Protecting Consensus Seeking NIDS Modules against Multiple Attackers

Michel Toulouse
Vietnamese-German University
Binh Duong New City, Vietnam
michel.toulouse@vgu.edu.vn

Phuong Khanh Nguyen
Hanoi university of Science and Technology
Hanoi, Vietnam
phuongnk@soict.hust.edu.vn

ABSTRACT

This work concerns distributed consensus algorithms and application to a network intrusion detection system (NIDS) [21]. We consider the problem of defending the system against multiple data falsification attacks (Byzantine attacks), a vulnerability of distributed peer-to-peer consensus algorithms that has not been widely addressed in its practicality. We consider both naive (independent) and colluding attackers. We test three defense strategy implementations, two classified as outlier detection methods and one reputation-based method. We have narrowed our attention to outlier and reputation-based methods because they are relatively light computationally speaking. We have left out control theoretic methods which are likely the most effective methods, however their computational cost increase rapidly with the number of attackers. We compare the efficiency of these three implementations for their computational cost, detection performance, convergence behavior and possible impacts on the intrusion detection accuracy of the NIDS. Tests are performed based on simulations of distributed denial of service attacks using the KSL-KDD data set.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; *Mobile and wireless security*; *Denial-of-service attacks*;

KEYWORDS

Network Intrusion Detection System; Anomaly-based; Outlier Detection; Reputation-based detection; Average-Consensus Algorithm;

ACM Reference Format:
Michel Toulouse and Phuong Khanh Nguyen. 2017. Protecting Consensus Seeking NIDS Modules against Multiple Attackers. In *SoICT '17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3155133.3155185

1 INTRODUCTION

Consensus seeking is a process in which humans reach decision unanimity from an initially divergent set of opinions. The state of unanimity is not achieved by delegating the decision to a single entity, rather it is a process of local opinion exchanges among pairs or overlapping subgroups of humans. The automation of consensus is now understood as a form of machine learning, but the mathematical formalization of this process started much earlier in the 50's as stochastic models, though the original formalization is best remembered today for the 1974 statistical model in "Reaching a consensus" by Morris H. DeGroot [4] (see historical references therein). This model has soon inspired algorithms for distributed computation over networks such as classifying multi-source data over sensor networks [1], distributed optimization [23], and as a model of flocking behaviors by physicists Vicsek et al. [24]. More recently we find consensus algorithms in a wide range of distributed applications in which data are collected or stored redundantly and then fused such as in sensor networks, distributed robotics, cognitive radio networks, distributed agent systems, duplicated databases, reconnaissance systems, aircraft altimeters, autonomous vehicles formation, air traffic control, etc. See [14] for an extensive survey on the background, applications and key mathematical issues related to linear consensus algorithms.

Consensus algorithms specify how data are exchanged between a node and its neighbors in the network and how exchanged data are processed by each node. Like in human consensus, this is an iterative process. First nodes in the same neighborhood agree on a common opinion based on their local data. Nodes repeatedly exchange and process agreed information among neighbors until the initial state from each node has diffused across the whole network, causing nodes to converge on a consensus value that reflects the initial state of all nodes in the network. A key issue therefore in the design of a consensus algorithm is the proof that the iterative process converges to the desired consensus. However this issue is largely settled from the wide body of mathematical results on convergence conditions that has been published (see [25] and references therein).

Consensus algorithms can also be found in critical infrastructure distributed applications (IoT [11], financial markets [22]). The next important issue to be addressed is resilience of convergence conditions under faulty equipments (network nodes or links failures) and more recently malicious actors. What types of attack vulnerabilities consensus algorithms arbor and what are the necessary conditions to design robust consensus algorithms subjected to adversarial environments. The first works credited for addressing these issues are Pease et all [17] and Lamport et all [10] in what has since been known as "the Byzantine agreement problem", and later on the work of N. A. Lynch [13]. The Byzantine agreement problem addresses consensus convergence issues when some of the network nodes have been captured or infiltrated by malicious actors. Contrary to faulty nodes, Byzantine nodes appear to work normally while

SoICT '17, December 7–8, 2017, Nha Trang City, Viet Nam

ACM ISBN 978-1-4503-5328-1/17/12…$15.00
https://doi.org/10.1145/3155133.3155185

in fact they can be sending falsified data or disrupting consensus computation, or both. Byzantine nodes could be extremely difficult to detect, honest nodes believe they receive genuine information, but then fail to agree or reach a consensus induces by the Byzantine attackers.

Byzantine attacks are not only more difficult to detect than faulty nodes, the likelihood of a system been subjected to multiple attackers is much greater than multiple nodes faulting. Defense strategies against Byzantine attacks do not usually differentiate between single and multiple attackers. Most of the proposed defense methods are assumed to work for multiple attackers but in practice are tested for a single attacker [5, 9, 12, 15, 16, 18, 26, 28, 29]. Expectedly, generalized frameworks do not differentiate between single or multiple attackers. The formulation of the Byzantine General Problem in [10] also assume multiplicity of the traitors, providing bounds on the maximum number of attackers beyond which the Byzantine agreement problem has no solution. But is it so in the practicality, it is not clear whether the proposed defense strategies against Byzantine attacks are as efficient when multiple nodes are compromised. For example, several defense strategies are based on outliers detection, but detection of outliers works as long as the outliers are a small minority of the nodes. There is also the computational cost of detecting several attackers, some methods, such as control theoretical approaches [15, 16, 18], are clearly more expensive to run in the context of multiple attackers, and may not be implementable in network environments such as low power wireless networks.

It is not clear whether this question can be addressed inside a formal framework. Consequently, this study addresses the issue of multiple attacks in a more empirical setting. We test implementations of defense strategies suitable to detect multiple attackers. More specifically, we have adapted the outlier methods in [5, 12] and reputation-based methods in [28] to the consensus-based NIDS described in [21]. We have left out for now control theoretic methods [15, 16, 18] which are very effective methods when dealing with a single attacker [19, 20], but their computational cost increases quite rapidly with the number of attackers. We focus on defense strategies that are relatively light from a computational standpoint. We analyze these methods in their capacity to detect multiple attacks as well as other criteria.

