Algebraic Watchdog: Mitigating Misbehavior in Wireless Network Coding

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/jsac.2011.111202">http://dx.doi.org/10.1109/jsac.2011.111202</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers (IEEE)</td>
</tr>
<tr>
<td>Version</td>
<td>Author’s final manuscript</td>
</tr>
<tr>
<td>Citable link</td>
<td><a href="http://hdl.handle.net/1721.1/73449">http://hdl.handle.net/1721.1/73449</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike 3.0</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/3.0/">http://creativecommons.org/licenses/by-nc-sa/3.0/</a></td>
</tr>
</tbody>
</table>
Algebraic Watchdog: Mitigating Misbehavior in Wireless Network Coding

MinJi Kim*, Muriel Médard*, João Barros†

Abstract—We propose a secure scheme for wireless network coding, called the algebraic watchdog. By enabling nodes to detect malicious behaviors probabilistically and use overhead messages to police their downstream neighbors locally, the algebraic watchdog delivers a secure global self-checking network. Unlike traditional Byzantine detection protocols which are receiver-based, this protocol gives the senders an active role in checking the node downstream. The key idea is inspired by Marti et al.’s watchdog-pathrater, which attempts to detect and mitigate the effects of routing misbehavior.

As an initial building block of a such system, we first focus on a two-hop network. We present a graphical model to understand the inference process nodes execute to police their downstream neighbors; as well as to compute, analyze, and approximate the probabilities of misdetection and false detection. In addition, we present an algebraic analysis of the performance using an hypothesis testing framework that provides exact formulae for probabilities of false detection and misdetection.

We then extend the algebraic watchdog to a more general network setting, and propose a protocol in which we can establish trust in coded systems in a distributed manner. We develop a graphical model to detect the presence of an adversarial node downstream within a general multi-hop network. The structure of the graphical model (a trellis) lends itself to well-known algorithms, such as the Viterbi algorithm, which can compute the probabilities of misdetection and false detection. We show analytically that as long as the min-cut is not dominated by the Byzantine adversaries, upstream nodes can monitor downstream neighbors and allow reliable communication with certain probability. Finally, we present simulation results that support our analysis.

I. INTRODUCTION

There have been numerous contributions to secure wireless networks, including key management, secure routing, Byzantine detection, and various protocol designs (for a general survey on this topic, see [1][2][3][4][5][6][7][8]). These types of threats are particularly important in military communications and networking, which are highly dynamic in nature and must not fail when adversaries succeed in compromising some of the nodes in the network. We consider the problem of Byzantine detection. The traditional approach is receiver-based – i.e. the receiver of the corrupted data detects the presence of an upstream adversary. However, this detection may come too late as the adversary is partially successful in disrupting the network (even if it is detected). It has wasted network bandwidth, while the source is still unaware of the need for retransmission.

Reference [9] introduces a protocol for routing wireless networks, called the watchdog and pathrater, in which upstream nodes police their downstream neighbors using promiscuous monitoring. Promiscuous monitoring means that if a node $v$ is within range of a node $v'$, it can overhear communication to and from $v'$ even if those communication do not directly involve $v$. This scheme successfully detects adversaries and removes misbehaving nodes from the network by dynamically adjusting the routing paths. However, the protocol requires a significant overhead (12% to 24%) owing to increased control traffic and numerous cryptographic messages [9].

Our goal is to design and analyze a watchdog-inspired protocol for wireless networks using network coding. We propose a new scheme called the algebraic watchdog, in which nodes can detect malicious behaviors probabilistically by taking advantage of the broadcast nature of the wireless medium. Although we focus on detecting malicious or misbehaving nodes, the same approach can be applied to faulty or failing nodes. Our ultimate goal is a robust self-checking network. The key difference between the our work [10] and that of [9] is that we allow network coding. Network coding [11][12] is advantageous as it not only increases throughput and robustness against failures and erasures but also it is resilient in dynamic/unstable networks where state information may change rapidly or may be hard to obtain.

The key challenge in algebraic watchdog is that, by incorporating network coding, we can no longer recognize packets individually. In [9], a node $v$ can monitor its downstream neighbor $v'$ by checking that the packet transmitted by $v'$ is a copy of what $v$ transmitted to $v'$. However, with network coding, this is no longer possible as transmitted packets are a function of the received packets. Furthermore, $v$ may not have full information regarding the packets received at $v'$; thus, node $v$ is faced with the challenge of inferring the packets received at $v'$ and ensuring that $v'$ is transmitting a valid function of the received packets. We note that [13] combines source coding with watchdog; thus, [13] does not face the same problem as the algebraic watchdog.

The paper is organized as follows. In Section II, we briefly discuss the intuition behind algebraic watchdog. In Section III, we present the background and related material. In Section IV, we introduce our problem statement and network model. In Section V, we analyze the protocol for a simple two-hop network, first algebraically in Section V-B and then graphically in Section V-A. In Section VI, we extend the analysis for algebraic watchdog to a more general two-hop network, and in Section VIII, we present an algebraic watchdog protocol for a multi-hop network. We present simulation results in Section

This work was partially presented at IEEE ISIT 2009 (Seoul, Korea) titled “An Algebraic Watchdog for Wireless Network Coding”, and at IEEE ITW 2010 (Dublin, Ireland) titled “A Multi-hop Multi-source Algebraic Watchdog”.  

* M. Kim and M. Médard (minjikim, medard)@mit.edu) are with the Research Laboratory of Electronics at the Massachusetts Institute of Technology, MA USA. †J.J. Barros (barros@fe.up.pt) is with the Instituto de Telecomunicações, Faculdade de Engenharia da Universidade do Porto, Portugal.
II. INTUITION

Consider a network in which the sources are well-behaving (If the sources are malicious, there is no “uncorrupted” information flow to protect). In such a case, the sources can monitor their downstream neighbors as shown in Figure 1. Assume that nodes \( v_1, v_2, v_3, \) and \( v_4 \) are sources. Nodes in \( S_0 = \{v_1, v_2, v_3\} \) can monitor \( v_5 \) collectively or independently. In addition, \( v_3 \) and \( v_4 \) can monitor \( v_6 \). This enforces \( v_5 \) and \( v_6 \) to send valid information. Note that we do not make any assumption on whether \( v_5 \) and \( v_6 \) are malicious or not – they are forced to send valid information regardless of their true nature.

