Defense Strategies against Byzantine Attacks in a Consensus-Based Network Intrusion Detection System

Michel Toulouse, Hai Le and Cao Vien Phung
Vietnamese German University, Vietnam
E-mail: michel.toulouse, hai.lh@vgu.edu.vn, caovienphung@gmail.com

Denis Hock
Frankfurt University of Applied Sciences, Germany
E-mail: dehock@fb2.fra-uas.de

Keywords: network security, intrusion detection, consensus algorithms, Byzantine attacks

Received: March 29, 2017

The purpose of a Network Intrusion Detection System (NIDS) is to monitor network traffic such to detect malicious usages of network facilities. NIDSs can also be part of the affected network facilities and be the subject of attacks aiming at degrading their detection capabilities. The present paper investigates such vulnerabilities in a recent consensus-based NIDS proposal [1]. This system uses an average consensus algorithm to share information among the NIDS modules and to develop coordinated responses to network intrusions. It is known however that consensus algorithms are not resilient to compromised nodes sharing falsified information, i.e. they can be the target of Byzantine attacks. Our work proposes two different strategies aiming at identifying compromised NIDS modules sharing falsified information. Also, a simple approach is proposed to isolate compromised modules, returning the NIDS into a non-compromised state. Validations of the defense strategies are provided through several simulations of Distributed Denial of Service attacks using the NSL-KDD data set. The efficiency of the proposed methods at identifying compromised NIDS nodes and maintaining the accuracy of the NIDS is compared. The computational cost for protecting the consensus-based NIDS against Byzantine attacks is evaluated. Finally we analyze the behavior of the consensus-based NIDS once a compromised module has been isolated.

Povzetek: Sistemi za odkrivanje napadov v omrežjih temeljijo na pojavih nenavadnega prometa, vendar so občutljivi na napade. Prispevek opisuje obrambo pred bizantinskimi napadi.

1 Introduction

Network intrusion detection systems are part of a vast array of tools that protect computer infrastructures against malicious activities. The specific task of NIDSs is to monitor computer network infrastructures, seeking to identify malicious intends through the analysis of network traffic. Today's computer networks are quite large, composed of several heterogeneous sub-networks. Consequently, traffic monitoring often needs to be done distributively with sensors and traffic analysis modules placed at different strategic locations, in charge of monitoring and analyzing the traffic of a specific sub-network.

Usually, NIDS monitoring modules are connected by a network, allowing security information collected about a sub-network to be shared with other NIDS modules. Which information is shared and how it is shared often characterize the organization of NIDSs as centralized, hierarchical or distributed [2]. The monitoring modules of centralized and hierarchical NIDS architectures, which can be limited to simply collecting data, send their information up in the hierarchy for further analysis. To the extend that analysis and responses depend on a single or few modules in the NIDS, these systems can be completely incapacitated by attacks that target the more intelligent modules. A common mitigation for these risks is to avoid a single point of failure by using distributed Intrusion Detection Systems [3, 4]. Modules in distributed intrusion detection systems are often full scale sensing and analytical devices. The modules cooperate by sharing information to address attacks from concurrent sources (such as distributed denial of service), to develop network wide coordinated responses to attacks or simply to increase the detection accuracy of each NIDS module. Early distributed systems [5, 6], where also build upon a master–slave architecture and require the data to be sent to a central location for further analysis. Today, using peer-to-peer systems [7, 8, 9, 10], it is possible to recognize attacks by analyzing shared information in a fully distributed manner.

While it is more difficult to completely disable distributed systems compared to centralized ones, modules of a distributed system can still be the target of attacks aiming to disable locally the system or to mask attacks in some sub-networks to other nodes of the distributed NIDS. The present research addresses the vulnerability of a recently proposed fully distributed NIDS [1]. This system uses an average-consensus algorithm for computing network wide security information that can then be used to recognize attacks and activate coordinated responses to malignant activities. However, it is well known that consensus algorithms are not resilient to compromised nodes sharing falsified information, i.e. they can be the target of Byzantine attacks.

Consensus algorithms are based on peer-to-peer communications among neighbor nodes of a computer network (no routing). They are distributed iterative algorithms in which each node of the network repeatedly updates its current value based on its own previous value and the previous values of its neighbors in the network. The objective is to reach a "consensus", i.e. each node computes a same output that depends on *initial values* distributed across the network while using only local updates. Repeating such local computation, and given overlapping neighborhoods, a consensus eventually emerges by diffusion of local updates.

Consensus algorithms have a long history in computer science where they provide solutions to distributed computing problems. For example, consensus algorithms solve the leader election problem, where processes must select one of them to coordinate tasks in a distributed system [11]. Consensus algorithms have also found applications or research interests in physics [12], process control [13], robotic [14], operations research [15], services at IoT edge nodes [16], not to mention its application in the controversial bitcoin currency [17].

Average consensus refers to a particular form of consensus where cooperative nodes compute the average sum of their initial values. Average consensus algorithms also have a wide range of applications, for example we find them recently in wireless network applications such as cooperative spectrum sensing in cognitive radio networks [18], distributed detection in wireless networks [19], sensor networks [20].

Consensus algorithms vulnerabilities to sharing falsified information have been known for a long time. Originally, consensus algorithms solved the problem of reaching agreement assuming a non-faulty non-adversarial computing environment. In reality, links can fail, nodes can stop transmitting data to neighbors (faulty links, nodes) or nodes can transmit incorrect data, possibly falsified by an adversarial actor (Byzantine nodes). The resilience of consensus algorithms has been analyzed in the context of fault-tolerant systems in Lamport, Pease and Shostack [21, 22]. The problem of reaching consensus in faulty and adversarial environments became known as the *Byzantine agreement problem* [22]. The problem asks under which conditions consensus can be reached in the presence of Byzantine faults. In [21], it is proved that resilient consensus algorithms cannot be designed in a fully connected network (complete graph) of n processors if the number m of Byzantine nodes is $3m + 1 \geq n$. The Byzantine agreement problem in [22] refers specifically to attacks in which Byzantine nodes modify the initial values for which consensus is computed (data falsification attacks). Since then, the Byzantine agreement problem has been adapted to consider new failure conditions, i.e. different attack models, as well as quite diverse network settings.

Research studies aiming at detecting Byzantine nodes are of a particular relevance to our work. In [23, 24], a technique based on the detection of outliers is applied to find compromised nodes in a consensus-based spectrum sensing algorithm for ad hoc wireless networks. In [24], several attack models are proposed to subvert the spectrum sensing algorithm. One attack model is a covert adaptive data injection attack, which adjusts attack strategies by manipulating the sensing results. The proposed defense consists to isolate neighbor nodes that send numerical data that deviate too much from some norm. In [25], the detection of Byzantine nodes is derived from reputation-based trust management strategies. In this paper, one type of attacks consists of malicious robots injecting false data to neighbors in a multirobot system controlled by a consensus algorithm for the purpose of formation control. The proposed defense system consists to decrease the consensus weight contribution of a node that has its reputation drops during the computation of consensus states. Defense strategies against Byzantine attacks also originate from research in process control and control theory, fields where one of the focus is to provide methodological approaches to detect faulty components in a system. In [26, 27, 28] different approaches based on control theory are proposed to detect Byzantine attacks on consensus algorithms. In [26, 27], using model-based fault detection techniques, it is shown that if the network of consensus nodes is $2k + 1$ connected then up to k Byzantine nodes can be identified. However model-based proposals for detecting multiple attackers seem computationally costly, they likely have only limited applicability.

Our work focuses on detecting Byzantine attackers in the consensus-based NIDS of [1]. One of our two detection techniques, *outlier detection*, derives from outlier methods monitoring applications of consensus algorithms to cooperative spectrum sensing [24]. The second detection technique, *fault detection*, derives from a model-based fault detection technique in process control and control theory [27]. We also introduce an approach to remove compromised NIDS modules such that the intrusion detection system can be returned to a non-compromised state.

The removal of compromised modules conflicts with some mathematical assumptions about average consensus algorithms. Indeed, proofs that neighbor to neighbor data exchanges converge to a consensus are valid under the assumption the network is static. The removal of a compromised module changes the network topology of the NIDS, thus the system is no longer guarantee to work correctly even under normal circumstances (no attacks). Here, the relevant background research comes from dynamic consensus theory concerned with applications of consensus algorithms to dynamic network topologies, facing issues such as communication time-delays, failing physical links or network nodes and mobile wireless networks [29, 30, 31]. While our objective is only to logically remove (isolate) a compromised NIDS module, the network conditions this is creating are quite similar to the work in [32, 33, 34], therefore we have drawn our solution more specifically from these researches.

