rangle$)
5:     $\langle u_j, k, u_k \rangle \leftarrow$ get_entry_from_table($\langle u_j, *, * \rangle, SNT$)
6:     $u_j \leftarrow u_j + 1$
7:     add_entry_to_table($\langle u_j, k, u_k \rangle, SNT$)
8:   else
9:     $\langle u'_j, k, u_k \rangle \leftarrow$ get_entry_from_table($\langle u'_j, *, * \rangle, SNT$)
10:    $BCP \leftarrow \langle i, \mathbf{s}_i(t'), u_k, S_{K_i}(\mathbf{s}_i(t')) \rangle$
11:    reliable_unicast($k, BCP$)
12:    if $k$ is not source then
13:      add_entry_to_table($\langle k, i, \mathbf{s}_i(t) \rangle, UAT$)
14:      start_timer($\langle k, i, \mathbf{s}_i(t) \rangle$)
```

Expired timer $\langle k, i, \mathbf{s}_i(t) \rangle$ s.t. $\langle k, i, \mathbf{s}_i(t) \rangle \in UAT$

```
1: if no recent accusations with same originator $i$ then
2:   accuse($k$)
```

6.4 Buffer Monitoring (BM)

We now present a monitoring-based solution to defend against entropy attacks. Each forwarder is assigned one or more watchdogs. A larger number of watchdogs provides resilience to watchdog failure or misbehavior. The watchdog nodes will ensure that coded packets broadcast by a watched forwarder are random linear combinations of all received coded packets, and the coefficients of this linear combination are chosen according to a publicly known pseudo-random function. This scheme is proactive in nature and thus can immediately detect an attack. However, as any proactive scheme, it has additional overhead for each coded packet broadcast. In addition, there are some network topology constraints that might prevent some flows from having each forwarder assigned the desired number of watchdogs.

For a watchdog to determine whether a coded packet broadcast by a watched forwarder is random linear combinations of all received coded packets by that forwarder, the watchdog must know about all coded packets received by that forwarder for the generation. This poses two challenges. First, the watchdogs must have wireless links to both the watched node and every upstream neighbor of the watched node, which may prohibit BM from being applied to certain topologies. This challenge is a fundamental constraint imposed by the topology, and we analyze in Section 7.2 the feasibility of selecting watchdogs in a realistic wireless network. Second, once a valid set of watchdogs are chosen, we need to send minimal amount of data to the watchdog to ensure accurate detection while not hindering the opportunistic routing of the random network coding system.

6.4.1 Detection at a Watchdog

To determine whether a single coded packet $\mathbf{c}_i(t)$ from a node i at time t is consistent with traffic that entered node i, the watchdog must determine the coefficients $\mathbf{r}_i(t)$ used to create the coded packet from the equation:

$$\mathbf{r}_i(t) * \mathbf{B}_i(t) = \mathbf{c}_i(t) \tag{9}$$

This is an overconstrained system of $n + m$ equations and n unknowns. Only n equations are needed to determine the relevant elements of $\mathbf{r}_i(t)$. There are some elements of $\mathbf{r}_i(t)$ that correspond to rows of zero vectors in $\mathbf{B}_i(t)$ which cannot be determined, but these are irrelevant elements as they do not affect the coded packet being broadcast.

The impact of this result is that a watchdog only requires the coding headers of the coded packets sent and received by a watched forwarder. A coded packet with typical network coding system parameters has 32 bytes for the coding header while the entire coded packet is 1500 bytes. It is important to only reliably multicast a small portion of the total traffic since random network coding systems gain many advantages by forwarding data opportunistically instead of sending the data reliably each hop. This fundamental characteristic of random network coding systems is still retained with the additional overhead of sending a small portion of a coded packet, the coding header, with reliable multicast. However, simply determining the value $\mathbf{r}_i(t)$ does not completely defend against global entropy attacks as it is difficult to determine whether the values are chosen randomly or to cause a subtle entropy attack.

