On Locating Byzantine Attackers

Mahdi Jafari Siavoshani, Christina Fragouli, Suhas Diggavi
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: {mahdi.jafarisiavoshani,christina.fragouli,suhas.diggavi}@epfl.ch

Abstract — **We examine networks that employ network coding and are subject to Byzantine attacks. We assume that an appropriate network error correcting scheme is employed that is able to correct (up to a certain number of) Byzantine errors. Given this setup, we formulate the problem of locating these malicious nodes that insert errors. We utilize the subspace properties of (randomized) network coding to develop algorithms to locate the Byzantine attackers.**

I. Introduction

Over the past few years, network error correcting codes, that are capable of correcting errors inserted in the network, have been developed [3, 6, 7, 8, 9, 10]. These schemes are therefore capable of delivering information despite the presence of Byzantine attacks in the network, as long as the number of such attacks is limited. There have been codes proposed for the detection and correction of Byzantine errors with provable rate bounds for given error correction capabilities, along with a number of practical approaches (see for example [6] and references therein). These network error correcting schemes are designed to work without knowledge of the network topology.

In this paper we ask a new question: can we use properties of network error correcting codes as well as knowledge of the network topology to locate the nodes that insert the errors? This is motivated by recent work [4, 5], where it was shown that using subspace properties of random network coding, one can in many cases infer the network topology. Therefore, in this paper, we ask the question of whether we can locate the Byzantine attackers, using properties of the network code, and the knowledge of the topology.

Hence, in some sense, we are here interested in the dual aspect of the network error correction problem: once a Byzantine attack is detected, we would like to *identify* which of the network node(s) has (have) launched the attack. This would allow us to isolate the malicious attacker from the network, and thus avoid the computational complexity of correcting errors as well as prevent a possible escalation of the attack.

In a network coded system, the adverserial nodes in the network disrupt the normal operation of the information flow by inserting erroneous packets into the network. This can be done by inserting spurious data packets into their outgoing edges. One way in which these erroneous packets can be prevented from disrupting information flow is by reducing the transmission rate to below the min-cut of the network, and using the redundancy to protect against errors. One such technique, using subspaces to code information was proposed in [3]. In this approach, the source sends a basis of the subspace corresponding to the message. In the absence of errors, the linear operations of the intermediate nodes do not alter the sent subspace, and hence the receiver decodes the message by collecting the basis of the transmitted subspace. A malicious attacker inserts vectors that do not belong in the transmitted subspace. Therefore, if the message codebook uses subspaces that are "far enough" apart (according to an appropriately defined distance measure), then one can correct these errors [3]. Note that in this technique, we do not need any knowledge of the network topology for the error correction mechanism. All that is needed is that the intermediate nodes do not alter the transmitted subspace (which can be done if they do linear operations).

As mentioned earlier, our approach to locating adversaries uses the framework developed in [4], where it was shown that under randomized network coding, the subspaces at the nodes of the network give information about the topology. Therefore, the basic premise in this paper is to use the structure of the erroneous subspace inserted by the adversary to reveal information about its location, when we already know the network topology.

Other than the new problem formulation, our contributions in this paper are the following. For the case of the single adverserial node, we present simple algorithms that allow to identify the adversary within an uncertainty of at most two nodes. When there are multiple adversaries, we discuss a number of algorithms, and their respective capabilities to identify adversaries. Though these are preliminary ideas, we believe this is a promising line of research.

Note that for the development of the ideas in this paper, we will assume that subspace network error correcting codes are used. However, the approach we will describe could also apply to the more traditional network error correction, where a coding vector is appended to each transmitted packet, and intermediate nodes perform randomized network coding. We note that for such network error correcting the intermediate nodes in the network (including the malicious ones) do not need to know the position of the coding vectors within the packet, since they simply need to apply the same operations to all symbols in the packet. Thus corrupting a packet will corrupt the coding vector as well. We can then apply our approach to the subspaces spanned by coding vectors.

