A general discrete-time method to achieve resilience in consensus algorithms

Guilherme Ramos, Daniel Silvestre, Carlos Silvestre

Abstract— **In this paper, we approach the problem of a set of network agents reaching resilient consensus in the presence of a subset of attacked nodes. We devise a generalized method, with polynomial time complexity, which receives as input a discrete-time, synchronous-communication consensus algorithm, a dynamic network of agents, and the maximum number of attacked nodes. The distributed algorithm enables each normal node to detect and discard the values of the attacked agents while reaching the consensus of normal agents for the input consensus algorithm. Hence, the proposed method adds an extra layer of resilience to a given discrete-time and synchronous-communication consensus algorithm. Finally, we demonstrate the effectiveness of the method with experimental results, showing some attack circumstances which we can counter, where the state-of-the-art methods fail.**

I. Introduction

The study of cybersecurity in networked control systems (NCS) is of utmost importance. We face a change of paradigm with the IoT (Internet of Things) with networked control systems often being connected to the Internet. This setting opens the door to malicious attacks that can drive systems to dangerous states, translating to physical faults or serious accidents. An example of NCSs is the collective agreement among a set of agents in a network (regarding temperature, computer loads, power generation, to name a few) referred to as consensus problems.

Consensus algorithms (CAs) [1] exchange messages in order to have agents agreeing on an outcome in a distributed way. Moreover, this is a problem arising in diverse areas with examples ranging from: distributed optimization [2], [3]; motion coordination tasks like flocking, leader following [4]; rendezvous problems [5]; and resource allocation in computer networks [6]. Thus, the problem of consensus is the common denominator of crucial applications, such as the development of Distributed Kalman Filters to estimate the motion of a target in 2D, see [7].

Resilient algorithms for multi-agent systems have been developed in the literature and can be categorized into two main paths. The first one is fault detection and isolation, i.e., normal agents detect and isolate attacked nodes to reach consensus. The second one tries to obtain consensus ignoring suspicious agents that may or may not be attacked nodes.

A fault-tolerant algorithm for Byzantine consensus in asynchronous networks is proposed in [8], [9]. The method uses a less restrictive topological condition, and it can cope with synchronous networks, delay in the network communication, and time-varying graph networks.

In related work, authors have added a detection overlay to attain resilience in the CA. In [10], a gossip algorithm capable of dealing with worst-case and stochastic faults is developed. The approach may also be used to reach resilient consensus on a value in the intersection of the estimates that each node keeps for the other agents [11]. In [12], the work is extended to a larger family of gossip algorithms. These systems have fast convergence, but it suffers from an exponential time complexity in the isolation of attackers, contrasting with the proposed polynomial-time method.

To deal with misbehaving agents, the work of [13] sought two parameter-independent fault-tolerant CAs: (i) adaptively estimates the number of faulty agents which, in the presence of f faulty nodes, converges when the network of non-faulty agents is $(f+1)$-robust; (ii) a non-parametric method that converges if the network of non-faulty nodes is $(f+1)$-robust and all normal nodes hold the same number of in-neighbors. In [14], the vulnerabilities of consensus-based distributed optimization protocols when facing nodes deviating from the designated update rule and the performance limitations of any distributed optimization algorithm in the presence of adversaries are studied. With the notion of maximal f-local sets of graphs for cases where there are at most f attacked nodes in every agent neighborhood, lower bounds on the distance-to-optimality of feasible solutions are given.

To be robust to misbehaving agents, [15] presents a second-order sampled-data CA where: (i) each normal agent updates its state utilizing local information; (ii) malicious nodes may execute their updates arbitrarily. The authors detail a resilient CA, assuming that the (unknown to normal agents) network is sufficiently connected and that each normal node knows the maximum number of malicious agents. To avoid the effect of attacked nodes, each normal agent ignores neighbors having large and small position values. The work in [16] extends those mentioned above, yielding a clock synchronization CA for wireless sensor networks.