This paper is organized as follow. Section 2 briefly recalls the main features and the consensus algorithm of the system in [21]. Section 3 proposes an attack model, define issues related to independent and colluding attacks and finally describe the implementation of three detection strategies applicable to multiple attacks' scenarios. Section 4 report our experimental results and Section 5 concludes.

2 AVERAGE CONSENSUS AND NIDS

The network intrusion detection system proposed in [21] is a fully distributed NIDS which combines *anomaly-based* NIDS modules and an asymptotic average consensus algorithm. Assume we have a network as depicted in Figure 1 where red (sensor) nodes are NIDS modules. Each module observes the local traffic in sub-networks A, B, C and D using sensor capabilities. Each module is further capable of analyzing the observed traffic, and returns distributions about the likelihood that the observed local traffic is benign or anomalous.

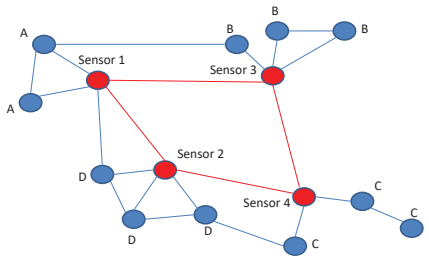

Figure 1: Network Intrusion Detection System.

Assume similar distributions have to be computed for the traffic in the whole network. One way is to send the local likelihood distributions to a designated module which computes their joint likelihoods and finally distributed the results downward to each other modules. In [21], this computation is performed distributively, each module running a local average procedure while exchanging local likelihoods with its neighbors.

2.1 Consensus algorithm

In Figure 1, together with the NIDS modules and the monitored network, a dedicated communication network composed of links $\{1,2\}$, $\{1,3\}$, $\{3,4\}$ and $\{2,4\}$ supports information sharing among NIDS modules. The dedicated links and the NIDS modules define an *NIDS network*. Mathematically, an NIDS network can be abstracted as a weighted adjacency matrix W, where $W_{ij} = 0$ when modules i and j are not adjacent in the network and $W_{ij} \neq 0$ is a weight associated to the link between i and j (in human consensus, W_{ij} represents the degree of trust i has for the opinion of j and vice-versa). To compute distributively network wide values, each module execute the following *consensus loop*:

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

where $x_i(t)$ is the *consensus value* of module i at iteration t of its consensus loop. At $t = 0$, $x_i(t)$ is initialized with some value computed by module i (see equation (3)). The set of neighbors to module i is denoted by $\mathcal{N}_i$, W_{ij} is the weight associated with edge (i, j). The consensus algorithm in [21] is an *asymptotic average consensus algorithm*, i.e. the consensus value of each module converges to $\frac{1}{n}\sum_{i=1}^{n} x_i(0)$ as $t \to \infty$. Convergence conditions are the following: 1- the graph corresponding to $W$is strongly connected; 2- the matrix W is stochastic, the weight of each row in W sum up to 1. The following weight matrix satisfies these two conditions:

$$W_{ij} = \begin{cases} \frac{1}{1+\max(d_i,d_j)} & \text{if } i \neq j \text{ and } j \in \mathcal{N}_i \\ 1 - \sum_{k \in \mathcal{N}_i} W_{ik} & \text{if } i = j \\ 0 & \text{if } i \neq j \text{ and } j \notin \mathcal{N}_i \end{cases} \tag{2}$$

where $d_i = |\mathcal{N}_i|$. It is the core weight matrix used in this paper.

2.2 Consensus-based NIDS

The consensus-based algorithm in [21] loops through four phases: 1- recording the feature values associated to local network traffic; 2- traffic analysis phase; 3- consensus phase where modules execute

the consensus loop in equation (1); 4- decision phase, where NIDS modules output a decision as to whether the network wide traffic is normal or anomalous and potentially take automatic mitigation actions against a perceived treat. Phases 1 and 2 process local traffic while phases 3 and 4 evaluate the network wide traffic activities.

2.2.1 Traffic analysis phase. Each NIDS module is an anomaly-based intrusion detection system, a naive Bayesian classifier computes the probability the observed feature values in phase 1 correspond to *anomalous* or *normal* traffic activities. Let m be the number feature values recorded in phase 1, o_j the value of feature j, h_a and h_n respectively the hypotheses that the network traffic is anomalous (h_a) and normal (h_n). $P(o_j|h)$ expresses the likelihood of the occurrence of value o_j under each hypothesis (given the historic anomalous h_a or normal h_n occurrences). Assuming conditional independence of the m features, the joint likelihood $P(O_i|h)$ of NIDS module i is the product of the feature's likelihoods:

$$P(O_i|h) = \prod_{j=1}^{m} P(o_j|h). \tag{3}$$

Module i returns $P(O_i|h)$.

2.2.2 Consensus phase. The consensus phase corresponds to the parallel execution of the n consensus loops, where n is the number of NIDS modules. The consensus phase lasts from the moment the first module initiates its consensus loop until the last module exits from its consensus loop. Mathematically, a consensus phase is expressed as

$$x(t+1) = Wx(t), \;\; t = 0, 1, \ldots \tag{4}$$

where $x(t)$ is a vector of n entries, $x_i(t)$ is the consensus value of module i at iteration t of its consensus loop. The consensus phase runs in lock step, iteration $t+1$ starts only once each module i has computed $x_i(t)$.

The consensus loop of a module i is constrained by the following stopping condition $|x_i(t+1) - x_i(t)| < \epsilon$, i.e. the consensus loop stops once the difference between the consensus value at iteration t and iteration $t+1$ is smaller than a pre-defined threshold ϵ. If the weight matrix satisfies the convergence conditions, all modules eventually meet this stopping condition, though for different values of t. We are interested to count the number of iterations of a consensus phase as the expression of how fast modules converge to an acceptable approximation of $\frac{1}{n}\sum_{i=1}^{n} x_i(0)$. Let t_i denotes the last iteration of consensus loop i (i.e. the iteration where module i stops). The number of iterations of a consensus phase is $c = \max\{t_i\}, i = 1..n$. This value is a measure of the convergence speed.