All the solutions proposed in this paper are based on local knowledge. Decisions to categorize modules as compromised and to further removing the compromised module depend on information gathered from neighbors only. The computation to detect and remove compromised modules is therefore fully distributed, thus keeping the consensus-based NIDS fully distributed. Last, we have designed our defense strategies to protect the NIDS against a single compromised module. Detecting and removing multiple compromised modules following potentially coordinated attackers is left to a future work.

The major contributions of this paper include presenting the impact of malicious peers on the detection capability of our consensus based Network Intrusion Detection Systems (NIDS) scheme. We analyze the vulnerabilities of consensus-based NIDS by proposing a Byzantine attack model, which aim to adjust and stealthily manipulate results. Our defense strategies detect and remove compromised NIDS modules without impacting the logical functionality of the system. We compare these strategies under various detection parameters and network topologies through extensive simulations and analysis using a real NIDS and the NSL-KDD data set [35]. Our results demonstrate that the conducted method can indeed unveil peers with malicious intend and disruptions in the information exchange of peer-to-peer NIDS.

In the remainder, we briefly describe the consensus-based NIDS in [1]. We point out variations of falsification attacks and outline our two detection techniques to adjust the trustworthiness of participating peers. Thereafter, we illustrate the salient features of our prediction model to identify Byzantine peers and describe a practical experiment we conducted to showcase its functionality.

2 Consensus based NIDS

This section describes the average consensus algorithm. Next, a summary of the consensus-based NIDS in [1] is provided. Lastly, we describe our approach to isolate compromised modules together with the mathematical background that supports this approach.

2.1 Average consensus

The average consensus algorithm computes the average sum $\frac{1}{n}\sum_{i=1}^{n} x_i$ of some initial values $x_1, x_2, \ldots, x_n$. It is a distributed algorithm where each process can be viewed as running independently on a particular node of an undirected graph. Let $G = (V, E)$ be such a graph where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of nodes, and E denotes the corresponding set of edges. Graphs have an adjacency structure represented by an $n \times n$ *adjacency matrix* (denoted by A here) where $a_{ij} = 1$ if and only if $(v_i, v_j) \in E$, $a_{ij} = 0$ otherwise. The adjacency structure of G defines for each node $v_i \in G$ a *neighborhood* $\mathcal{N}_i$ where $\mathcal{N}_i = \{v_j \in V | (v_i, v_j) \in E\}$.

Each node v_i of G computes the following recurrence equation:

$$x_i(t+1) = W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t), \quad (1)$$

where recurrence i is initialized with $x_i(0) = x_i$, the initial value of each node i (from now on we denote node v_i simply by i). The purpose of a consensus algorithm is to make "consensus", i.e. $x_i(t)$ converges asymptotically to $\frac{1}{n}\sum_{i=1}^{n} x_i$ for all nodes $i \in G$. As evidence from Equation (1), each node i obtains $x_i(t+1)$ using only its previous value $x_i(t)$ and the previous values $x_j(t)$ of the nodes that are in the neighborhood of i (x_j, $j \in \mathcal{N}_i$). Nonetheless, all nodes converge to $\frac{1}{n}\sum_{i=1}^{n} x_i$ because the diffusion of the local averages through neighborhoods that share common nodes accounts for all nodes computing the global average.

Whether nodes reach consensus and which particular consensus value is reached is determined by the dynamics of the linear dynamical system that equation (1) specifies, which in turn depends on the transition matrix W. Each entry W_{ij} of matrix Wrepresents a weight on edge $(i, j) \in G$. These individual weights have to be chosen carefully to ensure convergence, and convergence to a specific value. For example, in equation (1), making consensus on $\frac{1}{n}\sum_{i=1}^{n} x_i$ can be obtained by computing local averages of $x_i(t)$ and $x_j(t)$ for $j \in \mathcal{N}_i$ using $W_{ij} = \frac{1}{|\mathcal{N}_i|+1}$ for $(i, j) \in G$ (including self-edge (i, i)).

A system as in (1) can reach consensus if the weight matrix satisfies certain conditions, as stated in [36]. Two conditions concern our application of average consensus to network intrusion detection: 1- the undirected graph G needs to be connected, i.e. there is a path between each pair of nodes; 2- the weight matrix W must be row stochastic, i.e. $\sum_{j=1}^{n} W_{ij} = 1$, the sum of the weights of each row equal 1 (note that for undirected graph, $w_{ij} = w_{ji}$, therefore $W = W^T$, consequently the weight matrix W is doubly-stochastic, $\sum_{j=1}^{n} W_{ij} = \sum_{i=1}^{n} W_{ji} = 1$). Several weight matrices satisfy these conditions, the following matrices have been used for the consensus-based NIDS:

– *Metropolis-Hasting* matrix:

$$W_{ij} = \begin{cases} \frac{1}{1+\max(d_i,d_j)} & \text{if } i \neq j \text{ and } j \in \mathcal{N}_i \\ 1 - \sum_{k \in \mathcal{N}_i} W_{ik} & \text{if } i = j \\ 0 & \text{if } i \neq j \text{ and } j \notin \mathcal{N}_i \end{cases}$$

where $d_i = |\mathcal{N}_i|$.

– *Best-constant edge weight* matrix:

$$W_{ij} = \frac{2}{\lambda_1(L) + \lambda_{n-1}(L)}$$

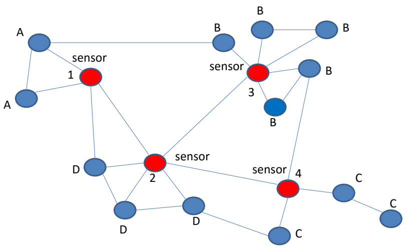

Figure 1: Network Intrusion Detection System.

where L is the Laplacian matrix of the NIDS network, λ_1, λ_{n-1} are the first and $n-1$ eigenvalues of L.

- *Local-degree weights* matrix where the weight of an edge is the largest degree of its two adjacent vertices

$$W_{ij} = \frac{1}{max\{d_i, d_j\}}.$$

- *Max-degree weight* where d_{max} is the largest degree of the vertices in the network

$$W_{ij} = \frac{1}{d_{max}}.$$

Note for the last three matrices, $W_{ii} = 1 - \sum_{k \in \mathcal{N}_i} W_{ik}$ and $W_{ij} = 0$ if $j \notin \mathcal{N}_i$. Note also these weight matrices guarantee asymptotic convergence, $x(t)$ converges to $\frac{1}{n}\sum_{i=1}^{n} x_i(0)$ as $t \to \infty$, we refer to this average consensus algorithm as the *asymptotic average consensus*. Weight matrices have an impact on the speed of convergence (the number of iterations needed to get close enough to the average sum) [37, 38]. Given an average consensus application, it worth to compare the convergence speed of different weight matrices to identify the one with the best performance. It worth noticing that the graph topology also impacts the convergence speed of average consensus algorithms [39].

2.2 Consensus-based NIDS

As pictured in Figure 1, a consensus-based NIDS is a set of modules each placed strategically on nodes of the monitored computer network such to observe traffic in the corresponding sub-network. Each module consists of traffic sensors that receive copies of all transported packets within the observed network and calculates an initial local probability for observing benign or malignant network traffic. The NIDS modules observing local network traffic are themselves connected by a physical network. Without lost of generality, we assume that the physical links connecting pairs of NIDS modules are direct (wired or wireless) physical links. The NIDS network is modeled by a graph where each node of the graph represents an NIDS module. It is assumed that this graph is connected. For the purpose of analysis and comparisons, we study specific topologies of NIDS networks, we refer to such specific network as an *NIDS network topology*.

2.2.1 Network traffic analysis

The detection method of each NIDS module is "anomaly based" using the well-known naive Bayes classifier. The analysis focuses on detecting Distributed Denial of Service (DDoS) attacks, such as Land-attack, Syn-flood and UDP-storm. The naive Bayes classifier assess the statistical normal behavior - the 'likelihood' of a set of values to occur - with the help of labeled historic data. Our set of m features includes most of the variables offered by the NLS KDD data set, such as the number of bytes, service, and number of connections. The probabilities of intrusion is computed for each of these features. $P(o_j|h)$ expresses the likelihood of the occurrence o_j given the historic anomalous h_a or normal h_n occurrences. Thus, if events receive the same values than benign or malignant network traffic during training, they result in a high probability for those. Assuming conditional independence of the m features, the joint likelihood $P(O_i|h)$ of NIDS module i is the product of all feature likelihoods:

$$P(O_i|h) = \prod_{j=1}^{m} P(o_j|h). \qquad (2)$$

Each NIDS module locally assigns the joint likelihood, indicating the abnormality of each event.