Similarly, in [17], the author allows each node in a network to ignore a possibly distinct number of extreme neighbors and show graph conditions that ensure network consensus. In [18], the authors introduced a resilient leader-follower consensus to arbitrary reference values. Under the same idea, where an agent ignores a number of the largest and the smallest received values, their work assures a consensus value belonging to the convex hull of initial nodes states. In this line, [19] designs a resilient CA for time-varying networks of dynamic agents; [20] deals with the case of quantized transmissions. In [21], a consensus+innovations estimator where each agent thresholds the gain of its innovations term is proposed, with polynomial convergence rate when there

G. Ramos (gramos@fe.up.pt) is with Dep. of Electrical and Computer Engineering, Faculty of Engineering, University of Porto, Portugal. He acknowledges the support of Institute for Systems and Robotics (ISR), Instituto Superior Técnico, University of Lisbon, Portugal, through scholarship BL112/2019. D. Silvestre and C. Silvestre are with the Dep. of Electrical and Computer Engineering, Faculty of Science and Technology, University of Macau, China. D. Silvestre is also with ISR. C. Silvestre is on leave from Instituto Superior Técnico/University of Lisbon, Portugal. This work was partially supported by project MYRG2018-00198-FST of the University of Macau, by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through ISR, under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020 and by FCT project POCI-01-0145-FEDER-031411-HARMONY.

are less than half of the attacked nodes.

Usually, resilient consensus approaches let each agent discard a set of larger and smaller neighbor values. Thus, sending a value in a specific range may still affect the consensus.

In this work, we say that an *attacked node* or *attacked agent* in a consensus network is an agent that tries to drive the regular agents' states to a desired, by the attacker, value. We also refer to such a node as a *faulty agent* or *faulty node*.

Main contributions: For discrete-time CAs with synchronous communication, and dynamic networks, we: (i) introduce a general distributed algorithm that allows each normal agent to identify, in polynomial-time, whenever there exist attackers in the network; (ii) we propose a general polynomial-time distributed method that receives as inputs a CA and the maximum number of attacked agents and is capable of identifying and ignoring the attacked agents' values. Lastly, we study the computational complexity of the presented methods and prove their soundness and completeness. In summary, we propose general methods that receive as input a discrete-time CA with synchronous communication and add an extra layer of resilience to the CA. One of the proposed methods aims to detect the existence of attacked nodes, and the other method aims to both detect attacked agents and correct the normal agents' states, disregarding the attacked agents' shared values.

Preliminaries and Notation: We denote by $\mathbb{N}$ the set of positive integers and by $\mathbb{N}_0$ the set of non-negative ones. Next, we revise graph theory concepts [22]. A *digraph* $\mathcal{G}$ is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $n > 1$ *nodes*, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of *edges*. Edges are ordered pairs representing an accessibility relationship between nodes. If $u, v \in \mathcal{V}$ and $(u, v) \in \mathcal{E}$, then node v directly accesses information shared by node u. We also call the digraph by *network* and the nodes by *agents*. A digraph is a *complete digraph* when each node can directly access information of all the other nodes. Let $v \in \mathcal{V}$, we define the *neighbors* of v as $\mathcal{N}_v = \{v\} \cup \{u : (u, v) \in \mathcal{E}\}$, and they are the set of nodes that v can directly access information. The *in-degree* of a node $v \in \mathcal{V}$ is $d_v = |\mathcal{N}_v|$, i.e., it is the number of neighbors of v. The *out-degree* of a node $v \in \mathcal{V}$ is $o_v = |\{u : v \in \mathcal{N}_u\}|$. Given a digraph $\mathcal{G}$, we define a *path* as a sequence of nodes $(v_1, v_2, \dots v_k)$ with $(v_i, v_{i+1}) \in \mathcal{E}$, for all $i = 1, \dots, k-1$. $\mathcal{G}$ is *strongly connected* if for any nodes $u, v \in \mathcal{V}$ there exists a path from u to v. A helpful way to describe a digraph is by its adjacency matrix, $A \in \mathbb{R}^{n \times n}$. For a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $A_{u,v} = 1$ if $(u, v) \in \mathcal{E}$, and $A_{u,v} = 0$, otherwise. A *subgraph/subnetwork*, $\mathcal{H} = (\mathcal{V}', \mathcal{E}')$, of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a digraph with $\mathcal{V}' \subset \mathcal{V}$ and $\mathcal{E}' \subset \mathcal{E}$. Let $\mathcal{A} \subset \mathcal{V}$, we define $\mathcal{H} = \mathcal{G} \setminus \mathcal{A}$ as the subgraph of $\mathcal{G}$, with $\mathcal{H} = (\mathcal{V} \setminus \mathcal{A}, \mathcal{E}')$, and $\mathcal{E}' = \{e \in \mathcal{E} : e = (u, v) \text{ and } u, v \notin \mathcal{A}\}$.