If it is the case that \( v_5 \) and \( v_6 \) are well-behaving, then we can employ the same scheme at \( v_5 \) or \( v_6 \) to check \( v_7 \)’s behavior. Thus, propagating trust within the network. Now, what if \( v_5 \) or \( v_6 \) are malicious? If both \( v_5 \) and \( v_6 \) are malicious, all flows to \( v_7 \) are controlled by malicious nodes – i.e. flows through \( v_7 \) are completely compromised. Therefore, even if \( v_7 \) is well-behaving, there is nothing that \( v_7 \) or \( v_1, v_3, v_4 \) can do to protect the flow through \( v_7 \). The only solution in this case would be to physically remove \( v_5 \) and \( v_6 \) from the network or to construct a new path to \( v_7 \).

The intuition is that as long as the min-cut to any node is not dominated by malicious nodes, then the remaining well-behaving nodes can check its neighborhood and enforce that the information flow is delivered correctly to the destination. For example, assume that only \( v_6 \) is malicious and \( v_5 \) is well-behaving in Figure 1. Since \( v_3 \) and \( v_4 \) monitor \( v_6 \), we know that despite \( v_6 \) being malicious, \( v_6 \) is forced to send valid information. Then, \( v_7 \) receives two valid information flows, which it is now responsible of forwarding. If \( v_7 \) is well-behaving, we do not have any problem. If \( v_7 \) is malicious, it may wish to inject errors to the information flow. In this case, \( v_7 \) is only liable to \( v_5 \); but it is liable to at least one well-behaving node \( v_5 \). Thus, it is not completely free to inject any error it chooses; it has to ensure that \( v_5 \) cannot detect its misbehavior, which may be difficult to accomplish.

In this paper, we show that this is indeed the case. We first start by studying a two-hop network, which would be equivalent to focusing on the operations performed by nodes in \( S_0 \) to check \( v_5 \). Then, we discuss how we can propagate this two-hop policing strategy to a multi-hop scenario.

III. BACKGROUND

A. Secure Network Coding

Network coding, first introduced in [11], allows algebraic mixing of information in the intermediate nodes. This mixing has been shown to have numerous performance benefits. It is known that network coding maximizes throughput for multicast [11] and increases robustness against failures [12] and erasures [14]. However, a major concern for network coded systems is their vulnerability to Byzantine adversaries. A single corrupted packet generated by a Byzantine adversary can contaminate all the information to a destination, and propagate to other destinations quickly. For example, in random linear network coding [14], one corrupted packet in a generation (i.e. a fixed set of packets) can prevent a receiver from decoding any data from that generation even if all the other packets it has received are valid.

There are several papers that attempt to address this problem. One approach is to correct the errors injected by the Byzantine adversaries using network error correction [15]. Reference [15] bounds the maximum achievable rate in an adversarial setting, and generalizes the Hamming, Gilbert-Varshamov, and Singleton bounds. Jaggi et al. [16] propose a distributed, rate-optimal, network coding scheme for multicast network that is resilient in the presence of Byzantine adversaries for sufficiently large field and packet size. Reference [17] generalizes [16] to provide correction guarantees against adversarial errors for any given field and packet size. In [18], Kim et al. compare the cost and benefit associated with these Byzantine detection schemes in terms of transmitted bits by allowing nodes to employ the detection schemes to drop polluted data.

B. Secure Routing Protocol: Watchdog and Pathrater

The problem of securing networks in the presence of Byzantine adversaries has been studied extensively, e.g. [5][6][7][8]. The watchdog and pathrater [9] are two extensions to the Dynamic Source Routing [19] protocol that attempt to detect and mitigate the effects of routing misbehavior. The watchdog detects misbehavior based on promiscuous monitoring of the transmissions of the downstream node to confirm if this relay correctly forwards the packets it receives. If a node bound to forward a packet fails to do so after a certain period of time, the watchdog increments a failure rating for that node and a node is deemed to be misbehaving when this failure rating exceeds a certain threshold. The pathrater then uses the gathered information to determine the best possible routes by avoiding misbehaving nodes. This mechanism, which does not punish these nodes (it actually relieves them from forwarding operations), provides an increase in the throughput of networks with misbehaving nodes [9].

C. Hypothesis Testing

Hypothesis testing is a method of deciding which of the two hypotheses, denoted \( H_0 \) and \( H_1 \), is true, given an observation denoted as \( U \). In this paper, \( H_0 \) is the hypothesis that \( v \) is well-behaving, \( H_1 \) is that \( v \) is malicious, and \( U \) is the information
gathered from overhearing. The observation $U$ is distributed differently depending whether $H_0$ or $H_1$ is true, and these distributions are denoted as $P_{U|H_0}$ and $P_{U|H_1}$, respectively.

An algorithm is used to choose between the hypotheses given the observation $U$. There are two types of error associated with the decision process:

- **Type 1 error, False detection**: Accepting hypothesis $H_1$ when $H_0$ is true (i.e., considering a well-behaling $v$ to be malicious), and the probability of this event is denoted $\gamma$.
- **Type 2 error, Misdetection**: Accepting $H_0$ when $H_1$ is true (i.e., considering a malicious $v$ to be well-behaling), and the probability of this event is denoted $\beta$.

The Neyman-Pearson theorem gives the optimal decision rule that gives the maximal tolerable $\beta$, we can minimize $\gamma$ by accepting hypothesis $H_0$ if and only if $\log \frac{P_{U|H_0}}{P_{U|H_1}} \geq t$ for some threshold $t$ dependant on $\gamma$. For more thorough survey on hypothesis testing in the context of authentication, see [20].

### IV. Problem Statement

We shall use elements from a field, and their bit-representation. We use the same character in italic font (i.e., $x$) for the field element, and in bold font (i.e., $\hat{x}$) for the bit-representation. We use underscore bold font (i.e., $x_i$) for vectors. For arithmetic operations in the field, we shall use the conventional notation (i.e., $+, -$). For bit-operation, we shall use $\oplus$ for addition, and $\otimes$ for multiplication.

We also require polynomial hash functions defined as follows (for a more detailed discussion on this topic, see [21]).

**Definition 4.1 (Polynomial hash functions):** For a finite field $F$ and $d \geq 1$, the class of polynomial hash functions on $F$ is defined as follows:

$$\mathcal{H}^d(F) = \{h_a | a = (a_0, ..., a_d) \in F^{d+1}\},$$

where $h_a(x) = \sum_{i=0}^{d} a_i x^i$ for $x \in F$.

We model a wireless network with a hypergraph $G = (V, E_1, E_2)$, where $V$ is the set of the nodes in the network, $E_1$ is the set of hyperedges representing the connectivity (wireless links), and $E_2$ is the set of hyperedges representing the interference. We use the hypergraph to capture the broadcast nature of the wireless medium. If $(v_1, v_2) \in E_1$ and $(v_1, v_3) \in E_2$ where $v_1, v_2, v_3 \in V$, then there is an intended transmission from $v_1$ to $v_2$, and $v_3$ can overhear this transmission (possibly incorrectly). There is a certain transition probability associated with the interference channels known to the nodes, and we model them with binary channels, $BSC(p_{ij})$ for $(v_i, v_j) \in E_2$.