2.2.2 Consensus phase

Following the sensing and data analysis by the Bayesian network, each NIDS module enters into a phase where it computes the average sum of the n log-likelihoods: $\frac{1}{n}\sum_{i=1}^{n} x_i(0)$, while communicating only with direct neighbors. This phase is labeled as the NIDS *consensus phase*. Let $x_i(0) = \log(P(O_i|h))$ be the initial state of module i, where $x_i(0)$ is the likelihood for module i to see a certain set of network features. As explained in section 2.1, the average sum is computed iteratively and independently by each module i as a weighted sum of x_i and the x_j for $j \in \mathcal{N}_i$ as defined in equation (1). We identify as the *consensus loop* the iterations of equation (1) and $x_i(t+1)$ as the *consensus value* of module i at iteration $t+1$. The consensus phase is the computation performed by the n consensus loops to reach consensus. This phase is defined in mathematical terms by the following dynamical system:

$$x(t+1) = Wx(t), \; t = 0, 1, \ldots \qquad (3)$$

where $x(t)$ is a vector of n entries denoting the n consensus values at iteration t of the consensus phase, and W is the weight matrix.

The *stopping condition* of consensus loop i (also known as 'convergence parameter' of the recurrence i) is given by

$|x_i(t+1) - x_i(t)| < \epsilon$, i.e. when the change in the consensus value from iteration t to iteration $t+1$ is smaller than a pre-defined threshold value ϵ. For weight matrices satisfying convergence assumptions, the value $|x_i(t+1) - x_i(t)|$ decreases asymptotically as $t \to \infty$, once this value is smaller than ϵ, the corresponding consensus loop is said to have converged. A consensus phase is completed once each consensus loop has converged. The *number of iterations* of a consensus phase is given by the consensus loop that needed the largest number of iterations to satisfy the stopping condition. The *convergence speed* of a consensus phase is the number of iterations needed for the consensus phase to complete. The value of ϵ is set such to minimize the number of iterations during the consensus phase while insuring accuracy of the decision about the state of the network traffic. The consensus phase is synchronous, all nodes must have completed the consensus loop at iteration t before proceeding to execute the consensus loop iteration $t+1$. Finally, as a matter of implementation, once an NIDS module has converged, it stops updating its consensus value but continues to send the last updated value to its neighbors.

2.3 Removing compromised modules

As discussed in the introduction, once a NIDS module j has been identified by a neighbor i as compromised, module j must be logically disconnected from i to maintain the integrity of the intrusion detection system. It is relatively simple to disconnect an NIDS module locally because the weight matrix W is known to each NIDS module i (or at least the weights associated to row i are known). Once a node i has identified a neighbor j as compromised, node i simply can apply the following change to the weight matrix: $W_{ij} = 0$. Unfortunately, $W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}(t)$ no longer sum up to 1, then W fails to satisfy one of the two consensus convergence conditions. In order to fully address this issue, we have revisited the convergence proofs of average consensus, more specifically the convergence proofs for *dynamic consensus* (consensus under dynamic network topologies).

The consensus algorithm described in Section 2.1 is a *static consensus algorithm* because the weight matrix stay unchanged during the consensus phase. The weight matrix (which is actually a weighted adjacency matrix) mirrors the physical network topology underlying the NIDS. Static consensus cannot be used for applications where the underlying network topology is dynamic, i.e. where links or nodes fail, or where nodes enter and leave the network dynamically such as for wireless ad-hoc network. Dynamic consensus theory formally addresses consensus convergence issues arising in dynamical networks. Dynamic consensus is relevant to our work as the impact on the NIDS of logically removing a compromised module is (model wise) the same as a failing node. The convergence theory of dynamic consensus is the mathematical support to our solution strategy for the removal of compromised nodes in a consensus-based NIDS. There are several avenues in control theory to address dynamic network, the work in [32, 33, 34] is directly related to our problem.

As stated in section 2.1, the two convergence conditions the consensus phase of NIDS must satisfy are network connectivity and stochastic weight matrix. In [32], it is shown that the connectivity condition is surprisingly mild for dynamic network topologies, the collection of dynamically changing topologies during the consensus phase only needs to be jointly connected to guarantee convergence. In our work this condition is always satisfied. Only one NIDS module is removed during a consensus phase, therefore the collection of network topologies is limited to two. For the NIDS network topologies tested in the experimentation section of this paper, each topology is connected.

We have violation of convergence condition related to the stochastic weight matrix, this is fixed as followed. Once a node i has identified a neighbor j as compromised, node i set $W_{ij} = 0$, thus locally removing the link (i, j). The situation where $W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}(t) < 1$ from setting $W_{ij} = 0$ is eliminated by increasing the weight of the self-edge by the same amount W_{ij}: $W_{ii} = W_{ii} + W_{ij}$. This solution is only implementable if the information for updating the weights of row i in W can be computed locally. This is the case for the Metropolis-Hasting weight matrix as the weight of each edge depends on the degree of adjacent nodes. This will not work for weight matrices like the best-constant edge weight matrix or the max-degree weight matrix where the weights depend on global information (such as the max-degree node in the network). Tests in this paper where nodes are logically disconnected in a NIDS network topology are based on the Metropolis-Hasting weight matrix.

3 Detection of Byzantine attacks

Byzantine attacks aim at degrading the accuracy of the network intrusion detection system. Accuracy is defined as follow:

$$\frac{TP + TN}{TP + TN + FP + FN},$$

where TP (*True Positive*) is the number of attacks detected when it is actually an attack; TN (*True Negative*) is the number of normals detected when it is actually normal; FP (*False Positive*) is the number of attacks detected when it is actually normal; FN (*False Negative*) is the number of normals detected when it is actually an attack. Byzantine attacks of NIDS modules can aim at masking malicious traffic by decreasing the probability of attacks initially computed by the naive Bayesian. Attackers may also increase the probability of attacks computed by the naive Bayesian, thus increasing the number of false positives, the reliability of the system is then questioned by the system administrators.

This section first provides an attack model on the consensus phase, this model is used by tests conducted in the

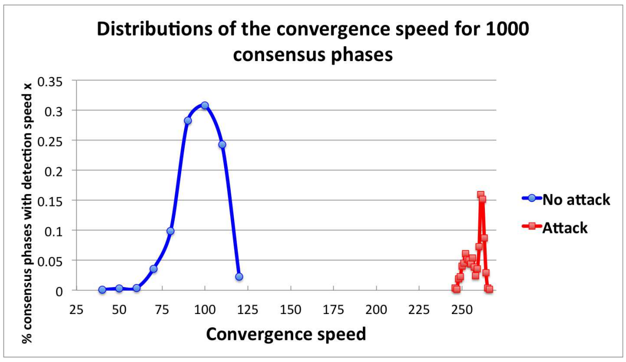

Figure 2: Convergence speeds with and without loop disruption.

next section. Second, two techniques are described which aim at identifying compromised NIDS modules.

3.1 Byzantine attack model

Byzantine attacks on the consensus phase of consensus-based intrusion detection algorithms can take the following forms [27, 40]:

1. Data falsification attacks: Sensor values are falsified, thus the consensus loop is initialized with values originating from falsified network traffic readings;

2. Consensus loop disruptions:

   (a) the attacker ignore the consensus value computed at each iteration and keeps transmitting the same constant c;

   (b) the attacker send to its neighbors a falsified consensus value [27].

Figure 2 illustrates the impact of a type 2(a) attack on the convergence speed of the consensus phase. It plots the distribution of the convergence speed of 1000 consensus phases each having only honest NIDS modules (No attack) versus a scenario where each consensus phase has one compromised module sending the same constant value c to its neighbors (Attack). Figure 2 shows that convergence speed is much slower in a compromised system, each consensus phase needing between 250 to 300 iterations to converge, while in a system without a compromised module consensus phases need between 40 to 125 iterations to converge. Moreover, NIDS modules in a compromised system fail to converge to the average consensus $\frac{1}{N}\sum_{i=1}^{N} x_i(0)$, rather they all converge to c [41].

In this paper we seek to discover consensus loop disruption attacks of type 2(b). Equation (4) below models this type of attacks inside the consensus loop of a compromised NIDS module:

$$x_j(t+1) = W_{jj}x_j(t) + \sum_{i\in\mathcal{N}_j} W_{ji}x_i(t) + u_j(t). \quad (4)$$

This recurrence equation is similar to equation (1) excepts for the variable $u_j(t)$ which models the value selected by the attacker for modifying the true consensus value of the compromised node j. The falsified consensus value $x_j(t+1)$ is sent to all the neighbors of node j at iteration $t+1$. Other Byzantine attack models, including multiple colluding attackers, are described in [24, 42].