Instead of attempting to determine whether a $\mathbf{r}_i(t)$ used by a watched forwarder is truly random, we require all nodes to generate the coding coefficients based on a Pseudo-Random Function (PRF). The PRF is keyed with a key known to all nodes in the network (to guarantee the coefficients are pseudo-random, the key only needs to be picked at random and does not need to be secret). The inputs to the PRF are

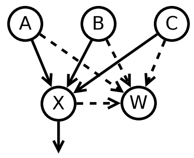

Figure 3: Scenario for monitoring solution. X is the node being monitored, W is the watchdog node, and A, B, C are upstream neighbors of X. Solid lines indicate wireless links of the topology that are used for routing data. Dashed lines indicate wireless links of the topology to send data to the watchdog node.

the node's ID along with a sequence number for the packet. The usage of a PRF makes a watchdog's job simple and deterministic to check whether a set of coefficients used by the watched forwarder is truly random. Also, the inputs to the PRF cannot be controlled by the attacker as a sequence number increases by one with each broadcast coded packet and the node ID does not change. Due to this constraint, the global entropy attacker has no opportunity to guess inputs to the PRF that may produce coefficients that result in an entropy attack. The use of the PRF is computationally inexpensive, and the random coefficients chosen by the PRF are uniformly random which is optimal to satisfy the high decoding probabilities in random network coding systems.

6.4.2 Protocol Description

Figure 3 shows an example of a node with one watchdog and three upstream neighbors. The node X is being watched by a watchdog W. The watchdog must receive all coding headers from the upstream neighbors A, B, and C along the wireless links indicated by the dashed lines. Also, the watchdog must receive the coding headers that are broadcast by node X. With this information, along with knowledge of a global PRF used by each node, the watchdog can deterministically check whether node X is correctly creating coded packets or is performing an entropy attack.

Algorithm 2 describes the specific actions of a node j in a monitoring defense. The node j is a forwarder, a watchdog, or both. We use the notation of two sets that exist for each node (these sets may be empty): $W(i)$ are the watchdogs of node i and $D(i)$ are the downstream neighbors of node i.

To ensure that only a small portion of each coded packet is sent reliably, the coding headers and coded data are sent separately in a Coding Header Packet (CHP) and Coded Data Packet (CDP). The CDP is broadcast unreliably (lines 1-2 of broadcasting a coded packet) and the CHP is reliably multicast to the appropriate set of nodes (lines 3-5 of broadcasting a coded packet). The appropriate set of nodes are the downstream nodes, watchdogs of the downstream nodes, and the watchdog of the broadcasting node (line 4 of broadcasting a coded packet). The watchdogs must receive the CHP to ensure that it has been formed correctly. The downstream nodes must receive the CHP to either reconstruct the coded packet if the downstream node correctly received the CDP (lines 1-2 of receiving CHP from upstream neighbor) or to notify watchdogs that they lost the CDP with a Dropped Data Packet (DDP) (lines 3-5 of receiving a CHP from upstream neighbor). Lastly, watchdogs of downstream neighbors must receive the CHP so that they have a view of the buffer information at the downstream neighbor.