A node $v_i \in V$ transmits coded information $x_i$ by transmitting a packet $p_i$, where $p_i = [a_i, h_{i1}, h_{i2}, x_i]$ is a $\{0,1\}$-vector. A valid packet $p_i$ is defined as below:

- $a_i$ corresponds to the coding coefficients $a_j$, $j \in I_i$, where $I_i \subseteq V$ is the set of nodes adjacent to $v_i$ in $E_1$.
- $h_{i1}$ corresponds to the hash $h(x_j)$, $v_j \in I_i$, where $h(\cdot)$ is a $\delta$-bit polynomial hash function,
- $h_{i2}$ corresponds to the polynomial hash $h(x_i)$,
- $x_i$ is the $n$-bit representation of $x_i = \sum_{j \in I_i} a_j x_j \in (F_2^n)$.

Figure 2 illustrates the structure of a valid packet. For simplicity, we assume the payload $x_i$ to be a single symbol. We design and analyze our protocol for a single symbol. However, the protocol applies (and therefore, the analysis) to packets with multiple symbols by applying the protocol on each symbol separately.

The payload $x_i$ is coded with a $(n, k_i)$-code $C_i$ with minimum distance $d_i$. Code $C_i$ is an error-correcting code of rate $R_i = \frac{k_i}{n} = 1 - \frac{d_i}{n}$, and is tailored for the forward communication. For instance, $v_1$ uses code $C_1$, chosen appropriately for the channel $(v_1, v_2) \in E_1$, to transmit the payload $x_1$.

We assume that the payload $x_i$ is $n$-bits, and the hash $h(\cdot)$ is $\delta$-bits. We assume that the hash function used, $h(\cdot)$, is known to all nodes, including the adversary. In addition, we assume that $a_i, h_{i1}, h_{i2}, x_i$ are part of the header information, and are sufficiently coded to allow the nodes to correctly receive them even under noisy channel conditions. Protecting the header sufficiently will induce some overhead, but the assumption remains a reasonable one to make. First, the header is smaller than the message itself. Second, even in the routing case, the header and the state information need to be coded sufficiently.

Third, the hashes $h_{i1}$ and $h_{i2}$ are contained within one hop. A node that receives $p_i = [a_i, h_{i1}, h_{i2}, x_i]$ does not need to repeat $h_{i1}$, only $h_{i2}$. Therefore, the overhead associated with the hashes is proportional to the in-degree of a node, and does not accumulate with the routing path length.

Assume that $v_i$ transmits $p_i = [a_i, h_{i1}, h_{i2}, x_i]$, where $x_i = x_i + e$, $e \in \{0, 1\}^n$. If $v_i$ is misbehaving, then $e \neq 0$. Our goal is to detect with high probability when $e \neq 0$. Even if $|e|$ is small (i.e., the hamming distance between $x_i$ and $x_1$ is small), the algebraic interpretation of $x_i$ and $x_i$ may differ significantly. For example, consider $n = 4$, $x_i = [0000]$, and $x_i = [1000]$. Then, $e = [1000]$ and $|e| = 1$. However, the algebraic interpretations of $x_i$ and $x_i$ are 0 and 8, respectively. Thus, even a single bit flip can alter the message significantly.

### A. Threat Model

We assume powerful adversaries, who can eavesdrop their neighbor’s transmissions, has the power to inject or corrupt packets, and are computationally unbounded. Thus, the adversary will find $x_i$ that will allow its misbehavior to be undetected, if there is any such $x_i$. However, the adversary does not know the specific realization of the random errors introduced by the channels. We denote the rate at which an adversary injects error (i.e., performs bit flips to the payload) to be $p_{ad}$. The adversaries’ objective is to corrupt the information flow without being detected by other nodes.

Our goal is to detect probabilistically a malicious behavior that is beyond the channel noise, represented by $BSC(p_{kk})$. Note that the algebraic watchdog does not completely eliminate errors introduced by the adversaries; its objective is to
limit the errors introduced by the adversaries to be at most that of the channel. Channel errors (or those introduced by adversaries below the channel noise level) can be corrected using appropriate error correction schemes, which will be necessary even without Byzantine adversaries in the network.

The notion that adversarial errors should sometimes be treated as channel noise has been introduced previously in [18]. Under heavy attack, attacks should be treated with special attention; while under light attack, the attacks can be treated as noise and corrected using error-correction schemes. The results in this paper partially reiterate this idea.

V. TWO-HOP NETWORK: AN EXAMPLE

Consider a network (or a small neighborhood of nodes in a larger network) with nodes \( v_1, v_2, \ldots, v_m, v_{m+1}, v_{m+2} \). Nodes \( v_i, \ i \in [1, m] \), want to transmit \( x_i \) to \( v_{m+2} \) via \( v_{m+1} \). A single node \( v_i \), \( i \in [1, m] \), cannot check whether \( v_{m+1} \) is misbehaving or not even if \( v_1 \) overhears \( x_{m+1} \), since without any information about \( x_j \) for \( j \in [1, m] \), \( x_{m+1} \) is completely random to \( v_i \). On the other hand, if \( v_i \) knows \( x_{m+1} \) and \( x_j \) for all \( j \in [1, m] \), then \( v_i \) can verify that \( v_{m+1} \) is behaving with certainty; however, this requires at least \( m-1 \) additional reliable transmissions to \( v_i \).

We take advantage of the wireless setting, in which nodes can overhear their neighbors’ transmissions. In Figure 3, we use the solid lines to represent the intended channels \( E_1 \), and dotted lines for the interference channels \( E_2 \) which we model with binary channels as mentioned in Section IV. Each node checks whether its neighbors are transmitting values that are consistent with the gathered information. If a node detects that its neighbor is misbehaving, then it can alert other nodes in the network and isolate the misbehaving node.

In the next subsections, we shall use an example with \( m = 2 \), as shown Figure 4. We introduce the graphical model which explains how a node \( v_1 \) checks its neighbor’s behavior. Then, we use an algebraic approach to analyze and compute \( \gamma \) and \( \beta \) for this example network with \( m = 2 \). In this section, we assume for simplicity that nodes do not code the payload – i.e. an error-correcting code of rate \( R_i = 1 \) is used.