The paper is organized as follows. We formally state our problem and introduce notation in Section II. We investigate basic subspace properties in Section III. We examine the case of a single adversary in Section IV and discuss how our observations extend in the case of multiple adversaries in Section V. In Section VI, we end with a discussion of how we plan to extend the preliminary ideas presented in this paper.

II. Problem Formulation

Consider a network represented as a directed acyclic graph $G = (V, E)$. We have a source, sending information to N receivers, and one (or more) Byzantine adversaries, located at intermediate nodes of the network. We assume complete knowledge of the network topology, and consider the source and the receivers to be trustworthy (authenticated) nodes, that are guaranteed not to be adversaries.

We can restrict the Byzantine attack in several ways, depending on the edges where the attack is launched, the number of corrupted vectors inserted, and the vertices (network nodes) that the adversary has access to. In this paper we will distinguish between the cases where

I. there is a single Byzantine attacker located in a vertex of the network, and

II. there are multiple independent attackers, located on different vertices, that act without coordinating with each other.

Moreover, we will consider the cases where, an attacker located on a single vertex inserts corrupted packets on

(a) exactly one of the vertex outgoing edges,

(b) all the outgoing edges, or

(c) a subset of the outgoing edges.

We are interested in understanding under what conditions we can uniquely identify the attacker's location (or, up to what uncertainty we can identify the attacker), under the above scenarios.

II-A. Network Operation

In this section, we set up the notation used for the subspace-based network coding scheme studied in this paper.

The source S sends n vectors, that span a n-dimensional subspace Π_S of space W, where W is defined over a finite field $\mathbb{F}_q$, with $q \gg 1$. In particular, Π_S belong to a codebook $\mathcal{C}$, $\Pi_S \in \mathcal{C}$, which is designed to correct network errors [3].

Each intermediate network node i performs randomized network coding, that is, it sends random (uniform) linear combinations over $\mathbb{F}_q$ of its collected packets to its neighbors. We say that node $i \in V$ at time t observes a subspace $\Pi_i \subseteq \mathbb{F}_q^n$, if $\Pi_i(t)$ is the space spanned by the received vectors at node i up to time t (for simplicity of notation we omit the time index t).

Let $\Pi_i \cup \Pi_j$ refer to the common span and $\Pi_i \cap \Pi_j$ to the intersection of subspaces Π_i and Π_j.

If v has $\mathrm{In}(v)$ incoming edges (parent nodes), then since we use randomized network coding,

$$\Pi_v = \cup_{i=1}^{\mathrm{In}(v)} \Pi_v^{(i)},$$

where $\Pi_v^{(i)}$ is the subspace v has received up to time t from the parent node i.

In the absence of any erasures or adversaries in the network each receiver R collects the exact space Π_S. Assume that there is an adversary who attacks one of the nodes in the network by combining a t-dimensional subspace Π_E with its incoming space and sending the resulting vectors to its children. In addition we will assume that $t < n$. Then receiver R collects $n \leq m \leq n+t$ innovative vectors that span a subspace Π_R. We may write

$$\Pi_R = \mathcal{H}_m(\Pi_S \cup \Pi_E),$$

where $\mathcal{H}_m$ is an operator that acts on a space and selects an m-dimensional subspace of it. The operator $\mathcal{H}_m$ depends on the topology of the network and the code that is used in the network. If nodes in the network perform random network coding, $\mathcal{H}_m$ has some random structure.

We assume that the receiver is able to at least detect that a Byzantine attack is under way. Moreover, we assume that the receiver is able to obtain the subspace Π_S that the source has sent. This might be, either because the receiver has correctly decoded the sent message, or, because after detecting the presence of an attack, has requested the source subspace through a secure channel from the source node.

III. Basic properties

We here investigate how the insertion of the error subspace Π_E affects the subspaces that the intermediate network nodes observe.