From this point on, we use the discrete-time variable $t \in \mathbb{N}_0$. Given a sequence of values $\{s^{(t)}\}_{t \in \mathbb{N}_0}$ or a function $f : \mathbb{R} \to \mathbb{R}$, if the sequence or the function has limit, i.e., $\lim_{t\to\infty} s^{(t)} = a$ or $\lim_{t\to\infty} f(t) = b$, then we write compactly that $s^{(t)} \to a$ or $f(t) \to b$. For a set $\mathcal{S} \subset \mathbb{N}$, we define its subsets by $\wp(\mathcal{S})$. We identify all permutations of a set in $\wp(\mathcal{S})$ with its sorted version, e.g., we identify $\{7, 2, 5\}$ with $\{2, 5, 7\}$. For example, if $\mathcal{S} = \{1, 2, 3\}$, then $\wp(\mathcal{S}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. We denote the subsets of $\mathcal{S}$ of size $i \leq |\mathcal{S}|$ by $\wp(\mathcal{S}, i) = \{w \in \wp(\mathcal{S}) : |w| = i\}$. For $v \in \mathbb{R}^n$, we denote its i-th entry by $v[i]$ for $i \in \{1, \dots, n\}$. When useful, we index vector positions by keys (as to define dictionaries in computer science), e.g. for $v \in \mathbb{R}^8$ and for the above $\wp(\mathcal{S})$, we may index the entries of v by the elements of $\wp(\mathcal{S})$, e.g. $v[\emptyset]$ or $v[\{2, 3\}]$.

Finally, we use the standard universal and existential quantifiers, $\forall$ and $\exists$. Also, we use the "exists one and only one" quantifier, $\exists!_x.\varphi(x) \equiv \exists_x.\varphi(x) \wedge \forall_{y \neq x}.\neg\varphi(y)$.

II. Problem Statement

In this work, we are interested in the case where an attacker (malicious entity) has a particular goal (not only trying to prevent consensus convergence). To this end, an attacker wants to deviate the consensus of a network to a specific value a that may be harmful to the system. Remark that the attacker may actuate agents to transmit distinct values to several neighbors, it may act as a *Byzantine* node, as much as the values are converging asymptotically to a. Let the set of attacked nodes of network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be denoted by $\mathcal{A}$, with $\mathcal{A} \subset \mathcal{V}$. The goal is to create an algorithm that receives as input a CA, a network of agents and the maximum number of attacked agents and allows normal agents to identify the attacked nodes, ignoring their values in the final agreement. In other words, the objective is to devise a method that adds an extra layer of resilience to a given discrete-time and synchronous-communication consensus algorithm.

In what follows, we denote by C a discrete-time and synchronous communication CA. C receives as input a (possibly dynamic) digraph $\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{E}^{(t)})$ and the number of time steps to run the CA $N \in \mathbb{N}$. C outputs a vector with size $|\mathcal{V}|$ with the state that each node arrived after N time-steps.

Problem 1: Let C be a CA to run for $N \in \mathbb{N}$ time steps, a network of agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{A} \subset \mathcal{V}$ a set of attacked nodes such that if $u \in \mathcal{A}$ then $x_u^{(t)} \to a$ (a is unknown for the normal nodes). The goal is that normal nodes, $v \in \mathcal{V} \setminus \mathcal{A}$: 1) identify the presence of attacked nodes, i.e. if $|\mathcal{A}| > 0$; 2) compute the consensus resulting from C for N time steps considering the subnetwork without attacked agents, $\mathcal{G} \setminus \mathcal{A}$.

By detection of an attack, we mean that a normal agent is capable of identifying the existence of attacked agents (but they may not know which ones are the attacked agents). In this paper, we assume that we deal with attacks that can be detectable. By detectable, we mean that we do not consider the possibility of having an initial set of agents' states without attacked nodes and another initial set of agents' states with attacked nodes, that only change their initial states, yielding the same execution trace of the CA. To approach Problem 1, we require further assumptions.

Assumption 1: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a digraph, $\mathcal{A}$ a set of attacked agents and C a CA and $\mathsf{C}(\mathcal{G}, N)$ be the consensus value of nodes $\mathcal{V} \setminus \mathcal{A}$ resulting from applying C for $N \in \mathbb{N}$ time steps. Let $\varepsilon > 0$ denote the precision utilized for computations. For all $v \in \mathcal{V} \setminus \mathcal{A}$ and for all $u \in \mathcal{V}$ it follows that $\lim_{t\to\infty} |\mathsf{C}(\mathcal{G} \setminus \{u\}, t) - x_v^{(t)}| > \varepsilon$. ◇

Assumption 1 only requires that no agent has a state equal to the consensus of the subnetwork excluding that agent.