3.2 Detection techniques

We describe two detection techniques that handle consensus loop disruptions of type 2(b) by a single Byzantine attacker. The first detection technique is an outlier detection procedure. This procedure is executed by each module i and evaluates at each consensus loop iteration the potential that a neighbor of module i is compromised. The second detection technique is an adaptation to cyber-attacks of a model-based fault-detection technique in process engineering and control theory. Like the first one, it is a procedure executed by each module, observing its neighbors, seeking to identify a compromised one.

3.2.1 Outlier detection

Outlier detection techniques have been applied to detect Byzantine attacks in wireless sensor networks [43]. These techniques use distance thresholds between the value $x_j(t)$ sent by a neighbor j to node i and some reference value r_i. For example, if $r_i(t) = x_i(t)$, neighbor j is flagged as compromised if $|x_j(t) - x_i(t)| > \lambda$ for some threshold value λ. However this idea had to be refined. For example, a unique predefined threshold for all nodes may easily be discovered by intruders. Furthermore, as nodes of a consensus algorithm converge to a same value, the absolute differences $|x_j(t) - x_i(t)|$ between two nodes i and j converge to zero as $t \to \infty$, rendering the outlier detection potentially insensitive when the absolute differences get smaller than λ.

Adaptive thresholds have been proposed to address the above issues [23, 24]. It consists for each node i to compute a local threshold λ_i and to adapt the threshold at each consensus iteration to the reduction of absolute differences $|x_j(t) - x_i(t)|$. In [23], the threshold

$$\lambda_i(t+1) = \frac{\sum_{j\in\mathcal{N}_i} |x_j(t+1) - x_i(t+1)|}{\sum_{j\in\mathcal{N}_i} |x_j(t) - x_i(t)|}\lambda_i(t) \quad (5)$$

(for properly initialized $\lambda_i(0)$) is computed by each node i and at each iteration of the consensus phase. The rule in equation (5) computes λ_i using the diffusion dynamics of consensus algorithms, so unless the attacker can get multi-hops information access, it cannot foresee the value of its neighbor thresholds. Consequently, the attacker cannot adapt its consensus loop disruption attack to keep the values under the radar of the detection procedure. As the network converges towards consensus, the value λ converges toward zero, leading to the attackers to be eventually filter out.

Note that $\lambda_i(t)$ partitions neighbors of node i into two sets, those neighbors j that have a deviation $|x_j(t) - x_i(t)| \geq \lambda_i(t)$ are considered suspicious, they constitute the neighborhood $\mathcal{N}_i^F$ of states that have less weight in the computation of the consensus value $x_i(t+1)$:

$$x_i(t+1) = x_i(t) + \epsilon \sum_{j \in \mathcal{N}_i^T} x_j(t) + \frac{\epsilon}{a} \sum_{j \in \mathcal{N}_i^F} x_j(t)$$

for some constant a. Our outlier detection method computes the threshold λ as in equation (5). Those neighbors j that have a deviation $|x_j(t) - x_i(t)| \geq \lambda_i(t)$ are flagged as suspicious. We use a majority rule similar to [24] to convert the status of a neighbor NIDS module j from suspicious to attacker. Let B be the number of common neighbors between module i and module j. If more then $\lceil \frac{B}{2} \rceil$ neighbors of module i report j as suspicious then module j is considered as compromised, it is disconnected/removed from the intrusion detection system. Note that we assume a single attacker, if the majority rule identifies more than one neighbor as compromised, the one with the largest deviation is disconnected from the NIDS network.

3.2.2 Model-based fault detection

Fault detection is a field of control engineering concerned with identifying and locating faulty components in a system. The techniques in this field essentially compare measurements of the actual behavior of a system with its anticipated behavior. In *model-based* fault detection, the anticipated behavior is described using mathematical models [44], the measured system variables are compared with their model estimates. Comparisons between the system and the model show deviations when there is a fault in the real system. Such difference between the system and its model is called *residual* or *residual vector*. There exist several implementations of the model-based approach, the *observer-based technique* [45] 1- seeks to discriminate between deviations caused by faults in the real process from those caused by the estimations; 2- provides a residual vector that indicates the faulty system components (so called directional residual). Observer-based approaches to cybersecurity have been proposed recently in different contexts [46, 47, 48, 49, 50]. We focus more specifically on applications of observer-based fault detection to identify Byzantine attackers in consensus-based algorithms [26, 27].

In order to detect Byzantine attackers during the consensus phase of the NIDS, the design of the consensus loop of each NIDS module is modified to include new matrices based computation that estimate the consensus state vector $x(t)$, we name *observer* this new function of the consensus loop. At each iteration of the consensus loop, the observer computes a state vector $x^o(t)$ estimating $x(t)$, where $x(t)$ is the vector storing the consensus values at iteration t of the consensus loop. We first model the consensus loop disruption attack of equation (4) in matrix form:

$$x(t+1) = Wx(t) + \mathbf{I}_n u(t) \tag{6}$$

where $\mathbf{I}_n$ is the n-dimension identity matrix, and where $u_i(t) = 0$ whenever NIDS module i behave normally. The observer requires inputs from the state vector $x(t)$, i.e. the values $x_j(t) \in x(t)$ where $j \in \mathcal{N}_i$. These values are stored in a vector y_i. The consensus loop of each NIDS module i is now defined as follow:

$$\begin{aligned} x(t+1) &= Wx(t) + \mathbf{I}_n u(t) \\ y_i(t) &= C_i x(t) \end{aligned} \tag{7}$$

where C_i is a $(deg_i + 1) \times N$ matrix in which entry $C_i[k,l] = 1$ if $l \in \mathcal{N}_i$, otherwise $C_i[k,l] = 0$. The vector $y_i(t)$ has $(deg_i + 1)$ entries, each entry j of $y_i(t)$ stores the state $x_j(t)$ at time t of modules $j \in \mathcal{N}_i$.

Equation (7) represents the consensus loop of a given module i as if it could access all the consensus values at iteration t, though in fact module i can only access $x_i(t)$ and $x_j(t)$ for $j \in \mathcal{N}_i$. The other entries of vector $x(t)$ are not needed during the computation performed by the recurrence relation of node i, so it is not incorrect to model these entries as if they were available. Note that each NIDS module i knows the consensus matrix W, the matrix C_i, $x_j(t) \in x(t)$ for $j \in \mathcal{N}_i$, and the identity matrix $\mathbf{I}_n$. However, the set of non-zero u_i is unknown to the non-malicious modules. To detect a malicious neighbor of module i, the consensus loop of "each module" computes the following matrix operations [27]:

$$\begin{aligned} z(t+1) &= (W + GC_i)z(t) - Gy_i(t) \\ x^o(t) &= Lz(t) + Ky_i(l) \end{aligned} \tag{8}$$

where $z(t)$ is the state of the observer and $x^o(t)$ is the estimation by the observer of module i of the consensus state $x(t)$. The matrices to compute $z(t+1)$ and $x^o(t)$ are defined as follow: $G = -W_{\mathcal{N}_i}$, $K = C_i^T$, $L = \mathbf{I}_n - KC_i$, where $W_{\mathcal{N}_i}$ are the columns of W with indexes in $\mathcal{N}_i$. The system in (8) has roots in the observability theory of control theory, a detailed analysis of this system is beyond the scope of this paper, we refer to [45] for an historical development and analysis of observer-based fault detection systems. The analysis of (8) can be simplified as the consensus system (7) satisfies some conditions [26]. It can be show that as $t \to \infty$, $x^o(t) \to x(t)$, consequently the *estimation error* $e(t) = x^o(t) - x(t)$ converges to $\mathbf{0}$. We are also given that equation (8) under the consensus system in (7) simplifies to:

$$x_j^o(t) = \begin{cases} x_j(t) & \text{if } j = i \text{ or } j \in \mathcal{N}_i \\ z_j(t) & \text{otherwise} \end{cases} \tag{9}$$

and that the state of the observer $z(t+1)$ can be expressed in terms of the consensus matrix [27]:

$$z(t+1) = Wx^o(t). \tag{10}$$

The *iteration error* $\varepsilon(t)$:

$$\varepsilon(t) = |x^o(t+1) - Wx^o(t)|$$

can then be used as residual vector. From (9) and (10), $\varepsilon_j(t) = 0$ for $j \neq i$ and $j \notin \mathcal{N}_i$. If $\varepsilon_j(t) \neq 0$, either $x^o_j(t) \neq x_j(t)$ (estimation error is greater than 0), or $u_j \neq 0$. Since the estimation error dissipates as $t \to \infty$, we have $(x^o(t+1) - Wx^o(t)) \to \mathbf{I}_n u(t)$ as $t \to \infty$. If $u_j \neq 0$ for some $j \in \{1, \ldots, n\}$ then $(x^o_j(t+1) - Wx^o_j(t)) \to u_j(t)$, the corresponding module j is detected as compromised.