2.2.3 Decision phase. Decisions are computed independently by each module. At the end of a consensus phase, each module has approximated the network wide joint likelihood $P(O|h) = \prod_{i=1}^{n} P(O_i|h)$ through the distributed computation of the sum of the local likelihoods. The Bayes rule is applied to compute for module i an approximation $\approx P(h_a|O)_i$ of the probabilities $P(h_a|O)$ that the observed traffic is anomalous, similarly an approximation $\approx P(h_n|O)_i$ of the probabilities $P(h_n|O)$ is computed that the observed traffic is normal. The decision criterion to raise an alert is $\frac{\approx P(h_a|O)_i}{\approx P(h_n|O)_i} > \tau$ for some system pre-defined threshold value τ. This ratio is never completely the same from one module to another, however, if there is consensus, all modules will output the same decision.

3 DEFENSE STRATEGIES AGAINST MULTIPLE ATTACKERS

In general, attacks on a defense system like an NIDS aim at incapacitating the detection capabilities of the system. This can be achieved through external attacks disabling the system in part or completely, or internally, inducing the system to report no attack when intrusions take place, reporting attacks when none take place (false positives) or causing honest nodes to fail reaching consensus. Insider attacks origin from attackers capturing one or several NIDS modules. Attacks are executed by interfering with the consensus phase protocol. These attacks are difficult to detect as the system appear to function normally. According to [8, 15], insider attacks on the consensus phase can take the following forms:

(1) Data falsification attacks: The consensus loop is initialized with falsified data.
(2) Consensus disruption:
   (a) compromised nodes ignore the consensus value computed at each iteration and keeps transmitting the same constant c, so $x_i(t+1) = c$;
   (b) compromised nodes send to their neighbors falsified consensus values [15]. This last form of attacks can be modeled as in the following equation:

$$x_i(t+1) = W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t) + u_i(t). \tag{5}$$

In this paper, we focus on insider attacks of type 2.b (equation (5)). Compromised modules disturb the consensus loop by adding a value $u_i(t)$ to the consensus value $x_i(t+1)$. This form of attack aims to alter module's decisions in phase 4 of the consensus-based NIDS. In one scenario, attackers may increase the weight of the hypothesis h_n in the likelihood distribution at each consensus iteration of compromised modules. NIDS modules may converge to a network wide distribution where $\frac{\approx P(h_a|O)_i}{\approx P(h_n|O)_i} < \tau$, outputting no attack decisions when one is taking place, or causing honest modules to deliver different decisions.

3.1 Multiple attackers

Attacks executed by multiple attackers take essentially two forms, naive (independent) and colluded attacks [2]. Naive attacks occur when several modules are compromised by different actors. We model these attacks by setting $u_i(t)$ and $u_j(t)$ with different values if i and j are compromised modules and $i \neq j$. Undetected, such attacks are likely to prevent the consensus phase from converging or cause it to converge very slowly, but unlikely to be successful in falsifying module's decisions. In colluding attacks, we have $u_i(t) = u_j(t) > 0$ if $u_i(t)$ and $u_j(t)$ are compromised modules by colluding attackers, $i \neq j$. The value injected by $u_i(t)$ and $u_j(t)$ is consistent with altering modules' decisions.

3.2 Detecting multiple attackers

The bulk of recent defense methods against different forms of data falsifications, and consensus disruption in particular, origin from the wireless network community, particularly sensor and ad hoc

networks (see [6, 7, 27] and references therein). As these systems are physically decentralized, network intrusion detection systems are susceptible to insider attacks from existing network nodes becoming compromised or from nodes joining the network without sufficient credentials.

Among the defense strategies proposed in the recent literature, we focus primary on methods designed to handle multiple attacks that can be implemented in a fully distributed manner. We first consider a class of outlier defense strategies that specifically address data falsification and consensus disruption attacks as modeled in equation (5) (see [12] and references therein). Second, we propose a defense strategy inspired from attack detections in randomized consensus algorithms [3] (one consensus phase iteration in randomized consensus updates the state of only two nodes as follow: $x_i(t+1) = x_j(t+1) = \frac{x_i(t)+x_j(t)}{2}$). Randomized consensus algorithms rely on observing the consensus phase over several consensus iterations in order to detect anomalies. The strategy proposed in [3] is designed to handle multiple attackers, we extend it to the deterministic consensus algorithm in Section 2.1. Last, we consider detection of multiple attackers following reputation-based methods. These methods are widely use to prevent dysfunctionalities in wireless networks caused by malicious, faulty or selfish nodes. We analyze the reputation-based method in [28] which is specifically designed to handle consensus loop disruptions.

3.2.1 Method 1. Based on [12], this method is typical of a class of threshold based methods that have been applied to detect intrusion in consensus-based cooperative spectrum sensing applications. It is an outlier based method where a neighbor $j \in \mathcal{N}_i$ is an outlier if the consensus value $x_j(t)$ of node j at iteration t deviates too much from $x_i(t)$.

Neighbors $j \in \mathcal{N}_i$ are classified into two sets: neighbors $j \in \mathcal{N}_i^F$ if the deviation $|x_j(t)-x_i(t)| > \lambda_i(t)$ for some threshold $\lambda_i(t)$, $j \in \mathcal{N}_i^T$ if the deviation $|x_j(t)-x_i(t)| \leq \lambda_i(t)$. The rational for this approach is the following. The value $x_i(t)$ is the local average sum of $x_i(t-1)$ and $x_j(t-1)$ for $j \in \mathcal{N}_i$. As all nodes converge to $\frac{1}{n}\sum_{i=1}^n x_i(0)$, the difference $|x_j(t) - x_j(t-1)|$ is expected to become increasingly small. Therefore $x_j(t)$ been closed to $x_j(t-1)$ is also closed to $x_i(t)$ unless external inputs have been injected into $x_j(t)$. Values of neighbors with large deviations are suspected as not genuine. Neighbors in $\mathcal{N}_i^F$ are considered as potentially compromised, to avoid that consensus be falsified, the contribution of $x_j, j \in \mathcal{N}_i^F$ to the update of x_i is reduced by a factor a as in the following update equation:

$$x_i(t+1) = W_{ii}x_i(t) + \sum_{j\in\mathcal{N}_i^T} W_{ij}x_j(t) + \sum_{j\in\mathcal{N}_i^F} \frac{W_{ij}}{a}x_j(t) \quad (6)$$