When j is a watchdog and receives a CHP from a node i where $j \in W(i)$, j must create the coding header matrix $\mathbf{V}_i$ of node j (lines 1-3 of received CHP from a watched node), and then check whether the CHP is consistent with $\mathbf{V}_i$ and the random linear combination from the PRF (lines 4-5 of

Algorithm 2 Buffer monitoring protocol for node j in addition to normal network coding actions

```
$CHP$ : packet with contents ⟨source, sequence number, coding header⟩
$CDP$ : packet with contents ⟨source, sequence number, coded data⟩
$DDP$ : packet with contents ⟨node dropping packet, source of packet, sequence number of packet ⟩
$WBT$ : table with entries ⟨ watched node, source, sequence number, coding header ⟩
$W(x)$ : set of watchdog nodes for node $x$
$D(x)$ : set of downstream neighbors for node $x$
$PRF(x,y)$ : pseudo-random function which maps $\mathbb{Z}^+ \times \mathbb{Z}^+$ to $\mathbb{F}_q^n$

Received CHP $\langle i, u_i, \mathbf{v}\rangle$ from upstream neighbor $i$
1: if Received CDP $\langle i, u_i, \mathbf{x}\rangle$ then
2:     Reconstruct coded packet $\mathbf{c} = \langle \mathbf{v}, \mathbf{x}\rangle$ and store in buffer
3: else
4:     $DDP \leftarrow \langle j, i, u_i\rangle$
5:     reliable_multicast($W(j), DDP$)

Broadcasting coded packet $\mathbf{c} = \langle \mathbf{v}, \mathbf{x}\rangle$ created by a random linear combination $PRF(j, u_j)$ of buffered packets
1: $CDP \leftarrow \langle j, u_j, \mathbf{x}\rangle$
2: broadcast($CDP$)
3: $S \leftarrow W(j) \cup D(j) \cup \left(\bigcup_{i \in D(j)} W(i)\right)$
4: $CHP \leftarrow \langle j, u_j, \mathbf{v}\rangle$
5: reliable_multicast($S, CHP$)

Received CHP $\langle i, u_i, \mathbf{v}\rangle$ from node $i$ s.t. $j \in W(i)$
1: initialize_empty_coding_header_matrix($\mathbf{V}_i$)
2: for all $\langle i, *, *, v\rangle$ in $WBT$ do
3:     add_coding_header_to_matrix($v, \mathbf{V}_i$)
4: if $\mathbf{V}_i * PRF(i, u_i) \neq \mathbf{v}$ then
5:     accuse($i$)

Received CHP $\langle k, u_k, \mathbf{v}\rangle$ from node $k$ s.t. $i \in D(k)$, $j \in W(i)$
1: add_entry_to_table($\langle i, k, u_k, \mathbf{v}\rangle, WBT$)

Received DDP $\langle i, k, u_k\rangle$ from node $i$ s.t. $i \in D(k)$, $i \in W(j)$
1: if $\langle i, k, u_k, *\rangle \in WBT$ then
2:     remove_entry_from_table($\langle i, k, u_k, *\rangle, WBT$)
3: else
4:     drop_future_receptions($\langle k, u_k, *\rangle$)
```

received CHP from a watched node). The information to perform this check is stored in the Watchdog Buffer Table (WBT) when upstream nodes send coded packets to node i (line 1 of received CHP from upstream neighbor of watched node). The WBT must be correctly modified when a node does not receive a coded packet. This is the purpose of broadcasting the DDP to watchdog nodes of j when j does not receive the corresponding CDP to a CHP. The DDP prompts the watchdogs to either remove the entry or drop a future reception of a CHP that corresponds to the dropped packet (lines 1-5 of receiving a DDP). If an attacker abuses the use of DDPs and claims to drop more coded packets than the the measured link qualities, then the watchdog can inform the routing layer of the change in link qualities which will route data around the attacker much like the modified link qualities in our NLA defense.

7. SECURITY ANALYSIS

We analyze UBP in terms of *attack strength* which denotes the proportion of time which a globally non-innovative coded packet can be sent undetected. The buffer monitoring is much stronger in terms of security since an attacker is not capable of evading detection by a watchdog. However, BM cannot be applied to an arbitrary flow of a topology and has a higher network overhead. So, we analyze the proportion of flows BM can be applied to in the Roofnet topology [2].

Table 2: Table summarizing notation, each row represents a random variable which include the notation and explanation for the random variable.