Together with the consensus loop in (7), the observer defined in (8) provides an algorithm where each NIDS module can detect whether one of its neighbor sends falsified consensus data. Each module i build a consensus system and an observer as described in equations (7) and (8). At each consensus iteration, each module i computes $\varepsilon(t) = |x^o(t+1) - Wx^o(t)|$. If $\varepsilon_j(t) \neq 0$ then module $j \in \mathcal{N}_i$ is compromised. Module i then removes logically module j from its neighborhood by modifying it's weight matrix according to the description in Section 2.3, thus stopping the injection of an external input by module j into the network intrusion detection system.

4 Empirical analysis

The above two Byzantine attacks detection techniques help the NIDS coping with adversarial environments by detecting compromised modules. In this section we analyze and compare the behavior of each technique. For example, these techniques have a computational cost, we measure the overhead for running each technique. We measure how fast attacks are detected and model the accuracy of the decisions made by the NIDS under each detection method. Last, as the removal of a compromised module is obtained by changing the weight matrix and the network topology, we measure whether these changes have any impact on the convergence speed of the consensus phases, i.e. whether the system returns to its full functioning capabilities after removing compromised modules.

To execute this empirical analysis, the two Byzantine attack identification techniques described in Section (3) have been coded as part of the consensus phase of the NIDS simulations described in [1]. We have run simulations for the following NIDS network topologies: rings with 9 and 25 nodes (NIDS modules), 2-dimensional torus with 9 and 25 nodes, Petersen graph (10 nodes 15 edges) and several random graphs having the same number of vertices and edges as in the Petersen graph. A simulation consists to execute 1000 iterations of one of the above NIDS network topologies. In one iteration, each NIDS module of the network topology reads the local network traffic from an entry of the NSL-KDD data set, performs an Bayesian analysis of the local traffic, then executes its consensus loop until convergence. Note, we have filtered attacks in the NSL-KDD data set to retain only denial of service attacks.

The consensus phase is implemented as follow. The Bayesian analysis of the local network traffic by module i returns two values: p_{A_i} the probability that the observed traffic at module i is intrusive; p_{N_i} the probability the observed traffic at module i is normal. These values are used to initialize the consensus loop of the corresponding module i: $x^A_i(0) = \log(p_{A_i})$ and $x^N_i(0) = \log(p_{N_i})$, for $i = 1..n$. During the consensus phase, for simulations involving the outlier detection technique, each NIDS module i computes the following recurrence relations until $|x^A_i(t+1) - x^A_i(t)| < \epsilon$ and $|x^N_i(t+1) - x^N_i(t)| < \epsilon$:

$$x^A_i(t+1) = W_{ii}x^A_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x^A_j(t) + u_i(t) \quad (11)$$

$$x^N_i(t+1) = W_{ii}x^N_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x^N_j(t) + u_i(t). \quad (12)$$

Similarly, in simulations involving the fault detection technique, each NIDS module i computes the solutions for the following iterative systems until each recurrence of the systems has converged:

$$\begin{aligned} x^A(t+1) &= Wx^A(t) + \mathbf{I}u^A \\ y^A_i(t) &= C_i x^A(t) \end{aligned} \quad (13)$$

$$\begin{aligned} x^N(t+1) &= Wx^N(t) + \mathbf{I}u^N \\ y^N_i(t) &= C_i x^N(t). \end{aligned} \quad (14)$$

The matrix operations described in Equation (8) are also computed at each iteration of the consensus loop of module i in simulations involving the fault detection technique.

Once a consensus phase is completed, each NIDS module i *decides* whether to raise an alert or not based on its consensus approximation ratio $\frac{x^A_i(t)}{x^N_i(t)}$ of the actual ratio $\frac{\sum_{i=1}^N \log(p_{A_i})}{n} / \frac{\sum_{i=1}^n \log(p_{N_i})}{n}$ and some predefined alert value ratio. As each module converges asymptotically to the same actual ratio $\frac{\sum_{i=1}^N \log(p_{A_i})}{n} / \frac{\sum_{i=1}^n \log(p_{N_i})}{n}$, all modules reach a same decision, which constitutes a form of coordinated response to perceived anomalies in the network traffic.

The consensus loop disruption of the attack model 2(b) in Section 3.1 is implemented as follow. Anomaly-based intrusion detection systems tend to have high false positive rates. We simulate attacks that aim to further increase the number of false positives. Attacks inject positive values in the consensus loop component (11) or (13). At each iteration of a simulation, a to be compromised NIDS module j is selected randomly, u^A_j is then assigned with a positive value. The magnitude of u^A_j has to be large enough to falsify the decision at the end of the consensus phase (i.e. raise an alert when traffic is normal), "if" the consensus loop disruption attack is not detected. For example, $u^A_j = 0.0005$ is to small, it does not have an impact on the decision. However, a value such as $u^A_j = 0.5$ can cause each module of the system to converge to an approximation $\frac{x^A_i(t)}{x^N_i(t)} > \frac{\sum_{i=1}^N \log(p_{A_i})}{n} / \frac{\sum_{i=1}^n \log(p_{N_i})}{n}$, thus possibly leading the NIDS to raise an alert when in fact there is no attack.

The value $u^A_j = 0.5$ is also suitable to obtain meaningful test results for the following reason. The values p_{A_i} and p_{N_i} returned by the Bayesian analysis of a module i are the product of likelihoods $\prod_{j=1}^{m} P(o_j|h)$, as the number of features is large, the product of likelihoods are very small. During the consensus phase, neighbor NIDS modules exchange log-likelihoods, which are in the range between -20 and -55. So $u_j^A = 0.5$ is a relativity small external input during the consensus phase. However, it is large enough so that our two detection techniques can always detect this attack, but failing to detect it soon enough can lead the consensus phase to converge to values quite different from $\frac{1}{n}\sum_{i=1}^{n} x_i(0)$.

In the following sections, we first evaluate the computational cost of running each of the two detection techniques. Subsidiary, we also report the number of consensus iterations needed to detect a compromised module. Next we analyze the efficiency of the detection techniques to prevent the occurrence of false positives at the conclusion of a consensus phase. Finally we analyze the impact of our technique to remove a compromised NIDS module on the convergence speed of consensus phases.

4.1 Computational costs

Table 1 reports the computational cost of running each detection technique. All the simulations are executed while no attack take place, these tests measure uniquely the overhead for running the code implementing the two detection techniques. The column "Cost" reports the time in milliseconds for running the NIDS network simulation for 1000 iterations. In Table 1, rows "no detection" give the cost of running a NIDS simulation without the execution of any detection code. Rows "outlier" and "fault" give the cost of running NIDS modules while also executing respectively the code for the outlier method and the fault method. The higher costs of the detection techniques compared to "no detection" for the same network size and topology reflects the cost for protecting the consensus-based NIDS with the corresponding detection techniques. Table 1 shows that the computational overhead for outlier is clearly less than for the fault detection method. These results were expected, each consensus loop iteration of the fault detection method runs several matrix operations compared to simple scalar operations for the outlier method.

4.2 Detection speed

Figures 3 to 8 detail with which rapidity, *detection speed*, the two detection techniques identify compromised modules. Each figure corresponds to a different network topology. The values on the x axis are the number of consensus iterations needed before the compromised module is identified. The y axis displays the percentage of the 1000 consensus phases that needed a given number of consensus iterations to detect a compromised module. These figures clearly show that the fault detection approach needs fewer iterations to detect Byzantine attacks. Combining the computational cost in Table 1, we observe that the outlier method has a more favorable computational overhead but requires more iterations to detect compromised modules.

Table 1: Consensus-based NIDS computational simulation costs in milliseconds.

Topology	Size	Detection	Cost
Ring	9	no detection	0.050
		outlier	0.276
		fault	0.921
	25	no detection	0.101
		outlier	1.131
		fault	3.286
Torus	9	no detection	0.027
		outlier	0.121
		fault	1.327
	25	no detection	0.043
		outlier	0.567
		fault	6.055
Petersen	10	no detection	0.005
		outlier	0.135
		fault	0.597
Random	10	no detection	0.013
		outlier	0.290
		fault	1.268

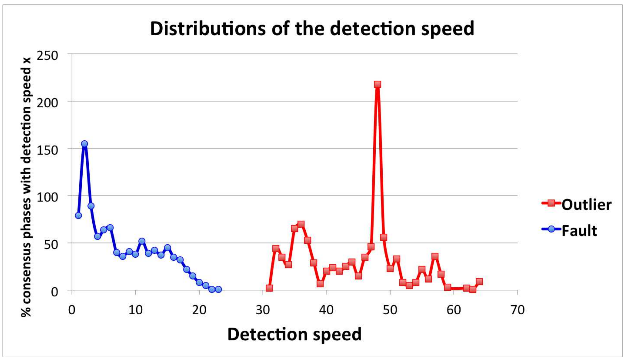

Figure 3: Detection speed of ring topology 9 nodes.