Note that this update rule invalidates the second convergence condition of average consensus (row i in W should sum up to 1). The weight matrix in (2) is modified as follows:

$$W_{ij} = \begin{cases} \frac{1}{1+\max(d_i,d_j)} & \text{if } i \neq j \text{ and } j \in \mathcal{N}_i \\ 1 - \sum_{k\in\mathcal{N}_i^T} W_{ik} - \sum_{k\in\mathcal{N}_i^F} (\frac{W_{ik}}{a}) & \text{if } i = j \\ 0 & \text{if } i \neq j \text{ and } j \notin \mathcal{N}_i \end{cases} \quad (7)$$

Knowing the differences $x_i(t+1) - x_i(t)$ decrease as $x_i(t)$ converges to $\frac{1}{n}\sum_i^n x_i(0)$, attackers could inject increasingly smaller $u_i(t)$ into the consensus loop in order to avoid detection. In [12] a rule is provided to compute the threshold value λ_i at iteration t:

$$\lambda_i(t+1) = \frac{\sum_{j\in\mathcal{N}_i} |x_j(t+1) - x_i(t+1)|}{\sum_{j\in\mathcal{N}_i} |x_j(t) - x_i(t)|}\lambda_i(t). \quad (8)$$

This adaptive threshold rule avoids that attacks fall under the radar of the detection system.

3.2.2 Method 2. This second method is a variation on Method 1. Though the intuition is the same, this is a distant adaptation of "detection through spatial differences" proposed in [5]. Here detection is based on a sample of several consensus phase iterations $\mathcal{K}(t) = t_1, t_2, \ldots, t_k$ where $t_l \in \mathcal{K}(t) \leq t$, for t the current iteration of the consensus phase. The consensus iterations in $\mathcal{K}(t)$ are applied to compute the difference between x_i and the true average sum of its neighbors:

$$X_{ii} = \sum_{t_l\in\mathcal{K}(t)} \left(|x_i(t_l) - \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} x_j(t_l)| \right) \quad (9)$$

In order to properly interpret equation (9), we first note that if all consensus loops have converged to a true consensus, $x_i(t) \approx x_j(t) \forall j \neq i = \frac{1}{n}\sum_{i=1}^n x_i(0)$. Therefore, $x_{ii} = |x_i(t_l) - \frac{\sum_{j\in\mathcal{N}_i} x_j(t_l)}{|\mathcal{N}_i|}| \approx 0$. If the nodes have not converged yet, then $x_i(t) \neq x_j(t)$ for at least some $j \neq i$. The value x_{ii} measures the difference between $x_i(t)$ and the true average sum of its neighbors. If node j is compromised in the sense of the attack model in equation (5), $|x_j(t) - x_i(t)| > x_{ii}$, i.e. the difference between $x_j(t)$ and $x_i(t)$ is greater than the difference between $x_i(t)$ and the true average value of its neighbors, which includes node j. This is used to detect outliers. A short discretion is needed here. Remember $x_i(t) = W_{ii}x_i(t-1) + \sum_{j\in\mathcal{N}_i} W_{ij}x_j(t-1)$ as computed by the weight matrix W. If W satisfied the consensus convergence conditions $x_j(t) - x_j(t-1) < 0$, $|x_j(t) - x_j(t-1)| > 0$. If nodes have not converged yet, we should have $\frac{\sum_{j\in\mathcal{N}_i} x_j(t)}{|\mathcal{N}_i|} < \frac{\sum_{j\in\mathcal{N}_i} x_j(t-1)}{|\mathcal{N}_i|}$. Because x_{ii} is comparing values from two different consensus iterations, we should expect a difference between $x_i(t)$ and its neighbors due to the consensus step, though this is a minor issue.

The sum of the differences between $x_i(t_l)$ and $x_j(t_l), j \in \mathcal{N}_i$ over the sample space $t_l \in \mathcal{K}(t)$ is compared with X_{ii}:

$$J_{ji} = \sum_{t_l\in\mathcal{K}(t)} (|x_j(t_l) - x_i(t_l)|) > \lambda_k X_{ii} \quad (10)$$

In equation (10), if $J_{ji} > \lambda_k X_{ii}$, the deviation of x_j is greater than the average deviation $|\sum_{j\in\mathcal{N}_i} W_{ij}x_j(t-1) - \frac{\sum_{j\in\mathcal{N}_i} x_j(t)}{|\mathcal{N}_i|}|$. It is as if node j seeks to steer away nodes in $\mathcal{N}_i$ from their true average, it is possibly a compromised node in the NIDS network.

As for Method 1, neighbors of node i are partitioned into two sets. If neighbor j is classified as an outlier according to equation (10) then $j \in \mathcal{N}_i^F$, otherwise $j \in \mathcal{N}_i^T$. The weights of nodes in $\mathcal{N}_i^F$ are updated as in equation (6). Similarly, the entry ii of the weight matrix is updated as in equation (7).

3.2.3 Method 3. This is an adaptation of [28] to the consensus-based NIDS in [21]. The method in [28] detects consensus loop disruptions. This detection method assumes that module i can eavesdrop on the communications of one-hop neighbors, so it is only applicable in wireless networks.