Assumption 2: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a digraph, $\mathcal{A}$ a set of attacked agents and C a CA. The CA C converges for every subgraph $\mathcal{H}$ of $\mathcal{G}$ with $\mathcal{H} = \mathcal{G} \setminus \mathcal{V}'$, where $|\mathcal{V}'| \leq |\mathcal{A}|$. ◇

Assumption 2 is a more general assumption. Essentially, we require the network without attacked agents to be connected in order to reach consensus. A usual and more restrictive robustness assumption is, for example, the one presented in [23]. The authors define more restrictive notions of robustness called r-robustness and (r, s)-robustness, which sometimes are computationally hard to verify. Further, Assumption 2 is also fair since we expect that C converges for the network without attacked nodes. From now on, we work under the Assumptions 1 and 2.

Our goal is to develop an algorithm ensuring that $\forall_{u\in\mathcal{V}\setminus\mathcal{A}} \lim_{t\to\infty} x_u(t) = x_\infty$ for the value x_∞ that depends on the underlying CA, denoting the consensus state. Also, the problem should be solved distributedly resorting only to the network dynamics information and the state values obtained using the CA at each point in time.

III. On the f-Resilience of Consensus Algorithms

Now, we present an algorithm to solve Problem 1. The current proposal allows each normal agent to detect all the attacked agents and to correct the consensus value.

The intuition of the algorithm that we formalize next is that each agent receives a value from its neighbors and updates an internal vector. The first entry of the vector is the result of computing an additional step of the CA C with the information received from all its neighbors. The subsequent entries correspond to computing one iteration of algorithm C discarding the values received for each possible subset of nodes, using the dictionary order, up to subsets of size f. After this update to its internal vector state, the node selects its next state to be the entry of that vector that it decides to correspond to the consensus state that excludes the attacked nodes if its internal vector already allows to pinpoint the attack agents. In the initial time steps, it cannot discern if there are attacked nodes and selects the first entry to be the next value it shares to its neighbors. For f maximum number of possible attacked agents, $|\mathcal{A}| \leq f$, and a computational precision $\varepsilon > 0$, the algorithm goes as follows.

Alg. *1:* Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a network of agents, C be a CA, and $N \in \mathbb{N}$ be the time steps for which we run C. Given $f \in \mathbb{N}_0$ and $\varepsilon > 0$, let $\mathcal{F} = \bigcup_{i=0}^{f} \wp(\mathcal{V}, i)$ be the set of all subsets of agents with sizes from 0 to f. For each time step $t \in \mathbb{N}$, the algorithm has the following three steps:

1. **(State Vector Initialization)** Each agent v sets $c_v^{(0)}[\mathcal{F}[i]] = x_v^{(0)}$;
2. **(State Vector Update)** For each time instance, $t \in \mathbb{N}$, each agent $u \in \mathcal{V}$ communicates a vector $c_u^{(t)}$ with size $|\mathcal{F}|$, where at the i-th entry $c_u^{(t)}[\mathcal{F}[i]]$ is the result of applying algorithm C for agent u for one iteration, utilizing the i-th entries of the last received vectors $c_v^{(t-1)}$, $v \in \mathcal{N}_u$, excluding the nodes in $\mathcal{N}_u$ that are in the $\mathcal{F}[i]$;
3. **(Selection of the Consensus State Among the State Vector Values)** For each $t \leq N$ every normal agent $u \in \mathcal{V} \setminus \mathcal{A}$ selects $x_u^{(t)}$ to be the j-th entry of $c_u^{(t)}$, $x_u^{(t)} = c_u^{(t)}[j]$, where j is given as

(i) if $\underset{\mathcal{S}\in\mathcal{F},\mathcal{S}\neq\emptyset}{\forall} \left| c_u^{(t)}[\emptyset] - c_u^{(t)}[\mathcal{S}] \right| \geq \varepsilon$, then $j = 1$, i.e. agent u considers the information of all neighbors;

(ii) else

$$\begin{aligned} j = \quad & \underset{1<j'\leq|\mathcal{F}|}{\arg\min} \; |\mathcal{F}[j']| \\ \text{s.t.} \quad & \mathcal{F}[j'] \in \wp(\mathcal{V}, i) \\ & \left| c_u^{(t)}[\emptyset] - c_u^{(t)}[\mathcal{F}[j']] \right| < \varepsilon \\ & \underset{\mathcal{S}\in\wp(\mathcal{V},i),\mathcal{S}\neq\mathcal{F}[j']}{\forall} \left| c_u^{(t)}[\emptyset] - c_u^{(t)}[\mathcal{S}] \right| \geq \varepsilon. \end{aligned}$$