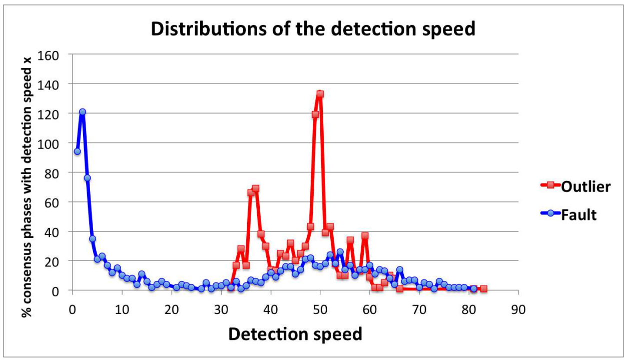

Figure 4: Detection speed of ring topology 25 nodes.

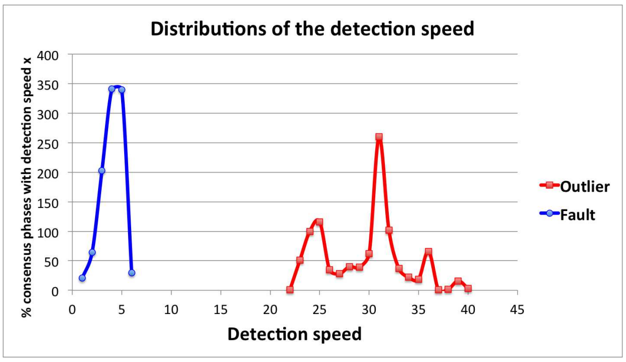

Figure 5: Detection speed of torus topology 9 nodes.

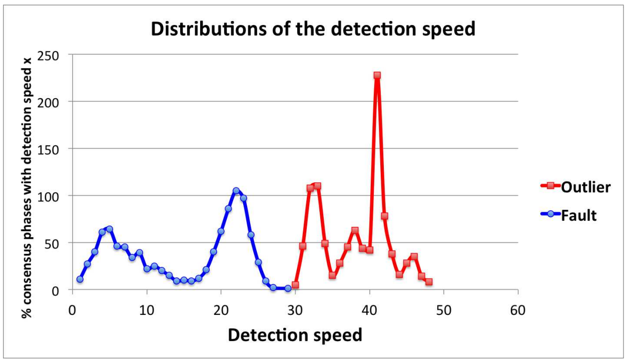

Figure 6: Detection speed of torus topology 25 nodes.

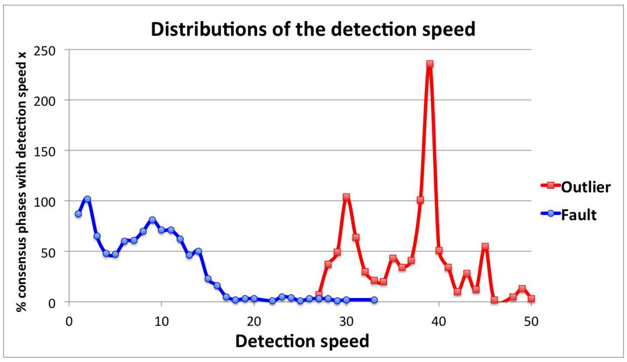

Figure 7: Detection speed of Petersen graph.

4.3 Intrusion detection accuracy

Disruption of the consensus loops by injecting external inputs has an impact on the accuracy of the decision made by the NIDS about the state of the network traffic. Table 2 measures how effective the two detection techniques are at maintaining the accuracy of the consensus-based NIDS. The "no attack" rows report the accuracy of the NIDS in a non-adversarial environment. The "no detection" rows report the accuracy of the NIDS when attacks take place while the NIDS is not protected. The "outlier" and "fault" rows report respectively the accuracy of NIDS protected by the outlier and fault detection methods.

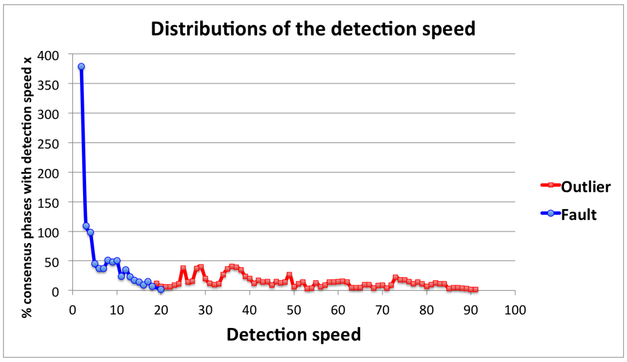

Figure 8: Detection speed of Random graphs.

The results of Table 2 are obtained without changing the weight matrix and network topology once a compromised NIDS module is identified (static consensus). Let l be the iteration of the consensus phase where module i identifies a neighbor module j as compromised. For $t > l$, module i applies the following update rule:

$$\begin{aligned} x_i^A(t+1) &= W_{ii}x_i^A(t) + \sum_{(k\in\mathcal{N}_i \wedge k\neq j)} W_{ik}x_k^A(t) \\ &\quad +W_{ij}x_j^A(t) - 0.5. \end{aligned}$$

This update is possible since $u_j^A = 0.5$ is known in the context of our simulations, though it is not know which module is compromised in this way. Module i removes 0.5 from the value sent by the compromised module j, therefore module i update its state with the true consensus values of its neighbors.

Table 2 shows the outlier detection method less accurate compared to the fault-detection method, this even Byzantine attacks are always detected and the compromised NIDS module neutralized. These results are explained by the number of consensus loop iterations needed to detect attackers. Figures 3 to 8 show the outlier method needing more iterations to detect compromised modules. The more iterations it takes to detect a compromised module, the more data injections take place prior to detection of the compromised module and the more time injected values have to diffuse across the NIDS modules, which cause the NIDS decision to till the wrong way more frequently in the case of the outlier method.

4.4 Convergence speed

This section analyzes the impact of removing a module while the NIDS computes consensus states. According to section 2.3, the technique we propose to isolate a compromised module satisfies the average consensus convergence conditions. Rows of the weight matrices sum up to 1. Each

Table 2: Accuracy of the NIDS.

Topology	Size	Detection	TP	TN	FP	FN
Ring	9	no attack	466	520	14	0
		no detection	521	0	479	0
		outlier	522	404	74	0
		fault	456	525	4	15
	25	no attack	527	473	0	0
		no detection	475	0	525	0
		outlier	497	503	58	0
		fault	506	489	0	5
Torus	9	no attack	493	491	16	0
		no detection	499	0	501	0
		outlier	495	438	67	0
		fault	478	511	0	11
	25	no attack	492	507	1	0
		no detection	497	0	503	0
		outlier	491	456	53	0
		fault	518	450	32	0
Petersen	10	no attack	501	487	12	0
		no detection	481	0	519	0
		outlier	477	458	65	0
		fault	481	516	0	3
Random	10	no attack	451	533	16	0
		no detection	485	0	515	0
		outlier	503	432	65	0
		fault	526	464	0	10

NIDS network topology tested in this empirical analysis section is such that it is still connected even after one module is removed. However, as our approach changes the weight matrix and the NIDS network topology, two factors that could impact the convergence speed, we still need to analyze the consensus phase convergence speed when modules are removed. In this section we compare the consensus phase convergence speed of the static consensus implementation of Section 4.3 running the Metropolis-Hasting weight matrix with the convergence speed when the consensus phase is implemented with the dynamic consensus procedure introduced in Section 2.3.

Figures 9 to 13 compare the convergence speed of static versus dynamic consensus for the outlier detection method while figures 14 to 18 compare the convergence speed of static versus dynamic consensus for the fault detection method. As we can see from figures 14 to 18, there is no significant differences in the convergence speed of static and dynamic consensus for the fault-based detection method, except for the Petersen graph. With the outlier detection method, as shown in figures 9 to 13, dynamic consensus converges faster for some of the network topologies.

It is not entirely clear why the convergence speed is better with dynamic consensus and some specific outlier simulations. Nonetheless, figures 9 to 18 show no significant decrease in the convergence speed of consensus phases once a module has been isolated. As accuracy is not impacted by the removal of a module, this is enough to conclude that the intrusion detection system returns to a fully functioning state.