Essentially, each node i computes redundantly the value $x^r_{ij}(t)$ of its neighbors by eavesdropping on the transmissions received by node j. The relevant transmissions received by j are $x_l(t-1)$, $l \in \mathcal{N}_j$. Therefore, for $j \in \mathcal{N}_i$

$$x^r_{ij}(t) = W_{jj}x_j(t-1) + \sum_{l \in \mathcal{N}_j} W_{jl}x_l(t-1) \tag{11}$$

Next, $x^r_{ij}(t)$ is compared with $x_j(t)$, the value sent to node i by node $j \in \mathcal{N}_i$: $x^r_{ij}(t) - x_j(t)$. Let $g_{ij}(t)$ be a variable that remembers the number of iterations from 0 to t where neighbor j has send a value x_j to node i that didn't differ by too much from x^r_{ij}. Essentially, $g_{ij}(t)$ records the number of iterations where node j has reported to node i a value that is believed to be a correct. The update rule of each variable g_{ij} is as follows:

$$g_{ij}(t) = \begin{cases} g_{ij}(t-1) + 1, & |x^r_{ij}(t) - x_j(t)| \leq \delta(t) \\ g_{ij}(t-1), & |x^r_{ij}(t) - x_j(t)| > \delta(t) \end{cases} \tag{12}$$

The reputation of node j from the perspective of node i is computed as follow:

$$rep_{ij}(t) = \frac{\eta g_{ij}(t)}{\eta t} \tag{13}$$

Here η is the "reputation coefficient" which is used to determine how fast the reputation decreases.

The consensus weight matrix is updated to reduce the influence of the nodes that have bad reputation.

$$W_{ij} = \min\left(\frac{rep_{ij}(t)}{\sum_{j \in \mathcal{N}_i} rep_{ij}(t)}, \frac{1}{|\mathcal{N}_i| + 1}\right) \tag{14}$$

As for the outlier method 1, the weights of the weight matrix are updated similarly as in equation (7) so that the sum of the values in each row = 1.

4 EXPERIMENTAL ANALYSIS

We have run simulations for 5 NIDS networks: two ring networks and two 2-D torus networks respectively each of size 9 and 25 NIDS modules, and one Petersen graph (10 modules and 15 links). A simulation consists for *each module* of a network to run 1000 times the loop describes at the beginning of Section 2.2: reads the local network traffic from an entry of the NSL-KDD data set, performs an Bayesian analysis of the local traffic, executes consensus loop during which one of the above three defense strategies is activated to detect whether some neighbors are compromised, last output a decision.

The consensus loop of each module i is implemented as follow. Analysis of the local network traffic returns two values: $P(O_i|h_a)$, the likelihood that the observed traffic at module i is anomalous; $P(O_i|h_n)$, the likelihood the observed traffic at module i is normal. These values are used to initialize the consensus loop of module i: $x^A_i(0) = \ln(P(O_i|h_a)$ and $x^N_i(0) = \ln(P(O_i|h_n)$, for $i = 1..n$. We have observed that consensus loops are initialized with values in the interval [-40,-175] as the values returned by Bayesian analysis

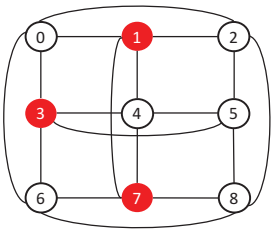

Figure 2: Torus 9 nodes.

are quite small. Each NIDS module i executes the following two components of the consensus loop until $|x^A_i(t+1) - x^A_i(t)| < \epsilon$ **and** $|x^N_i(t+1) - x^N_i(t)| < \epsilon$:

$$x^A_i(t+1) = W_{ii}x^A_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x^A_j(t) + u_i(t) \tag{15}$$

$$x^N_i(t+1) = W_{ii}x^N_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x^N_j(t) + u_i(t). \tag{16}$$

Once all consensus loops have converged, the corresponding consensus phase is completed, consequently each NIDS module i *decides* whether to raise an alert or not based on its ratio $\frac{x^A_i(t)}{x^N_i(t)}$ (which approximates the actual ratio $\frac{\sum_{i=1}^{N} \ln(p_{A_i})}{n} / \frac{\sum_{i=1}^{n} \ln(p_{N_i})}{n}$) and some predefined alert value λ_{NIDS}.

Tests in this paper are conducted with a slightly improved version of [21]. In this new version, consensus phases converge more rapidly and NIDS decisions are computed with a greater accuracy, closed to 100%, compared to $94 \approx 95\%$ in [21].

4.1 Attack implementations

Consensus disruptions from the attack model 2(b) are implemented as follow. Colluding attackers target a single module by having all the compromised modules to be neighbors of the same module. At each consensus phase, a module i is selected randomly to be the main victim of the colluding attack. The compromised neighbors of the victim are selected randomly. Figure 2 pictures this attack scenario where module 4 is the victim, surrounded by three compromised modules, modules 1, 3 and 7. At each iteration t of a consensus phase, each compromised module $j \in \mathcal{N}_i$ adds the same value $u_j(t) = -30$ to its consensus state $x_j(t)$ and send its falsified consensus value to module i. As each compromised module send falsified consensus values to all its neighbors, colluding attacks also disrupt the consensus loop of several other modules. For example, in the attack scenario of Figure 2, module 0 is fed with falsified consensus values from two compromised modules, module 2 is attacked by compromised module 1, etc. All together, a three modules colluding attack on a 9 modules Torus network is quite a comprehensive attack. In non-colluding attacks, at each consensus phase, k attackers are selected randomly among the n modules of the NIDS network. Each attacker j assigns randomly to $u_j(t)$ a value in the interval [-30, -40].

4.2 Parameter settings

This section describe briefly how parameters have been set. Our settings are quite basic in some cases, for example the attack values are

large in comparison with the initial readings. The purpose of this work is to compare different defense strategies on a same footing to see how well they can detect and neutralize multiple attackers. Some of the parameters would have been extremely difficult to fine tune in this context. One difficulty is the wide fluctuations in the initialization of the consensus loops (between -40 and -175 as seen already), which could cause the defense methods to flag some modules as compromised in the first few iterations of the consensus loop. Proper handling of this issue will require extra implementations or non-uniform fine tunings of the current implementations which is outside the scope of this work.

4.2.1 Attack settings. All attacks in our tests target the consensus loop component described in equation (15). An attack consist to assign $u_j^A(t) = -30$ or whatever value selected in non-colluding attacks. This attack makes $x_j^A(t)$ smaller, it aims at inducing the NIDS to report no attack when one occurs, reporting false negatives. The attack value -30 is relatively large with respect to the input values submitted to consensus phases, it is closed to the largest input values (-40). If not detected and neutralized, such attack definitely could impact the accuracy of the NIDS.