We can write the received subspace at arbitrary node i the same way as we did for the receiver R,

$$\Pi_i = \mathcal{H}_m^i(\Pi_S \cup \Pi_E).$$

Then it is possible to expand Π_i as follows,

$$\begin{aligned} \Pi_i &= \hat{\Pi}_{Si} \oplus \underbrace{(\hat{\Pi}_{Ei} \oplus \hat{\Pi}_i)}_{\Pi_i^\star} \\ &= \hat{\Pi}_{Si} \oplus \Pi_i^\star, \end{aligned} \tag{1}$$

where $\oplus$ denotes the direct sum of spaces, $\hat{\Pi}_{Si} \triangleq \Pi_i \cap \Pi_S \subseteq \Pi_S$, $\hat{\Pi}_{Ei} \triangleq \Pi_i \cap \Pi_E \subseteq \Pi_E$ and $\hat{\Pi}_i$ is the rest of Π_i which cannot be represented as just part of Π_S or Π_E. We underline that in general $\Pi_i^\star \not\subseteq \Pi_E$.

If the operator $\mathcal{H}_m^i$ selects Π_i uniformly at random, w.h.p[1] we will have

$$\begin{aligned} \dim(\hat{\Pi}_{Si}) &= m-t, \ \dim(\hat{\Pi}_{Ei}) = m-n, \\ \dim(\hat{\Pi}_i) &= t-(m-n). \end{aligned}$$

[1]With high probability.

The above results are a direct consequence of Lemma 1 in [4], and provide a lower bound for these dimensions in the general case. Using again Lemma 1 in [4], for two arbitrary nodes i and j, and without any further assumptions, we have the following inequalities,

$$\begin{aligned}\dim(\Pi_i \cap \Pi_j) &\geq 2m-(n+t)\\ &= (m-t)+(m-n),\end{aligned}$$

and,

$$\begin{aligned}\dim(\hat{\Pi}_{Si} \cap \hat{\Pi}_{Sj}) &\geq 2(m-t)-n\\ &= 2(m-n)+(n-2t).\end{aligned}$$

Lemma 1. *In the above scenario, two arbitrary node i and j in the network should gather $m > n+t/2$ innovative vectors to have* $\dim(\Pi_i^\star \cap \Pi_j^\star) > 0$.

Proof. From Lemma 1 in [4] we can write

$$\dim(\hat{\Pi}_{Ei} \cap \hat{\Pi}_{Ej}) \geq 2(m-n)-t.$$

So if we have $n+t/2 < m$, the two subspaces $\Pi_i^\star = \hat{\Pi}_{Ei} \oplus \hat{\Pi}_i$ and $\Pi_j^\star = \hat{\Pi}_{Ej} \oplus \hat{\Pi}_j$ have nonzero intersection. □

Thus we conclude that, if an adversary introduces Π_E, and intermediate nodes perform randomized network coding, it is not necessary that the nodes collecting corrupted information will collect a subspace of Π_E. Additionally, two nodes that collect corrupted information, may only have as common information a subspace of Π_S, unless they collect a sufficient number of innovative packets.

IV. The case of a single adversary

In this section we focus on the case where we want to locate a Byzantine adversary controlling a *single* vertex of the network graph. We will develop methods which are suitable for the cases where the adversary corrupts one edge, all edges, or a subset of its out-going edges (cases $(a), (b), (c)$ respectively of Section II).

In Section IV-A we illustrate the limitation of using *only* the information the receivers have observed along with the knowledge of the topology, to locate the adversary. This motivates requiring additional information from the intermediate nodes related to the subspaces observed by them. In Section IV-B, we show that such additional information allows us to localize the adversary either uniquely or within an ambiguity of at most two nodes.

IV-A. Identification using only topological information

In order to illustrate the ideas, we will first examine the case where the corrupted packets are inserted on a single edge of the network, say edge e_A. This corresponds to case (a) in Section II. The extension to cases (b) and (c) is straightforward.

Since each receiver R knows the subspaces $\{\Pi_R^{(i)}\}$ it has received from its $\text{In}(R)$ parents, it knows whether what it received is corrupted or not (a subspace of Π_S or not). Using this, we can infer some information regarding topological properties that the edge e_A should satisfy. In particular:

- If R receives corrupted vectors from an incoming edge e then there exists at least one path that connects e_A to e. Let P_e denote the set of paths[2] starting from the source and ending at edge e. Then e_A is part of at least one path in P_e.
- Conversely, if a receiver R does not receive corrupted packets from an incoming edge e, then e_A does not form part of any path in P_e. That is, there does not exist a path that connects e_A to e.