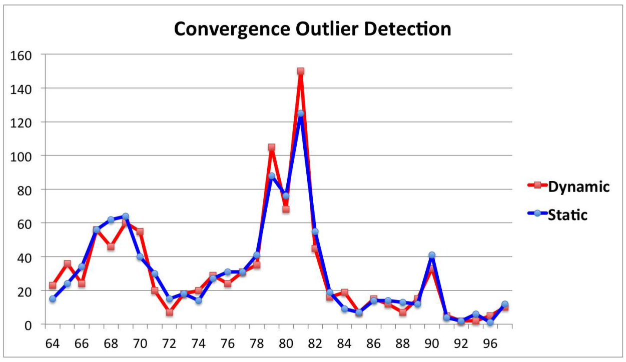

Figure 9: Convergence: dynamic topology, outlier detection, ring topology 9 nodes.

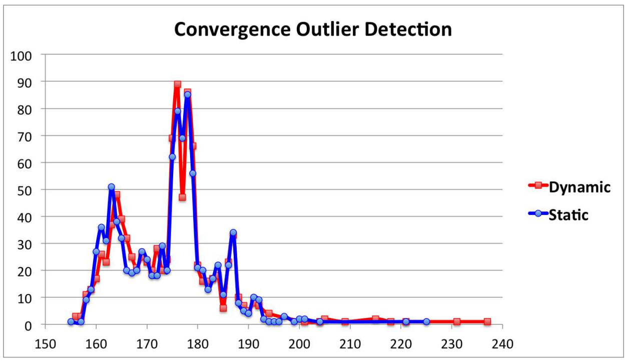

Figure 10: Convergence: dynamic topology, outlier detection, ring topology 25 nodes.

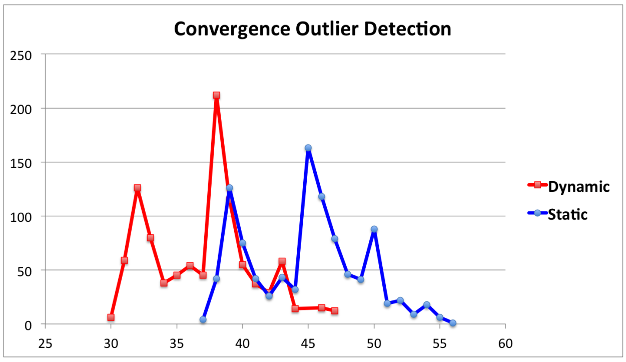

Figure 11: Convergence: dynamic topology, outlier detection, torus topology 9 nodes.

5 Conclusion

Local data exchanges of consensus-based distributed applications can be hacked by Byzantine attackers falsifying computed consensus information. Several solutions have been proposed in the literature that address Byzantine attacks on consensus algorithms. We have adapted two of these solutions, one from model-based fault detection and one from outlier detection to protect a consensus-based net-

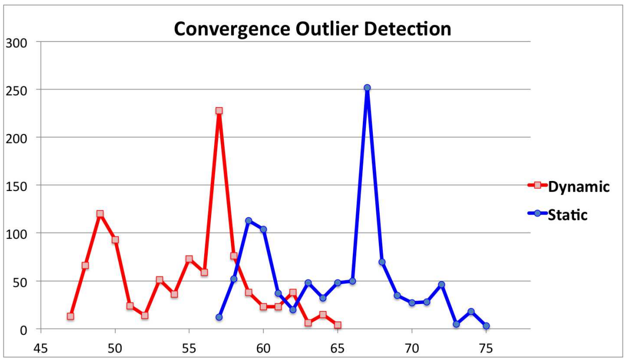

Figure 12: Convergence: dynamic topology, outlier detection, torus topology 25 nodes.

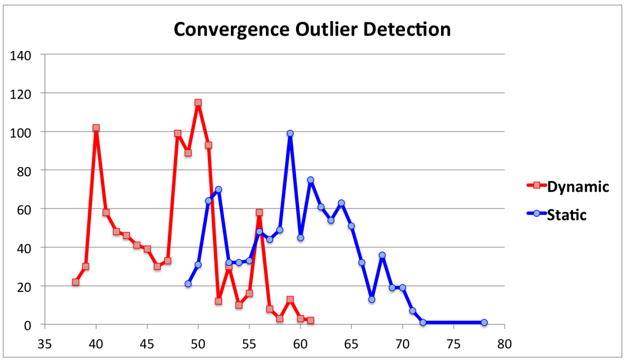

Figure 13: Convergence: dynamic topology, outlier detection, Petersen graph.

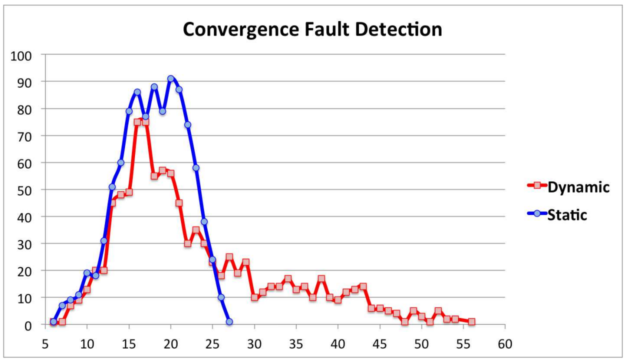

Figure 14: Convergence: dynamic topology, fault detection, ring topology 9 nodes.

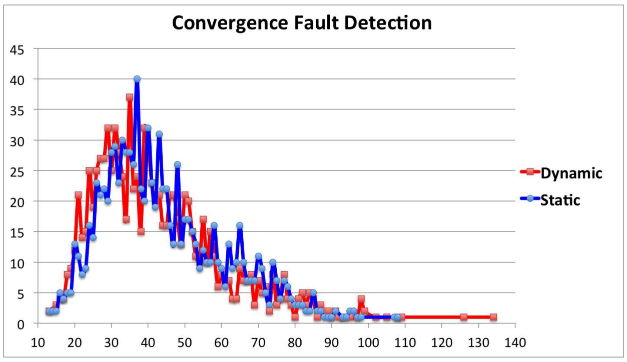

Figure 15: Convergence: dynamic topology, fault detection, ring topology 25 nodes.

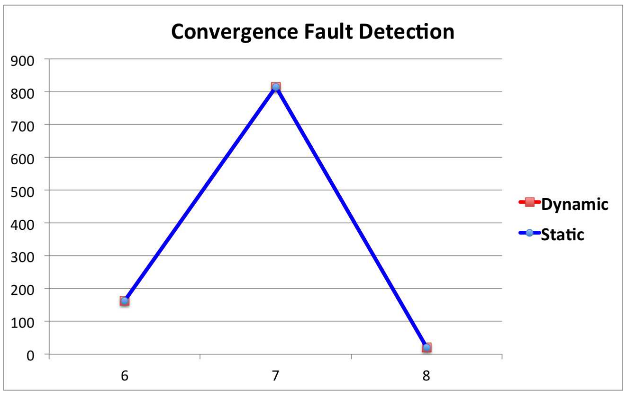

Figure 16: Convergence: dynamic topology, fault detection, torus topology 9 nodes.

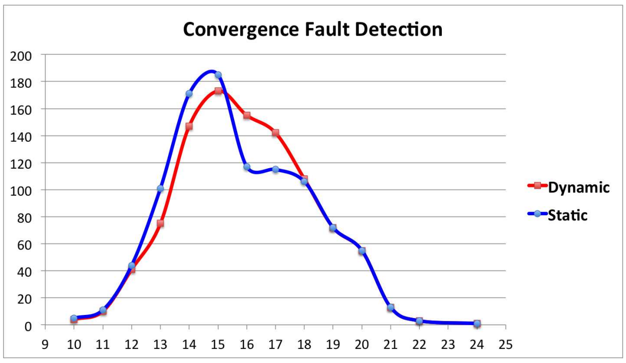

Figure 17: Convergence: dynamic topology, fault detection, torus topology 25 nodes.

work intrusion detection system. We have also applied results from dynamic consensus theory to derive a simple approach to isolate compromised modules from the network while continuing to satisfy the mathematical assumptions requested for convergence of the consensus phase.

Our results show the two methods we propose can be used to detect consensus loop disruptions and prevent falsifications of NIDS network traffic assessments. Though preliminary, our results also show significant computational costs of these approaches either in terms of the number of iterations to detect attacks (outlier detection) or in terms of the computational cost of each iteration (model-based detection). This might raise issues for deploying consensus-based NIDS in suitable environments such as wireless ad hoc networks.