4.2.2 Method 1. The main parameters of Method 1 are λ_i and a. λ_i is a threshold value that classifies the neighbors of module i in two classes $\mathcal{N}_i^F$ (suspected of been attackers) and $\mathcal{N}_i^T$ (believed to be honest). The tests we reported are based on a single and constant value $\lambda = 30$. We have set λ to be large such that fluctuations in initial readings are not confused to often for attacks. Our test do not make use of the adaptive update of equation (8).

The second parameter a is the factor by which an entry in the weight matrix is reduced. An entry is reduced each time the corresponding module is identified as compromised by the method. We have set $a = 3.5$.

4.2.3 Method 2. We have two parameters, the size of the set $\mathcal{K}$ and the threshold constant λ_k. We have tested different values for $|\mathcal{K}| = 1, 2, 3$. We obtained more accurate NIDS decisions with larger values of $|\mathcal{K}|$, an indication that the intuition is correct, sampling over a larger set of iterations helps improve decisions. So $|\mathcal{K}| = 3$ in our tests. The threshold constant λ_k has been set to $\lambda_k = 1.3$. As for Method 1, this constant is used to classify neighbors of module i as belonging to $\mathcal{N}_i^F$ or to $\mathcal{N}_i^T$. If this constant is too high, it will fail to detect attacks, if too low modules with widely fluctuating initial readings will be classify falsely as compromised.

4.2.4 Method 3. This method has only one parameter, the threshold parameter $\delta(t)$, the other parameter η has been set to $\eta = 1$, essentially playing no role in this current set of tests. We have use a same and constant value for $\delta(t)$, i.e. $\delta(t) = 2.3$. In fact $|x_{ij}^r(t) - x_j(t)| = 0$ when both i and j are not compromised. However, rounding errors occur when the weight matrix W is updated in Method 3. The value $\delta(t)$ is used mainly to filter out these rounding errors.

4.3 Computational cost

Table 1 reports the computational overhead of each of the three defense strategies. The tests are executed while no attack take place. The column "Cost" reports the time in milliseconds for running the NIDS network simulation during 1000 iterations. In Table 1, rows "no detection" give the cost of running a NIDS network without the execution of any detection code. Rows "Method 1", "Method 2" and "Method 3" give the cost of running NIDS modules while also executing respectively the code of each detection method. Table 1 shows clearly there is a cost for protecting against Byzantine attacks. The higher running costs for the defense strategies compared to "no detection" for the same network size and topology reflect those costs. From Table 1 we gather that Method 1 and Method 3 have about the same overhead, while Method 2 is about 1.5 to 2 times more expensive than the two other methods.

Table 1: Overheads for running each method

Topologies	Sizes	Detection	Cost
Ring	9	no detection	2656
		Method 1	3095
		Method 2	3539
		Method 3	3111
	25	no detection	7526
		Method 1	8182
		Method 2	12152
		Method 3	12048
Torus	9	no detection	2566
		Method 1	3758
		Method 2	4403
		Method 3	3082
	25	no detection	6261
		Method 1	10943
		Method 2	20849
		Method 3	9213
Petersen	10	no detection	2731
		Method 1	3791
		Method 2	5318
		Method 3	3091

4.4 Accuracies and convergence speed

Table 2 summarizes the main numerical results of this study. It records data about 1- NIDS accuracy, i.e. how well the network intrusion detection system performs; 2- the defense methods accuracy, i.e. how accurately each method identifies compromised modules; 3- the convergence speed of the consensus phases.

NIDS accuracy is related to the decision phases of the honest modules. It is the ratio of accurate decisions over all decisions. In Table 2 this is recorded under the column NIDS for each method. For example, the value 0.94 in row 2 of the NIDS column for Method 1 specifies that, out of 1000 decisions, which is 94% accuracy for the ring network with 9 nodes and one attacker. Even though each module output an independent decision at the end of a consensus phase, honest modules output the same decision, therefore the set of decisions made by honest modules is recorded as one decision.

As defined in Section 2.2.2, convergence speed is a measure of the computational effort made by the honest modules to agree on a decision. In columns "Speed", Table 2 reports the average number of iterations performed by the consensus phases during the 1000

Table 2: NIDS accuracy, convergence speed and methods accuracy

Topologies	Sizes	Attack scenarios	Method 1			Method 2			Method 3		
			NIDS	Speed	False	NIDS	Speed	False	NIDS	Speed	False
Ring	9	no attacker	0.96	72	94/0	0.93	46	455/0	0.98	34	0/0
		1 attacker	0.94	101	53/6	0.93	81	538/164	0.98	67	0/1064
	25	no attacker	0.97	177	483/0	0.94	92	2528/0	0.99	114	0/0
		1 attacker	0.97	181	411/6	0.94	117	2841/234	0.99	129	0/6161
Torus	9	no attacker	0.97	22	30/0	0.96	95	1580/0	0.98	10	0/0
		1 attacker	0.96	33	26/12	0.94	133	961/530	0.98	12	0/174
		2 colluding attackers	0.96	41	22/43	0.92	132	317/855	0.98	16	0/471
		2 non-colluding attackers	0.96	34	32/36	0.92	145	380/795	0.98	17	0/461
		3 colluding attackers	0.94	60	19/143	0.84	620	798/4567	0.98	30	0/1201
		3 non-colluding attackers	0.94	42	21/72	0.89	151	111/1218	0.98	23	0/893
	25	no attacker	0.99	68	283/0	0.97	219	9881/0	0.99	30	20/0
		1 attacker	0.99	72	258/11	0.97	213	7846/855	0.99	33	18/1502
		2 colluding attackers	0.99	84	265/30	0.96	224	6832/1862	0.99	37	16/3365
		2 non-colluding attackers	0.99	80	229/18	0.97	228	6591/1707	0.99	35	16/3216
		3 colluding attackers	0.98	95	255/71	0.94	271	6811/3523	0.99	43	15/5780
		3 non-colluding attackers	0.99	85	260/68	0.95	235	5180/2532	0.99	39	15/5202
Petersen	10	no attacker	0.98	27	46/0	0.95	142	1924/0	0.99	12	0/0
		1 attacker	0.97	40	36/9	0.94	142	1137/424	0.99	17	0/292
		2 colluding attackers	0.96	59	35/18	0.92	203	811/1339	0.99	29	0/927
		2 non-colluding attackers	0.97	50	31/14	0.93	181	754/929	0.99	24	0/788

iterations of a simulation. For example, the value 101 in row 2 of the Speed column for Method 1, means that, on average, a consensus phase executed 101 iterations for the ring network with 9 nodes and one attacker. Convergence speed is impacted obviously by attackers which disrupt the consensus loop of honest modules. Measurements reported in the columns "Speed" are an indication of how well each defense method is coping with attacks. Hidden in those measurements is a cost associated to modify the weight matrix each time a module is suspected of been compromised, this also has an impact on consensus convergence speed [20].