$T^{\mathbf{c}}_{i,j}$	Avg. time of a coded packet to propagate downstream from node i to j
$T^{B}_{i,j}$	Avg. time of a BCP to propagate upstream from node i to j
T^{S}_{i}	Avg. time between coded packet sends at node i
T^{E}	Exoneration time for a hybrid of UBP
$P^{R}_{i,j}$	Attack strength from node i to j under UBP
$P^{H}_{i,j}$	Attack strength from node i to j under a hybrid of UBP with an exoneration phase
$N_{i,j}$	Average number of coded packets that can be sent by node i that are globally non-innovative to downstream node j in UBP before i must send an innovative coded packet

7.1 Attack Strength of UBP

We aim to describe the attack strength in terms of the characteristics of the network. Specifically, attack strength represents the proportion of time that coded packets can be sent from an attacker that are globally non-innovative with respect to a victim's coding buffer and the attacker will not be detected as an entropy attacker. In the remainder of the time, the attacker cannot send a globally non-innovative coded packet without being detected. Also, since the sending times of coded packets are fixed by the protocol, the attack strength also represents the proportion of packets sent by the attacker that can be globally non-innovative while not being detected. The attack strength will depend on the network characteristics of the average time taken for both a coded packet (or combinations of the coded packet) to traverse downstream and a BCP to traverse upstream. These averages differ since coded packets are larger and sent opportunistically downstream, while BCPs are smaller and sent reliably upstream immediately at each hop. For this analysis, we use notation from Table 2.

Figure 4 shows the timeline of events that lead to points where an attacker can attack in UBP without detection. The scheme waits until an attack is detected downstream, and then it reacts by sending a BCP upstream along the path that contains the global entropy attacker. The attacker is detected by a timer expiring at the entropy attacker's immediate downstream neighbor which is the time it takes for the BCP to reach the source from the attacker and then an innovative coded packet to traverse downstream to the attacker. During this entire time, the attacker can consecutively send globally non-innovative coded packets and send an innovative coded packet when it knows the immediate downstream neighbor's UAT is about to expire.

We first determine the number of consecutive non-innovative coded packets that can be sent by attacker node i that target victim j without detection as:

$$N^{R}_{i,j} = 1 + \frac{T^{\mathbf{c}}_{i,j} + T^{B}_{j,src} + T^{\mathbf{c}}_{src,i}}{T^{S}_{i}} \tag{10}$$

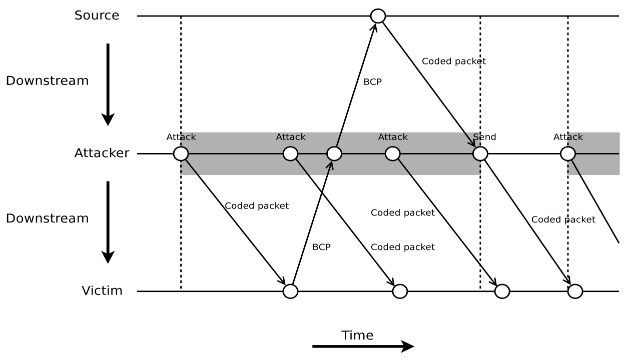

Figure 4: Example timeline for the source, attacker and victim that shows when an attacker can attack without facing detection with the UBP protocol which is denoted by the grey region. Note that there are most likely many hops from source to attacker and attacker to victim, but we assume that we can estimate the time it takes a BCP and coded packet to traverse these hops.

In addition to the one initial attack packet, there are several opportunities for the attacker to send globally non-innovative coded packets. The average number of opportunities is equal to the total time it takes before the attacker must send an innovative coded packet over the average time between coded packet sends of the attacker node.

We can then determine the attack strength from attacker node i to victim node j as:

$$P^R_{i,j} = \frac{N^R_{i,j}}{N^R_{i,j}+1} \quad (11)$$

For each consecutive non-innovative coded packet that can be sent by an entropy attacker, the entropy attacker must send one innovative coded packet to remain undetected.