Future work will address both protecting the consensus-based NIDS against disruptive attacks as well as getting the

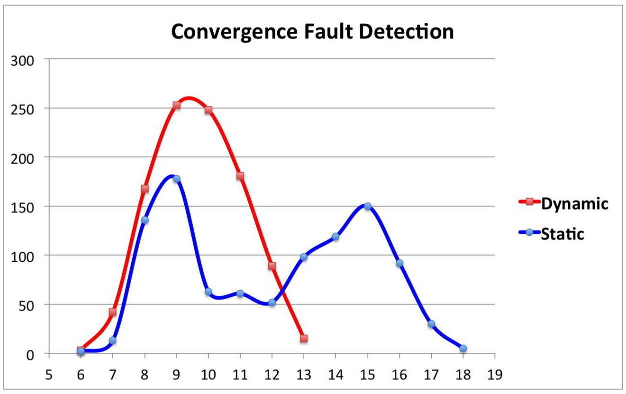

Figure 18: Convergence: dynamic topology, fault detection, Petersen graph.

system closer to deployment in wireless network environments. We will work on reducing computational cost, by speeding up for example the consensus phase, i.e. reducing the number of iterations needed for modules to come with agreed decisions. This will impact Byzantine fault detection which will need to be done at earlier stages and at a smaller computational cost. Byzantine fault detection will be broaden to other attack models, involving more than one compromised module, and possibly colluding attackers. Addressing multiple attackers seems achievable without too much research efforts using outlier or reputation-based methods. On the other hand, current model-based approaches in control theory seem too computationally demanding and will need more research before they can be used in a deployed system. Finally, we intend to broaden the cooperation among NIDS modules. This depends on consensus computing more functions of the initial values provided by the analysis phase. There is a wide range of functions that can be computed using distributed iterative methods similar to average consensus. This will bring more versatility in detecting network intrusions and allow for a wide range coordinated responses to address detected malicious network activities.

References

[1] M. Toulouse, B. Q. Minh, and P. Curtis, "A consensus based network intrusion detection system." in *IT Convergence and Security (ICITCS), 2015 5th International Conference on*. IEEE, 2015, pp. 1–6. [Online]. Available: http://dblp.uni-trier.de/db/conf/icitcs/icitcs2015.html#ToulouseMC15

[2] A. Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino JúNior, "Review: An intrusion detection and prevention system in cloud computing: A systematic review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 25–41, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2012.08.007

[3] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124 – 140, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016740480900073X

[4] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 55:1–55:33, May 2015. [Online]. Available: http://doi.acm.org/10.1145/2716260

[5] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance *et al.*, "Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype," in *Proceedings of the 14th national computer security conference*, vol. 1. Citeseer, 1991, pp. 167–176.

[6] T. Bass, "Multisensor data fusion for next generation distributed intrusion detection systems," in *In Proceedings of the IRIS National Symposium on Sensor and Data Fusion*, 1999, pp. 24–27.

[7] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. IEEE, 2003, pp. 226–231.

[8] C. V. Zhou, S. Karunasekera, and C. Leckie, "A peer-to-peer collaborative intrusion detection system," in *2005 13th IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Communic*, vol. 1, Nov 2005, pp. 118–123.

[9] M. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and p2p intrusion detection," in *In Proceedings of the IEEE Information Assurance Workshop (IAW*, 2005, pp. 333–339.

[10] M. Marchetti, M. Messori, and M. Colajanni, *Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 475–490. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04474-8_37

[11] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

[12] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, pp. 1226–1229, Aug 1995. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.75.1226

[13] R. Saber and R. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 2, June 2003, pp. 951–956.

[14] A. Fagiolini, M. Pellinacci, M. Valenti, G. Dini, and A. Bicchi, "Consensus-based distributed intrusion detection for multi-robot systems," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 120 – 127.

[15] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[16] S. Li, G. Oikonomou, T. Tryfonas, T. Chen, and L. Xu, "A distributed consensus algorithm for decision-making in service-oriented internet of things," *Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1461–1468, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6740862

[17] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton University Press, 2016.

[18] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," *Phys. Commun.*, vol. 4, no. 1, pp. 40–62, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.phycom.2010.12.003

[19] G. Xiong and S. Kishore, "Consensus-based distributed detection algorithm in wireless ad hoc networks," in *Signal Processing and Communication Systems, 2009. ICSPCS 2009. 3rd International Conference on*, Sept 2009, pp. 1–6.

[20] K. Avrachenkov, M. E. Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011, pp. 1–6.

[21] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980. [Online]. Available: http://doi.acm.org/10.1145/322186.322188

[22] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[23] S. Liu, H. Zhu, S. Li, X. Li, C. Chen, and X. Guan, "An adaptive deviation-tolerant secure scheme for distributed cooperative spectrum sensing," in *2012 IEEE Global Communications Conference, GLOBECOM 2012, Anaheim, CA, USA, December 3-7, 2012*, 2012, pp. 603–608. [Online]. Available: http://dx.doi.org/10.1109/GLOCOM.2012.6503179

[24] Q. Yan, M. Li, T. Jiang, W. Lou, and Y. T. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 900–908.

[25] W. Zeng and M.-Y. Chow, "A reputation-based secure distributed control methodology in D-NCS." *IEEE Trans. Industrial Electronics*, vol. 61, no. 11, pp. 6294–6303, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/tie/tie61.html#ZengC14

[26] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus computation in unreliable networks: A system theoretic approach," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 90 – 104, Jan. 2012.

[27] ——, "Distributed intrusion detection for secure consensus computations," in *Decision and Control, 2007 46th IEEE Conference on*, Dec 2007, pp. 5594–5599.

[28] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1495–1508, July 2011.

[29] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33 – 46, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731506001808

[30] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322 – 329, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109809004828

[31] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1109/tac.2004.834113

[32] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," 2006, unpublished. [Online]. Available: http://web.stanford.edu/~boyd/papers/pdf/avg_metropolis.pdf

[33] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proceedings of the Fifth International Conference*

on Information Processing in Sensor Networks, IPSN 2006, Nashville, Tennessee, USA, April 19-21, 2006, 2006, pp. 168–176. [Online]. Available: http://doi.acm.org/10.1145/1127777.1127806

[34] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1147685.1147698

[35] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 53–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=1736481.1736489

[36] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2006.08.010

[37] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 33–55, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1137/060678324

[38] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.

[39] S. Kar and J. M. F. Moura, "Topology for global average consensus," October 2006, pp. 276–280.

[40] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 145–158, March 2017.

[41] W. Ben-Ameur, P. Bianchi, and J. Jakubowicz, "Robust average consensus using total variation gossip algorithm," in *6th International ICST Conference on Performance Evaluation Methodologies and Tools, Cargese, Corsica, France, October 9-12, 2012*, 2012, pp. 99–106. [Online]. Available: http://dx.doi.org/10.4108/valuetools.2012.250316

[42] S. Mi, H. Han, C. Chen, J. Yan, and X. Guan, "A secure scheme for distributed consensus estimation against data falsification in heterogeneous wireless sensor networks," *Sensors*, vol. 16, no. 2, p. 252, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/2/252

[43] V. P. Illiano and E. C. Lupu, "Detecting malicious data injections in wireless sensor networks: A survey," *ACM Comput. Surv.*, vol. 48, no. 2, pp. 24:1–24:33, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2818184

[44] R. Isermann, "Model-based fault-detection and diagnosis - status and applications," *Annual Reviews in Control*, vol. 29, pp. 71–85, 2005.

[45] J. Chen, J. R. Patton, and H.-Y. Zhang, "Design of unknown input observers and robust fault detection filters," *International Journal of Control*, vol. 63, no. 1, pp. 85–105, 1996.

[46] Z. A. Biron, P. Pisu, and B. HomChaudhuri, "Observer design based cyber security for cyber physical systems," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, ser. CISR '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/2746266.2746272

[47] D. Ding, Z. Wang, D. W. C. Ho, and G. Wei, "Observer-based event-triggering consensus control for multiagent systems with lossy sensors and cyber-attacks," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2016.

[48] F. Pasqualetti, R. Carli, A. Bicchi, and F. Bullo, "Identifying cyber attacks via local model information," in *International Conference on Decision and Control - CDC 2010*, Atlanta, USA, December 2010, pp. 5961 – 5966.

[49] A. Teixeira, H. Sandberg, and K. H. Johansson, "Networked control systems under cyber attacks with applications to power networks," in *Proceedings of the 2010 American Control Conference*, June 2010, pp. 3690–3696.

[50] L. Negash, S. Kim, and H. Choi, "Distributed unknown-input-observers for cyber attack detection and isolation in formation flying uavs," *CoRR*, vol. abs/1701.06325, 2017. [Online]. Available: http://arxiv.org/abs/1701.06325