Last is the accuracy of the defense methods. This has been recorded as the number of times a module i has been identified as compromised when it was not (false positives) or was not identified as compromised when it was (false negatives). Under columns "False", these values are entered as (false positives)/(false negatives). False positives are computed as follow (false negatives are computed similarly): consider row 1 of ring 9 where the consensus phase last on average 72 iterations. A module i in a ring has two neighbors, therefore at each iteration of a consensus phase it could be falsely identified as compromised twice. So totally $10000 \times 72 \times 9 \times 2 = 1,296,000$ is the number of opportunities that modules can be labeled falsely as compromised. Table 2 reports 94 false positives for ring 9 nodes, which is 0.00072530864% of false positives. Similarly, for Method 2, Torus 25 nodes, 3 colluding attackers, there are $1000 \times 271 \times 25 \times 4 = 27100000$ opportunities that modules can be labeled falsely as compromised. The total number of false positive is 6811, which is 0.25132841328% of the opportunities , less than 1%. From our analysis, these numbers are too insignificant to impact NIDS accuracy. However, each false positive causes an entry of the weight matrix to be modified, which impacts the convergence speed of the consensus phase. The convergence speeds between Method 1 and Method 3 in row 1 are not the same, though everything else is equal. Method 1 has 94 false positives while Method 3 has none.

The types of attacks in Table 2 are described under the "Attack scenarios"' column. The "no attacker" label corresponds to simulations where all the NIDS modules are honest. The label "one attacker" refers to simulations where, at each consensus phase, one NIDS module is selected randomly to be a compromised module. The "two colluding attackers" label refers to simulations where, at each consensus phase, one NIDS module is selected randomly to be a victim and two of its neighbors are selected randomly to be compromised. Attack scenarios labeled "two non-colluding attackers" correspond to simulations where, at each consensus phase, two NIDS modules are selected randomly to be compromised modules. Attack scenarios "three colluding attackers" and "three non-colluding attackers" are same as the two previous attack scenarios except there are three attackers. We note that the range of attack scenarios differ among the network topologies. For example, Table 2 display results only for one attacker for ring networks. Based on results in [10, 17] and others, beyond a certain number of attackers, the Byzantine agreement problem has no solution. Ring networks cannot be defended against Byzantine attacks if the number of attackers is greater than one. Assume modules 1 and 4 in Figure 1 were compromised, the weight of edges adjacent to these two modules will decrease quickly, effectively disconnecting the network (the convergence condition 1 in Section 2.1 is no longer satisfied), which prevents honest modules 2 and 3 from exchanging information and making consensus . For the regular graphs (networks) tested in this work, they get disconnected if the number of attackers is greater or equal to the degree of the nodes in the graph. The degree of nodes in ring, 2-D torus and Petersen graphs is respectively two, four and

three. Table 2 reports results of attack scenarios of up to 3 colluding and non-colluding attackers, depending on the connectivity of each network topology.

Overall we observe from Table 2 that Method 3 dominates the two other methods both in terms of accuracy and convergence speed. Ring networks converge more slowly, this is expected as the diameter of rings $n/2$, is greater than for 2-D torus $n^{\frac{1}{2}}$ and Petersen graph 2. It takes more iterations for information to diffuse across all the modules of a ring. Attacks impact accuracy for Method 1 and Method 2. For all methods, convergence is slower under attacks, the larger the number of compromised modules, the slower the convergence. For all methods, it is easier to deal with non-colluding attackers. The three defense methods are potentially useable in practice to defend against multiple attackers, which is probably the most significant and "unexpected" conclusion from this study.

5 CONCLUSION

The object of this study was to test whether proposed defense strategies in the literature against multiple Byzantine attacks work in practice. We have implemented and tested variations of three defense strategies, two inspired from outlier detection techniques and one from reputation-based techniques. These implementations protected consensus seeking modules in a fully distributed network intrusion detection system. Our experimental results indicate that all three are promising approaches to defend in practice against multiple Byzantine attacks, this at a reasonable computational cost.

For future work, this study will be made more comprehensive by including control theoretical defense methods and potentially other outlier and reputation-based methods. More extensive attack scenarios will be considered to analyze the difficulty of fine tuning parameters and the capacity of the different methods to response to versatile attack scenarios.

ACKNOWLEDGMENTS

Funding for this project comes from the Professorship Start-Up Support Grant VGU-PSSG-02 of the Vietnamese-German University. The authors thank this institution for supporting this research.

REFERENCES

[1] J. A. Benediktsson and P. H. Swain. 1992. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics* 22, 4 (Jul 1992), 688–704. https://doi.org/10.1109/21.156582

[2] S. Bhattacharjee, R. Rajkumari, and N. Marchang. 2015. Effect of colluding attack in collaborative spectrum sensing. In *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*. 223–227. https://doi.org/10.1109/SPIN.2015.7095266

[3] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. 2006. Randomized Gossip Algorithms. *IEEE/ACM Trans. Netw.* 14, SI (June 2006), 2508–2530. https://doi.org/10.1109/TIT.2006.874516

[4] Morris H. Degroot. 1974. Reaching a consensus. *J. Amer. Statist. Assoc.* 69, 345 (1974), 118–121. http://www.jstor.org/stable/2285509

[5] R. Gentz, S. X. Wu, H. T. Wai, A. Scaglione, and A. Leshem. 2016. Data Injection Attacks in Randomized Gossiping. *IEEE Transactions on Signal and Information Processing over Networks* 2, 4 (Dec 2016), 523–538. https://doi.org/10.1109/TSIPN.2016.2614898

[6] Guangjie Han, Jinfang Jiang, Lei Shu, Jianwei Niu, and Han-Chieh Chao. 2014. Management and applications of trust in Wireless Sensor Networks: A survey. *J. Comput. System Sci.* 80, 3 (2014), 602 – 617. https://doi.org/10.1016/j.jcss.2013.06.014 Special Issue on Wireless Network Intrusion.