Note that a malicious \( v_3 \) would not inject errors in \( h_{x_3} \) only, because the destination \( v_4 \) can easily verify if \( h_{x_3} \) is equal to \( h(x_3) \). Therefore, \( h_{x_3} \) and \( x_3 \) are consistent. In addition, \( v_3 \) would not inject errors in \( h_{x_j} \), \( j \in I_3 \), as each node \( v_j \) can verify the hash of its message. On the other hand, a malicious \( v_3 \) can inject errors in \( x_3 \), forcing \( v_4 \) to receive incorrect coefficients \( \tilde{\alpha}_3 \)’s instead of \( \alpha_3 \)’s. However, any error introduced in \( x_3 \) can be translated to errors in \( x_3 \) by assuming that \( \tilde{\alpha}_3 \)’s are the correct coding coefficients. Therefore, we are concerned only with the case in which \( v_3 \) introduces errors in \( x_3 \) (and therefore, in \( h_{x_3} \) such that \( h_{x_3} = h(x_3) \)).

A. Graphical model approach

We present a graphical approach to model the problem for \( m = 2 \) systematically, and to explain how a node may check its neighbors. This approach may be advantageous as it lends easily to already existing graphical model algorithms as well as some approximation algorithms.

We shall consider the problem from \( v_1 \)'s perspective. As shown in Figure 6, the graphical model has four layers: Layer 1 contains \( 2^{n+h} \) vertices, each representing a bit-representation of \( [x_2, h(x_2)] \); Layer 2 contains \( 2^n \) vertices, each representing a bit-representation of \( x_2 \); Layer 3 contains \( 2^n \) vertices corresponding to \( x_3 \); and Layer 4 contains \( 2^{n+h} \) vertices corresponding to \( [x_3, h(x_3)] \). Edges exist between adjacent layers as follows:

- **Layer 1 to Layer 2**: An edge exists between a vertex \([v, u]\) in Layer 1 and a vertex \( w \) in Layer 2 if and only if \( h(w) = u \). The edge weight is normalized such that the total weight of edges leaving \([v, u]\) is 1, and the weight is proportional to:

\[ P(v) \text{ Channel statistics and } w \text{ is the original message}, \]

which is the probability that the inference channel outputs message \( v \) given an input message \( w \).

- **Layer 2 to Layer 3**: The edges represent a permutation. A vertex \( v \) in Layer 2 is adjacent to a vertex \( w = c + \alpha_3 v \) in Layer 3 if and only if \( w = c + \alpha_3 x_3 \) is a constant, \( v \) and \( w \) are the bit-representation of \( v \) and \( w \), respectively. The edge weights are all 1.

- **Layer 3 to Layer 4**: An edge exists between a vertex \( v \) in Layer 3 and a vertex \([w, u]\) in Layer 4 if and only if \( h(v) = u \). The edge weight is normalized such that the total weight leaving \( v \) is 1, and is proportional to:

\[ P(w) \text{ Channel statistics and } v \text{ is the original message}. \]

Node \( v_1 \) overhears the transmissions from \( v_2 \) to \( v_3 \) and from \( v_3 \) to \( v_4 \); therefore, it receives \([x_2, h(x_2)]\) and \([x_3, h(x_3)]\),
corresponding to the starting point in Layer 1 and the destination point in Layer 4 respectively. By computing the sum of the product of the weights of all possible paths between the starting and the destination points, v₁ computes the probability that v₃ is consistent with the information gathered.

This graphical model illustrates sequentially and visually the inference process v₁ executes. Furthermore, by using approximation algorithms and pruning algorithms, we may be able to simplify the computation as well as the structure of the graph. In addition, the graphical approach may be extend to larger networks, as we shall discuss in Section VI.

B. Algebraic approach

We explain the inference process described above using the graphical model introduced in Section V-A. Consider v₁. By assumption, v₁ correctly receives a₂, a₃, h₁₂, h₁₃, h₂₂, and h₃₃. In addition, v₁ receives x₂ = x₃ + e' and x₃ = x₄ + e", where e' and e" are outcomes of the interference channels. Given x₂ for j = {2, 3} and the transition probabilities, v₁ computes r_j→₁ such that the sum of the probability that the interference channel from v_j and v₁ outputs x̃_j given x ∈ B(x_j, r_j→₁) is greater or equal to 1 - ϵ where ϵ is a constant, and B(x, r) is a n-dimensional ball of radius r centered at x. Now, v₁ computes x̃_j = {x | h(x) = h(x_j)} ∩ B(x_j, r_j→₁) for j = {2, 3}. Then, v₁ computes α₁xₙ + α₂x̃_{2→ₚ} and α_2 x̃_{2→ₚ} for all x ∈ x̃_₂. Then, v₁ intersects x̃_₃ and the computed α₁xₙ + α₂ x̃_{2→ₚ}'s. If the intersection is empty, then v₁ claims that R is misbehaving.

The set {x | h(x) = h(x_j)} represents the Layer 2 vertices reachable from the starting point ([x_j, h(x_j)] in Layer 1), and x̃_₂ is a subset of the reachable Layer 2 vertices such that the total edge weight (which corresponds to the transition probability) from the starting point is greater than 1 - ϵ. Then, computing α₁xₙ + α₂ x̃_{2→ₚ} represents the permutation from Layers 2 to 3. Finally, the intersection with x₃ represents finding a set of Layer 3 vertices such that they are adjacent to the destination point ([x₃, h(x₃)] in Layer 4) and their total transition probability to the destination point is greater than 1 - ϵ.

Lemma 5.1: For n sufficiently large, the probability of false detection, γ ≤ ϵ for any arbitrary small constant ϵ.

Proof: Assume that v₃ is not malicious, and transmits x₄ and h₃₃ consistent with v₁’s check. Then, for n sufficiently large, v₁ can choose r₂→₁ and r₃→₁ such that the probability that the bit representation of x₃ = α₁xₙ + α₂ x̃_{2→ₚ} is in x̃_₃ and the probability that x₃ ∈ x̃_₃ is greater than 1 - ϵ. Therefore, x₃ ∩ {α₁xₙ + α₂ x̃_{2→ₚ} | ∀x ∈ x̃₂} ≠ ∅ with probability arbitrary close to 1. Therefore, a well-behaving v₃ passes v₁’s check with probability at least 1 - ϵ. Thus, γ ≤ ϵ.