Then, if $\mathcal{E}_1$ is the set of incoming edges to receivers that bring corrupted packets, while $\mathcal{E}_2$ the set of incoming edges to receivers that only bring source information, the edge e_A belongs in the set of edges $\mathcal{E}_A$, with

$$\mathcal{E}_A \triangleq \{\bigcap_{e\in\mathcal{E}_1} P_e - \bigcup_{e\in\mathcal{E}_2} P_e\},$$

where $\mathcal{A}-\mathcal{B}$ denotes the set where elements in $\mathcal{B}$ are removed from the set $\mathcal{A}$. The following example illustrates this approach.

Example 1. Consider the network in Fig. 1, and assume that R_1 receives corrupted packets from edge DR_1 and uncorrupted packets from AR_1, while R_2 receives only uncorrupted packets. Then $\mathcal{E}_A = \{DR_1\}$ and the

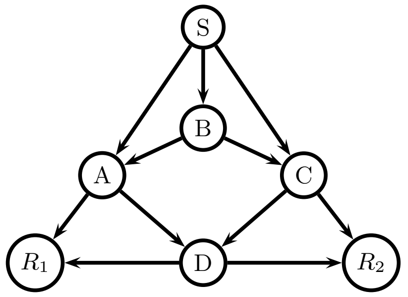

Fig. 1: The source S distributes packets to receivers R_1 and R_2.

attacker is located on node D. □

In Example 1, we were able to exactly identify the location of the adversary, because the set $\mathcal{E}_A$ contained a single edge, and node R_1 is trustworthy. It is easy to find network configurations where $\mathcal{E}_A$ contains multiple edges, or in fact all the network edges, and thus we can no longer identify the attacker. The following example illustrates one such case.

[2]In the following we are going to equivalently think of P_e as the set of all edges that take part in these paths.

Example 2. Consider the line network shown in Fig. 2. Suppose the attacker is node A. If the receiver R sees a corrupted packet, then using just the topology, the attacker could be *any* of the other nodes in the line network. This illustrates that just the topology and receiver information could lead to large ambiguity in the location of the attacker. □

Therefore, Example 2 motivates the ideas examined in Section IV-B that obtain additional information and utilize the structural properties of the subspaces observed.

IV-B. Identification using information from all network nodes

We will next discuss algorithms where a central authority, which we will call *controller*, requests from all nodes in the network to report some additional information, related to the subspaces they have received from their parents. The adversary could send inaccurate information to the controller, but the other nodes report the information accurately. Our task is to design the question to the nodes such that we can locate the adversary, despite its possible misdirection.

The controller may ask the nodes of the following types of information, listed in decreasing order of complexity:

Information 1: Each node v sends all subspaces $\Pi_v^{(i)}$ it has received from its parents, where $\Pi_v = \cup_{i=1}^{\text{In}(v)} \Pi_v^{(i)}$.

Information 2: Each node v sends a randomly chosen vector from each of the received subspaces $\Pi_v^{(i)}$ ($\text{In}(v)$ vectors in total).

Information 3: Each node v sends one randomly chosen vector from its subspace Π_v.

Information 2 and 3 is motivated by the following well-known observation [2, 4]: let Π_1 and Π_2 be two subspaces of $\mathbb{F}_q^n$, and assume that we randomly select a vector y from Π_1. Then, for $q \gg 1$, $y \in \Pi_2$ if and only if $\Pi_1 \subseteq \Pi_2$. Thus, a randomly selected vector from Π_v (Information 3) allows to check whether $\Pi_v \subseteq \Pi_S$ or not.

In fact, we will show in this section that for a single adversary it is sufficient to use[3] Information 2, and classify the edges of the network by simply testing whether the information flowing through each edge is a subspace of Π_S or not (i.e., is corrupted or not).