[7] Vittorio P. Illiano and Emil C. Lupu. 2015. Detecting Malicious Data Injections in Wireless Sensor Networks: A Survey. *ACM Comput. Surv.* 48, 2, Article 24 (Oct. 2015), 33 pages. https://doi.org/10.1145/2818184

[8] B. Kailkhura, S. Brahma, and P. K. Varshney. 2017. Data Falsification Attacks on Consensus-Based Detection Systems. *IEEE Transactions on Signal and Information Processing over Networks* 3, 1 (March 2017), 145–158. https://doi.org/10.1109/TSIPN.2016.2607119

[9] Sumit Kar, Srinivas Sethi, and Manmath Kumar Bhuyan. 2016. Security Challenges in Cognitive Radio Network and Defending Against Byzantine Attack: A Survey. *Int. J. Commun. Netw. Distrib. Syst.* 17, 2 (Jan. 2016), 120–146. https://doi.org/10.1504/IJCNDS.2016.079098

[10] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (July 1982), 382–401. https://doi.org/10.1145/357172.357176

[11] Shancang Li, George Oikonomou, Theo Tryfonas, Thomas Chen, and Li Xu. 2014. A distributed consensus algorithm for decision-making in service-oriented Internet of Things. *Transactions on Industrial Informatics* 10, 2 (2014), 1461–1468. https://doi.org/10.1109/TII.2014.2306331

[12] Sheng Liu, Haojin Zhu, Shuai Li, Xu Li, Cailian Chen, and Xinping Guan. 2012. 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*. 603–608. https://doi.org/10.1109/GLOCOM.2012.6503179

[13] Nancy A. Lynch. 1996. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[14] R. Olfati-Saber, J. A. Fax, and R. M. Murray. 2007. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* 95, 1 (Jan 2007), 215–233. https://doi.org/10.1109/JPROC.2006.887293

[15] F. Pasqualetti, A. Bicchi, and F. Bullo. 2007. Distributed intrusion detection for secure consensus computations. In *Decision and Control, 2007 46th IEEE Conference on*. 5594–5599. https://doi.org/10.1109/CDC.2007.4434297

[16] F. Pasqualetti, A. Bicchi, and F. Bullo. 2012. Consensus Computation in Unreliable Networks: A System Theoretic Approach. *IEEE Trans. Automat. Control* 57, 1 (Jan. 2012), 90 – 104.

[17] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (April 1980), 228–234. https://doi.org/10.1145/322186.322188

[18] S. Sundaram and C. N. Hadjicostis. 2011. Distributed Function Calculation via Linear Iterative Strategies in the Presence of Malicious Agents. *IEEE Trans. Automat. Control* 56, 7 (July 2011), 1495–1508. https://doi.org/10.1109/TAC.2010.2088690

[19] Michel Toulouse, Hai Le, Cao Vien Phung, and Denis Hock. 2016. Robust Consensus-based Network Intrusion Detection in Presence of Byzantine Attacks. In *Proceedings of the Seventh Symposium on Information and Communication Technology (SoICT '16)*. ACM, New York, NY, USA, 278–285. https://doi.org/10.1145/3011077.3011121

[20] Michel Toulouse, Hai Le, Cao Vien Phung, and Denis Hock. 2017. Defense Strategies against Byzantine Attacks in a Consensus-Based Network Intrusion Detection System. *Informatica, An International Journal of Computing and Informatics* 41, 2 (2017), 193–207.

[21] Michel Toulouse, Bui Quang Minh, and Philip Curtis. 2015. A Consensus Based Network Intrusion Detection System. In *IT Convergence and Security (ICITCS), 2015 5th International Conference on*. IEEE, 1–6. http://dblp.uni-trier.de/db/conf/icitcs/icitcs2015.html#ToulouseMC15

[22] F. Tschorsch and B. Scheuermann. 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys Tutorials* 18, 3 (thirdquarter 2016), 2084–2123. https://doi.org/10.1109/COMST.2016.2535718

[23] J. Tsitsiklis, D. Bertsekas, and M. Athans. 1986. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *Automatic Control, IEEE Transactions on* 31, 9 (Sept. 1986), 803–812.

[24] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. 1995. Novel Type of Phase Transition in a System of Self-Driven Particles. *Phys. Rev. Lett.* 75 (Aug 1995), 1226–1229. Issue 6. https://doi.org/10.1103/PhysRevLett.75.1226

[25] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. 2007. Distributed average consensus with least-mean-square deviation. *J. Parallel and Distrib. Comput.* 67, 1 (2007), 33 – 46. https://doi.org/10.1016/j.jpdc.2006.08.010

[26] Qiben Yan, Ming Li, Tingting Jiang, Wenjing Lou, and Y Thomas Hou. 2012. Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, 900–908.

[27] Yanli Yu, Keqiu Li, Wanlei Zhou, and Ping Li. 2012. Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures. *Journal of Network and Computer Applications* 35, 3 (2012), 867 – 880. https://doi.org/10.1016/j.jnca.2011.03.005 Special Issue on Trusted Computing and Communications.

[28] Wente Zeng and Mo-Yuen Chow. 2014. A Reputation-Based Secure Distributed Control Methodology in D-NCS. *IEEE Trans. Industrial Electronics* 61, 11 (2014), 6294–6303. http://dblp.uni-trier.de/db/journals/tie/tie61.html#ZengC14

[29] Linyuan Zhang, Guoru Ding, Qihui Wu, and Fei Song. 2016. Defending Against Byzantine Attack in Cooperative Spectrum Sensing: Defense Reference and Performance Analysis. *IEEE Access* 4 (2016), 4011–4024. https://doi.org/10.1109/ACCESS.2016.2593952