Lemma 5.2: The probability that a malicious v₃ is undetected from v₁’s perspective is given by

\[
\min\left\{ 1, \frac{\sum_{k=0}^{r_{3→2}} \binom{n}{k}}{2^{(h+n)}}, \frac{\sum_{k=0}^{r_{3→1}} \binom{n}{k}}{2^{(h+n)}}, \frac{\sum_{k=0}^{r_{3→1}} \binom{n}{k}}{2^{h}} \right\}.
\]

Proof: Assume that v₃ is malicious and injects errors into x₃. Consider an element z = α₁xₙ + α₂ x̃_{2→ₚ} + e = α₁xₙ + α₂ (x₄ + e₂) for some e and e₂. Note that, since we are using a field of size 2ⁿ, multiplying an element from the field by a randomly chosen constant has the effect of randomizing the product. Here, we consider two cases:

Case 1: If x₂ + e₂ ∉ x̃_₂, then v₃ fails v₁’s check.

Case 2: If x₂ + e₂ ∈ x̃_₂, then v₃ passes v₁’s check; however, v₃ is unlikely to pass v₂’s check. This is because α₁xₙ + α₂(x₄ + e₂) = α₁xₙ + α₂x₄ + α₂e₂ = α₁(xₙ + e₁) + α₂x₄ for some e₁. Here, for uniformly random α₁ and α₂, e₁ is also uniformly random. Therefore, the probability that v₃ will pass is the probability that the uniformly random vector xₙ + e₁ belongs to \( x̃_₁ = \{ x \mid h(x) = h(x₁) \} \cap B(x₁, r₁→₂) \) where v₂ overhears x̃_₁ from v₁, and the probability that the interference channel from v₁ to v₂ outputs x̃_₁ given x ∈ B(x₁, r₁→₂) is greater than 1 - ϵ.

P(A malicious v₃ passes v₂’s check)

\[
P(x₁ + e₁ ∈ x̃_₁) = \frac{\text{Vol}(x̃_₁)}{2^n},
\]

where \( \text{Vol}(\cdot) \) is equal to the number of (0,1)-vectors in the given set. Since \( \text{Vol}(B(x₁, r₁)) = \sum_{k=0}^{r₁} \binom{n}{k} \leq 2^{n₁} \), and the probability that h(x) is equal to a given value is \( \frac{1}{2^n} \), \( \text{Vol}(x̃₁) \) is given as follows:

\[
\text{Vol}(x̃₁) = \frac{\text{Vol}(B(x₁, r₁→₂))}{2^h} = \frac{\sum_{k=0}^{r₁→₂} \binom{n}{k}}{2^h}.
\]

Therefore, from v₁’s perspective, the probability that a z ∈ x̃₃ passes the checks, P(z passes check), is:

\[
0 \cdot P(x₂ + e₂ ∉ x̃_₂) + \frac{\sum_{k=0}^{r₁→₂} \binom{n}{k}}{2^{(h+n)}} \cdot P(x₂ + e₂ ∈ x̃_₂).
\]

Similarly, P(x₂ + e₂ ∈ x̃₃) = \( \frac{\sum_{k=0}^{r₂→ₚ} \binom{n}{k}}{2^{(h+n)}} \cdot \text{Vol}(x̃₃) = \frac{\sum_{k=0}^{r₂→ₚ} \binom{n}{k}}{2^{(h+n)}} \). Then, the probability that v₃ is undetected from v₁’s perspective is the probability that at least one z ∈ x̃₃ passes the check:

P(A malicious v₃ is undetected from v₁’s perspective)

\[
= \min\{1, P(\text{z passes check)} \cdot \text{Vol}(x̃₃)\}.
\]

Note that P(\text{z passes check}) \cdot \text{Vol}(x̃₃) is the expected number of z ∈ x̃₃ that passes the check; thus, given a high enough P(\text{z passes check}), would exceed 1. Therefore, we take \( \min\{1, P(\text{z passes check}) \cdot \text{Vol}(x̃₃)\} \) to get a valid probability. This proves the statement.

Lemma 5.3: The probability that a malicious v₃ is undetected from v₂’s perspective is given by

\[
\min\left\{ 1, \frac{\sum_{k=0}^{r₁→₂} \binom{n}{k}}{2^{(h+n)}}, \frac{\sum_{k=0}^{r₁→₁} \binom{n}{k}}{2^{(h+n)}}, \frac{\sum_{k=0}^{r₁→₁} \binom{n}{k}}{2^{h}} \right\},
\]

where v₂ overhears x₃ from v₃, and the probability that the interference channel from v₃ to v₂ outputs x̃₃ given x ∈ B(x₃, r₃→₂) is greater than 1 - ϵ.

Proof: By similar analysis as in proof of Lemma 5.2.

Theorem 5.1: The probability of misdetection, β, is:

\[
\beta = \min\left\{ 1, \frac{\sum_{k=0}^{r₁→₂} \binom{n}{k}}{2^{(h+n)}}, \frac{\sum_{k=0}^{r₁→₁} \binom{n}{k}}{2^{(h+n)}}, \frac{1}{2^h} \cdot \sum_{k=0}^{r₁→₁} \binom{n}{k} \right\},
\]

where \( r = \min\{r₁→₁, r₃→₂\} \).
Proof: The probability of misdetection is the minimum of the probability that \(v_1\) and \(v_2\) do not detect a malicious \(v_3\). Therefore, by Lemma 5.2 and 5.3, the statement is true.

Theorem 5.1 shows that the probability of misdetection \(\beta\) decreases with the hash size, as the hashes restrict the space of consistent codewords. In addition, since \(r_1 \rightarrow r_2, r_2 \rightarrow 1, r_3 \rightarrow 1, r_3 \rightarrow 2\) represent the uncertainty introduced by the interference channels, \(\beta\) increases with them. Lastly and the most interestingly, \(\beta\) decreases with \(n\), since \(\sum_{k=0}^{r} \binom{n}{k} < 2^n\) for \(r < n\). This is because network coding randomizes the messages over a field whose size is increasing exponentially with \(n\), and this makes it difficult for an adversary to introduce errors without introducing inconsistencies.

We can apply Theorem 5.1 even when \(v_1\) and \(v_2\) cannot overhear each other. In this case, both \(r_1 \rightarrow r_2\) and \(r_2 \rightarrow 1\) equal to \(n\), giving the probability of misdetection, \(\beta\), as shown in Figure 6. Figure 6 illustrates the transition matrix for the overheard information, which is captured by the transition matrix; however, the transition probabilities can be computed efficiently using hash collision lists as well.