IV-B.1. The line network

To build the intuition behind our approach, we first examine the case of the line network, depicted in Fig. 2, that corresponds to a single path connecting the source to the receiver. We saw in Example 2, that just topological information was insufficient to reduce ambiguity of the attacker's location. For the line network, cases $(a), (b), (c)$ in Section II coincide.

Assume now that the controller asks for Information 1, *i.e.,* all nodes to report their collected subspaces to the controller. The adversary has two courses of action: it can either correctly report the subspace it received from its parent node, or lie, and claim that it received a corrupted subspace from its parent. We do not know which of the two approaches the adversary has selected. However, in both cases, we can divide the network edges into two sets, the set of edges through which is reported to flow correct information, and the set of edges through which is reported to flow corrupted information.

[3]Using Information 2 or 3 these statements are made w.h.p, *i.e.,* the probability goes to one as field size $q \to \infty$.

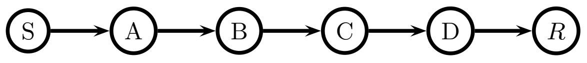

Fig. 2: The source S sends information to receiver R over a line network.

For example, if the adversary is node C in Fig. 2, the sets corresponding to the possible adversary actions are depicted in Fig. 3(i) and 3(ii) respectively. It is clear that the adversary is one of the two nodes connecting the edge on the border of the sets, that is, in the set of vertices $\{C,\ D\}$ for the case in Fig. 3(i), and in the set $\{B,\ C\}$ for the case in Fig. 3(ii). In particular, the adversary is one of the two adjoining nodes, of the first ancestral reported corrupted edge.

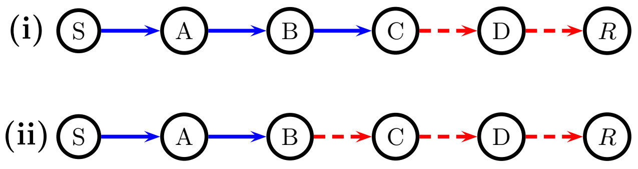

Fig. 3: Case (i): edge partition if the adversary node C reports the truth, and Case (ii): edge partition if the adversary node C lies. Edges bringing corrupted information are depicted as dashed.

Note that we can divide the edges in these two sets simply using Information 2 or 3 (these coincide for the line network) since it is sufficient to check whether each edge is corrupted.

Also note that networks where all nodes have out-degree one and arbitrary in-degree, can be treated in exactly the same way as the line network. Indeed, the identification in such networks can be, without loss of generality, decomposed in identification along single paths.

IV-B.2. General networks

Consider a directed acyclic graph, and assume that we impose a partial order on the edges of the graph, such that $e_1 > e_2$ if e_1 is an ancestor edge of e_2 (i.e., there exists a path from e_1 to e_2).

Following a similar approach to Section IV-B-1, and using Information 2, we divide the edges of the network into two sets: the set of edges E_C through which are reported to flow corrupted subspaces, and the remaining edges E_S through which the source information flows. Note that all the outgoing edges from the source belong in E_S, while the receiver observes at least one edge in E_C.

Consider case (a), where the adversary corrupts a single edge. Clearly, since there exists a single adversary, E_C forms a connected subgraph. Let e_A be highest order edge in this graph, i.e., $e_A > e$ for all $e \in E_C$. Then, similarly to the case of the line network, the adversary is one of the two nodes adjacent to this edge. We can make similar arguments for cases (b) and (c). This leads to the following lemma.

Lemma 2. *Using Information 1 we can narrow the location of the adversary up to a set of at most two nodes. With Information 2, the same result holds w.h.p.*

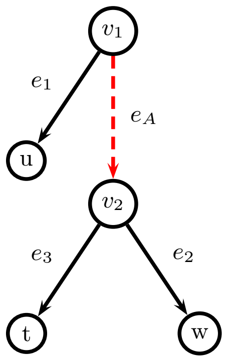

Fig. 4: Edge e_A and neighboring edges-nodes.

In fact, in some cases, we are able to uniquely identify the malicious attacker, as described by the following lemma.