The attack strength $P^R_{i,j}$ is always at least 0.5 which means that at least half of the coded packets can be globally non-innovative. The attack strength increases as the values $T^{\mathbf{c}}_{src,j}$ and $T^B_{j,src}$ become larger compared to T^S_i which happens in larger networks and when the node i does more broadcasting. The large attack strength is due to the exoneration of the attacker given just one innovative coded packet. Thus, the single upstream path found by UBP could enter an exoneration phase for a period of time which requires more overhead but detects a global entropy attack proactively. This results in a hybrid scheme with an attack strength of:

$$P^H_{i,j} = \frac{N_{i,j}}{N_{i,j}+1+\frac{T_E}{T^S_i}} \quad (12)$$

The exoneration period T_E can be varied to obtain various trade-offs between the additional overhead of the proactive detection in the exoneration period and the attack strength possible at the attacker.

An obvious way to enforce an exoneration period for a path is to assign watchdogs as in the BM scheme to these nodes. This would not impose the high overhead of BM throughout the entire network at all times as UBP can reactively determine which path an entropy attacker is on. Alternatively one could design alternate scheme that can provide an accurate proactive defense which can be used in conjunction with UBP in this same manner. We leave this as future work.

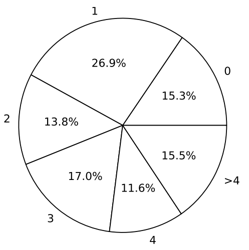

Figure 5: Given flows in the Roofnet topology we show the maximum valid assignment of n watchdogs per forwarder.

7.2 Watchdog Selection Constraints of BM

The buffer monitoring defense has a much higher level of security since it can ensure an attack strength of 0. The watchdog(s) of a forwarder have complete information about the coding headers of the forwarder, and the watchdog can deterministically assess whether the forwarder created a random combination using the entire coding buffer. This scheme cannot be employed for each flow of any topology.

The constraint for using the buffer monitoring defense is:

$$\forall f \in F, \left| \left(L(f) \cap \left(\bigcap_{u \in U(f)} L(u) \right) \right) - f \right| \geq n \quad (13)$$

F is the set of forwarders for a flow, $L(x)$ is the set of nodes that x has a wireless link to (this includes x itself), $U(x)$ is the set of upstream neighbors of node x, and n is the minimum number of watchdogs assigned to each node. This watchdog assignment allows both the nodes in the flow and nodes outside the flow to act as watchdogs for a forwarder. Furthermore, we allow a forwarders' upstream neighbor to act as its watchdog which will reduce the multicast overhead when the upstream neighbor must reliably multicast coding headers since the upstream neighbor does not need to spend communication overhead sending this coding header to itself.

We use the Roofnet data to represent a typical wireless network topology. There are 38 nodes in the network, so we take all $\binom{38}{2} = 1406$ possible flows. Out of these flows we discard 174 trivial flows that contain no forwarders and present results based on the remaining 1232 flows.

We present information about the maximum watchdog assignment per flow in Figure 5. 15.3% of flows cannot employ a buffer monitoring defense without changing the forwarder nodes that were optimally selected by the routing algorithm. These flows contain some forwarder that does not have a wireless link to any node in the topology that also has a wireless link to each upstream neighbor of the forwarder. At least one watchdog per forwarder is a minimal constraint, a network may aim to protect against an attacker that imposes false accusations. In this scenario, three watchdogs can be assigned to each forwarder to vote on detection, and only 44.1% of flows allow three watchdogs per forwarder.

8. CONCLUSION

We show via simulations the impact of entropy attacks on the overall routing of a wireless network coding system. We propose an effective defense against local entropy attacks and show the difficulties in defending against a global entropy attack. We propose two variations on a global entropy defense which differ in their defense capabilities and overhead. We provide analysis to quantify the defense capabilities of these global defense schemes.

9. REFERENCES

[1] Glomosim. `http://pcl.cs.ucla.edu/projects/glomosim/`.

[2] MIT roofnet. `http://pdos.csail.mit.edu/roofnet/doku.php`.