The algorithm introduced above is a dynamic program, and is similar to the Viterbi algorithm. Therefore, tools developed for dynamic programming/Viterbi algorithm can be used to compute the probabilities efficiently.
D. Decision making

Node $v_1$ computes the probability that the overheard $\tilde{x}_{m+1}$ and $h(x_{m+1})$ are consistent with the inferred $w(\cdot, m)$ to make a decision regarding $v_{m+1}$’s behavior. To do so, $v_1$ constructs an inverse transition matrix $T^{-1}$, which is a $2^n(1-\frac{m+1}{2}) \times 2^{n(1-\frac{m+1}{2})+\delta}$ matrix whose elements are defined as follows:

$$T^{-1}(y, \tilde{x}_{m+1}) = \begin{cases} \frac{p_{m+1}(\tilde{x}_{m+1}, y)}{M}, & \text{if } h(y) = h(x_{m+1}), \\ 0, & \text{otherwise} \end{cases}$$

$$M = \sum_{\{y| h(y) = h(x_{m+1})\}} p_{m+1}(\tilde{x}_{m+1}, y).$$

Unlike $T$ introduced in Section VI-A, $T^{-1}(x, \tilde{x}_{m+1})$ gives the probability of overhearing $[\tilde{x}_{m+1}, h(x_{m+1})]$ given that $x \in \{y| h(y) = h(x_{m+1})\}$ is the original codeword sent by $v_{m+1}$ and the channel statistics. Note that $T^{-1}$ is identical to $T$ except for the normalizing factor $M$. A graphical representation of $T^{-1}$ is shown in Figure 6c.

In Figure 6c, $s_1$ and $s_3$ are the inferred candidate linear combinations, i.e., $w(s_1, m) \neq 0$ and $w(s_2, m) \neq 0$; the end node indicates what node $v_1$ has overheard from $v_{m+1}$. Note that although $s_1$ is one of the inferred linear combinations, $s_1$ is not connected to the end node. This is because $h(s_1) \neq h(x_{m+1})$. On the other hand, $h(s_2) = h(x_{m+1})$; as a result, $s_2$ is connected to the end node although $w(s_2, m) = 0$.

We define an inferred linear combination $s$ as matched if $w(s, m) > 0$ and $h(s) = h(x_{m+1})$.

Node $v_1$ uses $T^{-1}$ to compute the total probability $p^*$ of hearing $[\tilde{x}_{m+1}, h(x_{m+1})]$ given the inferred linear combinations by computing the following equation:

$$p^* = \sum_{s \in S} w(s, m) \cdot T^{-1}(s, \tilde{x}_{m+1}).$$

Probability $p^*$ is the probability of overhearing $\tilde{x}_{m+1}$ given the channel statistics; thus, measures the likelihood that $v_{m+1}$ is consistent with the information gathered by $v_1$. Node $v_1$ can use $p^*$ to make a decision on $v_{m+1}$’s behavior. For example, $v_1$ can use a threshold decision rule to decide whether $v_{m+1}$ is misbehaving or not: $v_1$ claims that $v_{m+1}$ is malicious if $p^* \leq t$ where $t$ is a threshold value determined by the given channel statistics; otherwise, $v_1$ claims $v_{m+1}$ is well-behaving.

Depending on the decision policy used, we can use the hypothesis testing framework to analyze the probability of false positive and false negative. Section V provides such analysis for the simple two-hop network with a simple decision policy – if the inferred linear combination and the message overheard from the next hop node is non-empty, we declare the node well-behaving. However, the main purpose of this paper is to propose a method in which we can compute $p^*$, which can be used to establish trust within a network. We note that it would be worthwhile to look into specific decision policies and their performance (i.e., false positive/negative probabilities) as in [10].

VII. ANALYSIS FOR TWO-HOP NETWORK

We provide an analysis for the performance of algebraic watchdog for two-hop network.

Theorem 7.1: Consider a two-hop network as shown in Figure 3. Consider $v_j$, $j \in [1, m]$. Then, the number of matched codewords is:

$$2^n \left[ \sum_{i \geq j, i \in [1, m+1]} (H(p_{i+1}) - H(p_i)) \right] - m \cdot \delta.$$  

Proof: Without loss of generality, we consider $v_1$. The proof uses on concepts and techniques developed for list-decoding [22]. We first consider the overhearing of $v_k$’s transmission, $k \in [2, m]$. Node $v_1$ overhears $\tilde{x}_k$ from $v_k$. The noise introduced by the overhearing channel is characterized by $BSC(p_{k1})$; thus, $E[\Delta(x_k, \tilde{x}_k)] = np_{k1}$. Now, we consider the number of codewords that are within $B(\tilde{x}_k, np_{k1})$, the Hamming ball of radius $np_{k1}$ centered at $\tilde{x}_k$ is $|B(\tilde{x}_k, np_{k1})| = 2^n(H(p_{k1}) - H(\frac{p_{k1}}{2}))$. Node $v_1$ overhears the hash $h(\tilde{x}_k)$; thus, the number of codewords that $v_1$ considers is reduced to $2^n(H(p_{k1}) - H(\frac{p_{k1}}{2})) - \delta$. Using this information, $v_1$ computes the set of inferred linear combinations, i.e., $s$ where $w(s, m) > 0$. Note that $v_1$ knows precisely the values of $x_1$. Therefore, the number of inferred linear combinations is upper bounded by:

$$\prod_{k \in [2, m]} \left( 2^n(H(p_{k1}) - H(\frac{p_{k1}}{2})) - \delta \right)$$

Due to the finite field operations, these inferred linear combinations are randomly distributed over the space $\{0, 1\}^n$.

Now, we consider the overheard information, $\tilde{x}_{m+1}$ from the downstream node $v_{m+1}$. By similar analysis as above, we can derive that there are $2^{n(H(p_{m+1}) - H(\frac{p_{m+1}}{2})) - \delta}$ codewords in the hamming ball $B(\tilde{x}_{m+1}, np_{m+1, 1})$ with hash
value $h(x_{m+1})$. Thus, the probability that a randomly chosen codeword in the space of $\{0, 1\}^n$ is in $B(x_{m+1}, n_{m+1, 1}) \cap \{x|h(x) = h(x_{m+1})\}$ is given by
\[
\frac{2^n(H(p_{m+1, 1}) - H(\frac{d_{m+1}}{2^{m+1}}) - \delta)}{2^n}.
\]

Then, the expected number of matched codewords is the product of Equations (2) and (3).

If we assume that the hash is of length $\delta = cn$, then the statement in Theorem 7.1 is equal to:
\[
2^n \left[\sum_{x:j, i \in [1, m+1]} H(p_{ij}) - \left(\sum_{x:j, i \in [1, m+1]} H(\frac{d_{m+1}}{2^{m+1}}) + m\delta\right)\right].
\]

This highlights the tradeoff between the quality of overhearing channel and the redundancy (introduced by $C_i$’s and the hash $h$). If enough redundancy is introduced, then $C_i$ and $h$ together form an error-correcting code for the overhearing channels; thus, allows exact decoding to a single matched codeword.