Lemma 3. *If e_A connects vertex v_1 to vertex v_2, and if vertices v_1 and v_2 have outdegree greater or equal to two, then we can uniquely identify*[4] *the attacker in cases (a) and (b) of Section II.*

Proof. The proof is based on the fact that only a single node can lie. Thus we know that every other node, apart from v_1 or v_2, is trustworthy. Let e_1, e_2 and e_3 be outgoing edges of vertex v_1 and v_2 as depicted in Fig. 4.

- Case (a) (the adversary corrupts a single edge): If e_2 and e_3 are corrupted, the adversary can be only located on vertex v_1, while if only one of them is corrupted, the adversary is located on v_2.
- Case (b) (the adversary corrupts all its outgoing edges): If e_1 is corrupted, the adversary is located on vertex v_1, and otherwise on v_2.

□

V. The case of multiple adversaries

In the case of a single adversary, it was sufficient to divide the set of edges into two sets, E_S and E_C, as described in the previous section. In the presence of multiple adversaries, this may no longer be sufficient. An additional dimension is that realistically, we may not know the exact number of adversaries present. In the following, we discuss a number of algorithms, that offer more or less identifiability guarantees.

V-A. Identification using topological information

The approach in Section IV-A can be directly extended in the case of multiple adversaries, but again, offers no identifiability guarantees.

Example 3. Consider again the network in Fig. 1, and assume that R_1 receives corrupted packets only from edge DR_1 while R_2 receives corrupted packets only from edge DR_2. Then $\mathcal{E}_A = \{AD, CD, DR_1, DR_2\}$ and (depending on our assumptions) we may have,

– a single adversary located on node D,
– two adversaries, located on nodes A and C,
– two adversaries, located on nodes A and D, or nodes C and D, or
– three adversaries, located on nodes A, C, and D.

V-B. Identification using Information 2

Similarly to Section IV-B, we can divide the set of edges into two sets E_S and E_C, depending on whether the information flowing through each edge belongs in Π_S or not. Depending on the network topology, we may be able to uniquely identify the location of the attackers. However, this approach, although it guarantees to find at least one of the attackers (within an uncertainty of at most two nodes), does not necessarily find all the attackers, even if we know their exact number.

Intuitively, this is because an attacker might be "in the shadow" of another attacker, meaning that, it may corrupt only already corrupted vectors and thus not incur a detectable effect. More precisely, we say that node B is in the shadow of node A, if there exists a path that connects every incoming edge of B to a corrupted outgoing edge of A. The following example illustrates these points.

Example 4. For the example in Fig. 1, assume that each attacker corrupts all its outgoing edges, and consider the following two situations:

1. Assume that nodes A and C are attackers. If A reports truthfully while C lies we get $E_C = \{AD, AR_1, DR_1, DR_2, BC, CR_2, CD\}$, which allows to identify the attackers.
2. Assume that nodes B and D are attackers. Then we say that node D is in the shadow of node B, as it corrupts only already packets corrupted by B. Indeed, if $E_C = \{SB, BA, BC, AD, AR_1, DR_1, DR_2, BC, CR_2, CD\}$,

[4]Again, this occurs w.h.p. for Information 2 and 3.

knowing that the source is trustworthy, we can infer that node B is an attacker. However, any of the nodes A, C, and D can equally probably be the second attacker. All these nodes are in the shadow of node D. □

V-C. Identification using subset relationships

For each node $i \in V$, let $P(i) = \{u_1, \cdots, u_{p_i}\}$ denote the set of parents of i. We are going to treat $P(i)$ as a super node, and use the notation $\Pi_{P(i)} = \cup_{l=1}^{p_i} \Pi_{u_l}$ for the union of the subspaces of all nodes in $P(i)$. Also recall that $\Pi_j^{(i)}$ denotes the subspace received by node j from node i.