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

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

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

[6] S. Buchegger and J. Le Boudec. A robust reputation system for mobile ad-hoc networks. In *Proc. of P2PEcon*, 2004.

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

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

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

[10] 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.

[11] J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure network coding for wireless mesh networks: Threats, challenges, and directions. *Computer Communications*, 32(17):1790–1801, 2009.

[12] J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure high-throughput multicast routing in wireless mesh networks. *IEEE Transactions on Mobile Computing*, pages 653–668, 2010.

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

[14] T. Ho, B. Leong, R. Koetter, M. Medard, M. Eros, and D. R. Karger. Byzantine modification detection in multicast networks using randomized network coding. In *Proc. of ISIT*, 2004.

[15] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Proc. of Allerton*, 2003.

[16] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Proc. of NDSS*, 2004.

[17] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proc. of INFOCOM*, 2003.

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

[19] 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.

[20] Y. Jiang, Y. Fan, X. (Sherman) Shen, and C. Lin. A self-adaptive probabilistic packet filtering scheme against entropy attacks in network coding. *Computer Networks*, 53:3089–3101, December 2009.

[21] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293 – 315, 2003.

[22] 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.

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

[24] M. Kim, M. Médard, a. Barros, Jo and R. Kötter. An algebraic watchdog for wireless network coding. In *Proc. of ISIT*, 2009.

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

[26] 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.

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

[28] 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.

[29] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen. Ripple authentication for network coding. In *Proc. of INFOCOM*, 2010.

[30] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of MobiCom*, 2000.

[31] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[32] L. Rizzo and L. Vicisano. A reliable multicast data distribution protocol based on software fec techniques. In *Proc. of HPCS*, 1997.

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

[34] W. Wang, J. Kong, B. Bhargava, and M. Gerla. Visualisation of wormholes in underwater sensor networks: a distributed approach. *International Journal of Security and Networks*, 3(1):10–23, 2008.

[35] B. Yu and B. Xiao. Detecting selective forwarding attacks in wireless sensor networks. In *Proc. of IPDPS*, 2006.

[36] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proc. of INFOCOM*, 2008.

[37] X. Zhang and B. Li. DICE: a game theoretic framework for wireless multipath network coding. In *Proc. of Mobihoc*, 2008.

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

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

APPENDIX

A. COMPUTATION OVERHEAD

The originator of a BCP in UBP must compute a null space checksum and a signature for the BCP. The following benchmarked time values are performed on general commodity hardware[1]. As noted earlier in Section 6.3.1, creating a null space checksum requires the solution to a system of $R(\mathbf{V}_i(t))$ equations and $R(\mathbf{V}_i(t))$ unknowns where $R(\mathbf{V}_i(t)) < n$. Solving an n by n system of equations requires roughly 0.4 ms (where $n = 32$ and a symbol is 1 byte), which is the largest system of equations that may have to be solved. A single DSA sign requires roughly 1 ms of computation. Thus, overall, the originator of a BCP requires roughly 1.4 ms of computational overhead on general commodity hardware.

Nodes receiving a BCP in UBP must verify the signatures attached to these packets. Verifying a signature requires roughly 1.1 ms of computation. The reception of a BCP message requires a check of the null space checksum that was received which is simply a matrix multiplication. The computational time of a matrix multiplication on the coding headers of a coding buffer is negligible.

B. COMMUNICATION OVERHEAD

The communication overhead in UBP are the BCPs that are sent using reliable unicast due to our strategy of finding the single upstream path that the attacker is on. This communication overhead is quite small as the BCPs are small due to our use of checksums. Thus, we focus on the communication overhead of BM as it relies heavily on reliable multicasts to deliver header information reliably whenever a coded packet broadcast occurs.