The analysis also shows how adversarial errors can be interpreted. Assume that $v_{m+1}$ wants to inject errors at rate $p_{adv}$. Then, node $v_1$, although has an overhearing $BSC(p_{m+1, 1})$, effectively experiences an error rate of $p_{adv} + p_{m+1, 1} - p_{m+1, 1}$. Note that this does not change the set of the inferred linear combinations; but it affects $x_{m+1}$. Thus, overall, adversarial errors affect the set of matched codewords and the distribution of $p'$. As we shall see in Section IX, the difference in distribution of $p'$ between a well-behaving relay and adversarial relay can be used to detect malicious behavior.

VIII. PROTOCOL FOR ALGEBRAIC WATCHDOG

We use the two-hop algebraic watchdog from Section VI in a hop-by-hop manner to ensure a globally secure network. In Algorithm 1, we present a distributed algorithm for nodes to secure the their local neighborhood. Each node $v$ transmits/receives data as scheduled; however, node $v$ randomly chooses to check its neighborhood, at which point node $v$ listens to neighbors transmissions to perform the two-hop algebraic watchdog from Section VI.

### Algorithm 1: Distributed algebraic watchdog at $v$.

```plaintext
foreach node $v$ do

  According to the schedule, transmit and receive data; 
  if $v$ decides to check its neighborhood then

    Listen to neighbors’ transmissions:

    foreach downstream neighbor $v'$ do

      Perform Two-hop Algebraic Watchdog on $v'$;

    end

  end

end
```

Section VII noted that presence of adversarial error (at a rate above the channel noise) can be detected by a change in distribution of $p'$. Corollary 8.1 does not make any assumptions on whether packets $p_i$’s are valid or not. Instead, the claim states that $v_{m+1}$ transmits a valid packet given the packets $p_i$ it has received.

**Corollary 8.2:** Node $v$ can inject errors beyond the channel noise only if either of the two conditions are satisfied:

1. All its parent nodes $P(v) = \{u|(u, v) \in E_1\}$ are colluding Byzantine nodes;
2. All its downstream nodes, i.e. receivers of the transmission $p_i$, are colluding Byzantine nodes.

**Remark:** In Case 1), $v$ is not responsible to any well-behaving nodes. Node $v$ can transmit any packet without the risk of being detected by any well-behaving parent node. However, then, the min-cut to $v$ is dominated by adversaries, and the information flow through $v$ is completely compromised – regardless of whether $v$ is malicious or not.

In Case 2), $v$ can generate any hash value since its downstream nodes are colluding adversaries. Thus, it is not liable to transmit a consistent hash, which is necessary for $v$’s parent nodes to monitor $v$’s behavior. However, note that $v$ is not responsible in delivering any data to a well-behaving node. Even if $v$ were well-behaving, it cannot reach any well-behaving node without going through a malicious node in the next hop. Thus, the information flow through $v$ is again completely compromised.

Therefore, Corollary 8.2 shows that the algebraic watchdog can aid in ensuring correct delivery of data when the following assumption holds: for every intermediate node $v$ in the path between source to destination, $v$ has at least one well-behaving parent and at least one well-behaving child – i.e. there exists at least a path of well-behaving nodes. This is not a trivial result as we are not only considering a single-path network, but also multi-hop, multi-path network.

IX. SIMULATIONS

We present MATLAB simulation results that show the difference in distribution of $p'$ between the well-behaving and adversarial relay. We consider a setup in Figure 3. We set all $p_{i1}$, $i \in [2, m]$ to be equal, and we denote this probability as $p_s = p_{i1}$ for all $i$. We denote $p_{adv}$ to be the probability at which the adversary injects error; thus, the effective error that $v_1$ observes from an adversarial relay is combined effect of $p_{m+1, 1}$ and $p_{adv}$. The hash function $h(x) = ax + b \mod 2^{\delta}$ is randomly chosen over $a, b \in F_{2^\delta}$.

We set $n = 10$; thus, the coding field size is $2^{10}$. A typical packet can have a few hundreds to tens of thousand bits. Thus, a network coded packet with $n = 10$ could have a few tens to a few thousands of symbols over which to perform algebraic watchdog. It may be desirable to randomize which symbols a node performs algebraic watchdog on, or when to perform algebraic watchdog. This choice depends not only on the security requirement, but also on the computational and energy budget of the node.

For each set of parameters, we randomly generate symbols from $F_{2^{10}}$ ($n = 10$ bits) and run algebraic watchdog.
each symbol, under a non-adversarial setting, we assume that only channel randomly injects bit errors to the symbol; under adversarial setting, both the channel and the adversary randomly inject bit errors to the symbol. For each set of parameters, we run the algebraic watchdog 1000 times. Thus, this is equivalent to running the algebraic watchdog on a moderately-sized packet (10,000 bits) or over several smaller packets, which are network coded using field size of $\mathbb{F}_{2^{20}}$.

For simplicity, nodes in the simulation do not use error-correcting codes; thus, $d_i = 0$ for all $i$. This limits the power of the algebraic watchdog; thus, the results shown can be further improved by using error correcting codes $C_i$.

We denote $p_{adv}$ and $p_{relay}$ as the value of $p^*$ when the relay is adversarial and is well-behaving, respectively. We denote $\text{var}_{adv}$ and $\text{var}_{relay}$ to be the variance of $p_{adv}$ and $p_{relay}$. We shall show results that show the difference in distribution of $p_{adv}$ and $p_{relay}$ from $v_1$’s perspective. Note that this illustrates that only one good parent node, i.e. $v_1$ in our simulations, is sufficient to notice the difference in distribution of $p_{adv}$ and $p_{relay}$. Thus, confirming our analysis in Section VIII. With more parent nodes performing the check independently, we can improve the probability of detection.

Our simulation results coincide with our analysis and intuition. Figure 7 shows that adversarial above the channel noise can be detected. First of all, for all values of $p_{adv} > 0$, $p_{adv} < p_{relay}$; thus, showing that adversarial errors can be detected. Furthermore, the larger the adversarial error injection rate, the bigger the difference in the distributions of $p_{adv}$ and $p_{relay}$. When adversarial error rate is small, then the effective error $v_1$ sees in the packet can easily be construed as that of the channel noise, and thus, if appropriate channel error correcting code is used, can be corrected by the downstream node. As a result, the values/distributions of $p_{relay}$ and $p_{adv}$ are similar. However, as the adversarial error rate increases, there is a divergence between the two distributions. Note that the difference in the distributions of $p_{relay}$ and $p_{adv}$ is not only in the average value. The variance $\text{var}_{relay}$ is relatively constant throughout (approximately 0.18 throughout). On the other hand, $\text{var}_{adv}$ generally decreases with increase in $p_{adv}$. For small $p_{adv}$, the variance $\text{var}_{adv}$ is approximately 0.18; while for large $p_{adv}$, the variance $\text{var}_{adv}$ is approximately 0.08. This trend intuitively shows that, with increase in $p_{adv}$, not only do we detect that the adversarial relay more often (since the average value of $p_{adv}$ decreases), but we are more confident of the decision.