Our last algorithm checks, for every node i, whether

$$\Pi_j^{(i)} \overset{?}{\subseteq} \Pi_{P(i)} \quad \text{for node } j \text{ s.t. } e_{ij} \in E.$$

If this relationship is satisfied, we know that node i is not an adversary. If the relationship is not satisfied, that is $\Pi_j^{(i)} \nsubseteq \Pi_{P(i)}$ for at least one of the children of i, we know that maybe node i is an adversary. For sure we know that

$$\Pi_j^{(A)} \nsubseteq \Pi_{P(A)} \quad \text{for node } j \text{ s.t. } e_{Aj} \in E,$$

but depending on the space that the adversary reports, the above relation may not be satisfied for other nodes.

If the adversary pretends that it is a trustworthy node (just declares its received subspace from the parents) the above relation also fails for the children of A who receive corrupted subspaces. On the other hand, if the adversary tells the truth and declares its whole subspace, we have

$$\Pi_A^{(i)} \nsubseteq \Pi_{P(i)} \quad \text{for all parents } i \text{ of } A.$$

Thus the ambiguity set we have identified includes the adversary and its parents or children depending on the adversary's report.

Now it is possible to use the topological information to make the ambiguity set smaller. We know that the set contains the adversary and either its parent or its children. So the potential adversaries are nodes that are parents or children of all other nodes in the set. If there exists only one such node in the set we can identify the location of the adversary uniquely.

Repeating this procedure for every node in the network, we can identify sets of potential adversaries. This procedure allows to identify adversaries, even if one is in the shadow of another, and even if we do not know their exact number, provided they are "far enough" in the network to be distinguishable. More precisely, we have the following lemma, where distance refers to the length of the shortest path connecting two nodes.

Lemma 4. *If the pairwise distance between adversaries is greater than two, it is possible to find the exact number as well as the location of the attackers (within the described uncertainty of parent-children sets), using the subset method.*

Proof. We know that depending on the adversaries action there exists ambiguity in finding their exact location. In fact in the worst case, the uncertainty is within a set of nodes including the adversary, its parents and its children. So if the distance between adversaries is greater than two, the "uncertainty" sets do not overlap. In this case we can easily distinguish between different adversaries. □

VI. Conclusions and Discussion

Given a network subject to Byzantine attacks, we formulated and examined the problem of locating the adversaries. We showed that in the case of a single adversary, there exist simple algorithms that allow to identify the adversary within an uncertainty of two nodes. For the case of multiple adversaries, we discussed algorithms and conditions under which we can guarantee identifiability.

Our future work includes investigating the best one could do given constraints of resources and power of the adversaries. We are interested in particular in developing decentralized algorithms, where nodes are willing to cooperate exchange messages or certificates with their neighbors and identify the Byzantine attacker in a distributed manner without the use of a centralizer controller.

References

[1] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] T. Ho, R. Kötter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast", *IEEE Trans. Imform. Theory*, vol. 52, pp. 4413-4430, October 2006.

[3] R. Kötter and F. Kschischang, "Coding for errors and erasures in random network coding", *ISIT*, France, June, 2007.

[4] M. Jafarisiavoshani, C. Fragouli, and S. N. Diggavi, "Subspace properties of randomized network coding", *ITW*, pp 17–21, Norway, July 2007.

[5] M. Jafarisiavoshani, C. Fragouli, S. N. Diggavi, and C. Gkantsidis "Bottleneck discovery and overlay management in network coded peer-to-peer systems", *ACM SIGCOMM Workshop on Internet Network Management (INM)*, Japan, August 2007.

[6] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of byzantine adversaries", *Infocom*, pp. 616–624, 2007.

[7] T. Ho, B. Leong, R. Kötter, M. Médard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding", *ISIT*, June 2004.

[8] R. W. Yeung and N. Cai, "Network error correction, i: basic concepts and upper bounds", *Commun. Inf. Syst.*, vol. 6, pp. 19–35, 2006.

[9] N. Cai and R. W. Yeung, "Network error correction, ii: lower bounds", *Commun. Inf. Syst.*, vol. 6, pp. 37–54, 2006.

[10] Z. Zhang, "Network error correction coding in packetized networks", *ITW*, October 2006.