Ensuring reliability on the multicast requires overhead in terms of resending the packet until each destination has received the message. Previous work exists on sending large messages with reliable multicast at the link-layer [32]. However, their key contribution is the use of forward error correction codes to break a large message into several small messages. These small messages are easier to receive since the probability of packet delivery is higher for smaller messages. Thus, the recipients just need to receive any number of small messages to reconstruct the original large message.

We reliably multicast much smaller messages that can be easily sent in one small packet (40 bytes). Thus, breaking the small message into even smaller messages will negate any performance improvements since each message has overhead of sending link-layer headers as well as physical layer overhead. Thus, we propose to send the small message multiple times until all receivers obtain the message. We analyze the number of times the message must be broadcast before each receiver obtains the message given the packet delivery probabilities on each link. We do not present analysis on the ACKs that must be sent from each receiver to the sender which would need to be sent in a way to avoid congestion.

We can analyze the number of times a message must be sent given that it is sent to N nodes over links with packet delivery probabilities of $p_1, p_2, ..., p_N$. Let X be a random variable that denotes the fewest number of times a message must be sent such that all N receivers receive the message at least once. We aim to calculate $Pr(X = k)$, so we can consider each receiver as an independent geometric random variable Y_i which corresponds to the link state p_i. We can express $Pr(X \leq k)$ in terms of the independent geometric random variables as follows:

$$Pr(X \leq k) = Pr\left(\bigcap_{i=1}^{N} Y_i \leq k\right) = \prod_{i=1}^{N} Pr(Y_i \leq k)$$

$$= \prod_{i=1}^{N}\left[\sum_{j=1}^{k} Pr(Y_i = j)\right] = \prod_{i=1}^{N}\left[\sum_{j=1}^{k}(1-p_i)^{j-1}p_i\right]$$

With $Pr(X \leq k)$ we can obtain $Pr(X = k)$ by $Pr(X = k) = Pr(X \leq k) - Pr(X \leq k-1)$. The function for $Pr(X = k)$ allows us to compute the expected number of broadcasts of a message such that each receiver obtains the message, $E[X]$. The average overhead for reliably multicasting M bytes of data will be $M * E[X]$.

We use a heuristic for summed link qualities to determine the best watchdog selection out of all possible watchdogs. Each forwarder has each downstream neighbor and the watchdogs of each downstream neighbor as recipients of a DHP. Given the link qualities from the topology and these sets of recipients we can apply the formula for $E[X]$ at each forwarder to obtain an average for a flow.

We use the Roofnet data to show the expected communication overhead when sending DHPs in BM with various number of watchdogs per forwarder. We consider the flows in Roofnet where an assignment of at least 3 watchdogs per nodes is possible (541 flows or 44.1% of non-trival flows). Figure 6 presents a CDF (Cumulative Distribution Function) for $E[X]$ of DHP reliable multicasts in BM. As expected, the overhead increases with more watchdogs being assigned to each node due to more recipients in each wireless multicast.

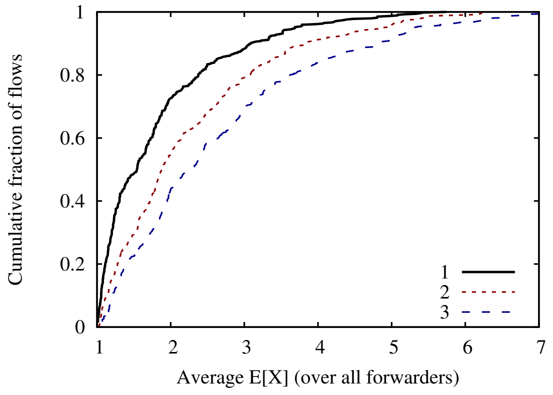

Figure 6: Given flows in the Roofnet topology, we show the communication overhead of reliable multicasts in BM varying the number of watchdogs per node to 1, 2, and 3.

[1]2.4Ghz processor and OpenSSL library for DSA signature computations