Figure 8 shows the effect of the size of the hash. With increase in redundancy (by using hash functions of length $\delta$), $v_1$ can detect malicious behavior better. This is true regardless of whether the relay is well-behaving or not. Node $v_1$’s ability to judge its downstream node increases with $\delta$. Thus, $p_{adv}$ for $\delta$ is generally higher than that of $\delta^*$ where $\delta > \delta^*$. This holds for $p_{relay}$ as well. However, for any fixed $\delta$, node $v_1$ can see a distinction between $p_{relay}$ and $p_{adv}$ as shown in Figure 8. A similar trend to that of Figure 7 can be seen for the distributions of $p_{adv}$ and $p_{relay}$ for each value of $\delta$. The interesting result is that even for $\delta = 0$, i.e. we include no redundancy or hash, node $v_1$ is able to distinguish an adversarial relay from a well-behaving relay.

![Figure 7: The average value of $p^*$ with well-behaving relay (denoted $p_{relay}$) and adversarial relay (denoted $p_{adv}$) over 1000 random iterations of algebraic watchdog. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, $\delta = 2$, and $p_s = p_{m+1,1} = 10\%$. We vary $p_{adv}$, the adversary’s error injection rate.](image1)

![Figure 8: The average value of $p^*$ with well-behaving relay (denoted $p_{relay}$) and adversarial relay (denoted $p_{adv}$) over 1000 random iterations of algebraic watchdog. We vary the value of $\delta$, the length of the hash function used, and $p_{adv}$, the adversary’s error injection rate. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = 10\%$. Results in Figure 9 confirms our intuition that the better $v_1$’s ability to collect information from $v_i$’s, in $e1$, the better its detection ability. If node $v_1$ is able to infer better or overhear $x_i$ with little or no errors, the better its inference on what the relay node should be transmitting. Thus, as overhearing channel progressively worsens ($p_s$ increases), $v_1$’s ability to detect malicious behavior deteriorates; thus, unable to distinguish between a malicious and a well-behaving relay.

Finally, we note the effect of $m$, the number of nodes in the network, in Figure 10. Node $v_1$’s ability to check $v_{m+1}$ is reduced with $m$. When $m$ increases, the number of messages $v_1$ has to infer increases, which increases the uncertainty within the system. However, it is important to note that as $m$ increases, there are more nodes $v_i$’s, $i \in [1, m]$ that can independently perform checks on $v_{m+1}$. This effect is not captured by the results shown in Figure 10.

![Figure 9: The average value of $p^*$ with well-behaving relay (denoted $p_{relay}$) and adversarial relay (denoted $p_{adv}$) over 1000 random iterations of algebraic watchdog. We vary the value of $\delta$, the length of the hash function used, and $p_{adv}$, the adversary’s error injection rate. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = 10\%$.](image2)

![Figure 10: The average value of $p^*$ with well-behaving relay (denoted $p_{relay}$) and adversarial relay (denoted $p_{adv}$) over 1000 random iterations of algebraic watchdog. We vary the value of $\delta$, the length of the hash function used, and $p_{adv}$, the adversary’s error injection rate. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = 10\%$. Results in Figure 9 confirms our intuition that the better $v_1$’s ability to collect information from $v_i$’s, in $e1$, the better its detection ability. If node $v_1$ is able to infer better or overhear $x_i$ with little or no errors, the better its inference on what the relay node should be transmitting. Thus, as overhearing channel progressively worsens ($p_s$ increases), $v_1$’s ability to detect malicious behavior deteriorates; thus, unable to distinguish between a malicious and a well-behaving relay.](image3)

![Figure 11: The average value of $p^*$ with well-behaving relay (denoted $p_{relay}$) and adversarial relay (denoted $p_{adv}$) over 1000 random iterations of algebraic watchdog. We vary the value of $\delta$, the length of the hash function used, and $p_{adv}$, the adversary’s error injection rate. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, and $p_s = p_{m+1,1} = 10\%$. Results in Figure 9 confirms our intuition that the better $v_1$’s ability to collect information from $v_i$’s, in $e1$, the better its detection ability. If node $v_1$ is able to infer better or overhear $x_i$ with little or no errors, the better its inference on what the relay node should be transmitting. Thus, as overhearing channel progressively worsens ($p_s$ increases), $v_1$’s ability to detect malicious behavior deteriorates; thus, unable to distinguish between a malicious and a well-behaving relay.](image4)

X. Conclusions and Future Work

We proposed the algebraic watchdog, in which nodes can verify their neighbors probabilistically and police them locally
value of $m$

We set $m = 3$, $n = 10$, and $p_{adv} = 10\%$.

Fig. 10: The average value of $p^*$ with well-behaving relay (denoted $p^*_{relay}$) and adversarial relay (denoted $p^*_{adv}$) over 1000 random iterations of algebraic watchdog. We vary the value of $p_s = p_{m+1,1}$, the quality of overhearing channels. The error bars represent the variance, $\text{var}_{relay}$ and $\text{var}_{adv}$. We set $m = 3$, $n = 10$, and $p_{adv} = 10\%$.

by the means of overheard messages in a coded network. Using the algebraic watchdog scheme, nodes can compute a probability of consistency, $p^*$, which can be used to detect malicious behavior. Once a node has been identified as malicious, these nodes can either be punished, eliminated, or excluded from the network by using reputation based schemes such as [9][23].

We first presented a graphical model and an analysis of the algebraic watchdog for two-hop networks. We then extended the algebraic watchdog to multi-hop, multi-source networks. We provided a trellis-like graphical model for the detection of the algebraic watchdog to multi-hop, multi-source networks. We then extended the algebraic watchdog scheme, nodes can compute a probability of consistency, $p^*$, which can be used to detect malicious behavior. Once a node has been identified as malicious, these nodes can either be punished, eliminated, or excluded from the network by using reputation based schemes such as [9][23].

Our ultimate goal is to design a network in which the participants check their neighborhood locally to enable a secure global network - i.e. a self-checking network. Possible future work includes developing inference methods and approximation algorithms to decide efficiently aggregate local trust information to a global trust state.

REFERENCES