That is, agent u identifies $\mathcal{F}[j']$ as the smallest set (with i attacked agents, $\mathcal{F}[j'] \in \wp(\mathcal{V}, i)$) that contains all the attacked nodes and selects the state as the entry of $c_u^{(t)}$ which discards the information of the agents in $\mathcal{F}[j]$; ●

In step 1., each agent $v \in \mathcal{V}$ sets all the entries of its initial vector state to be its initial value $x_v^{(0)}$. In step 2., the agents communicate their vector states to their neighbors, and they update their vector states following one step of the CA C for each entry of the received vector states. This update corresponds to refreshing the consensus state (each entry of the state vector) for each subnetwork of agents that excludes, at most, f of the network nodes. Intuitively, in step 3.(i) of Alg. 1, each normal agent verifies if its state vector has different values when considering distinct subsets of agents. If so, it decides that there are not attacked agents and considers all neighbors information. Otherwise, in step 3.(ii) of Alg. 1, each normal agent searches for the set of agents that does not contain attacked nodes. To do so, it searches its vector state and, for each entry of the vector corresponding to a number of discarded agents, it checks if there is a unique value different from the remaining ones and from the value that corresponds the entire network information.

To illustrate the algorithm, we consider a discrete-time and synchronous communication CA, considering the complete digraph with three agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, 3\}$ and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V} \text{ and } i \neq j\}$. Additionally, suppose that $f = 1$ and that agent 3 is the attacked agent, i.e. $\mathcal{A} = \{3\}$, and $\varepsilon = 0.05$. For illustrative purposes, agent 3 always share the same value, i.e., $x_3(t) = 0.1$ and $c_3[F[j]] = 0.1$, for any $t \in \mathbb{N}$ and for any $j = 1, \ldots, \sum_{i=0}^{f} \wp(\mathcal{V}, i)$. Therefore, the set $\mathcal{F}$ is $\mathcal{F} = \{\emptyset, \{1\}, \{2\}, \{3\}\}$. Last, consider that $x_1(0) = 0$ and $x_2(0) = 1$. In Table I, we illustrate the execution of Alg. 1 for the first 7 time steps. The evolution of each agent state is also depicted in Fig. 1. In a more realistic scenario, the attacker may select distinct values for the attacked node to share in vector $c_a^{(t)}$, $a \in \mathcal{A}$. To illustrate this, consider the upper mentioned example, but with the attacked node 3 sharing, for each time instance, the values obtained as: $c_a^{(t)} = \begin{bmatrix} 0.1{+}2^{-t} & 0.1{-}0.2(1.7^{-t}) & 0.1{+}0.6(1.5^{-t}) & 0.1{+}0.5(1.2^{-t}) \end{bmatrix}^\intercal$.

The proposed algorithm allows the detection and correction of the consensus state of each normal agent, see Fig. 2.

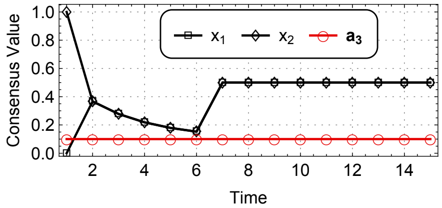


Fig. 1: State evolution of agents $\mathcal{V}$ using Alg. 1 to a discrete-time and synchronous communication CA, with $\mathcal{G}$ and set of attacked agents $\mathcal{A} = \{3\}$. $c_3[\mathcal{F}[i]] = c_3[\mathcal{F}[j]]$ for any $i, j = 1, \ldots, |\mathcal{F}|$.

Another noticeable point is how less restrictive Assumption 2 is when compared to the usual robustness requirements. In fact, Alg. 1 allows detecting and discarding attacked nodes even when more than half of the network nodes are controlled by an attacker. To illustrate this, consider the complete network with 5 agents, $\mathcal{G}$, and the set of attacked agents $\mathcal{A} = \{3, 4, 5\}$, with initial states represented in Fig. 3. Even though more than half of the agents are trying to deviate the consensus state of the normal agents, Alg. 1 with $f = 3$ and $\varepsilon = 0.05$ allows agents 1 and 2 to recover from such situation and achieve the consensus state which is the average

$c_u^{(t)}$	$c_u^{(t)}[p]$	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$
$c_1^{(t)}$	$c_1^{(t)}[\emptyset]$	**1.0**	**0.37**	**0.28**	**0.22**	**0.18**	**0.15**	0.14	0.12
	$c_1^{(t)}[\{1\}]$	1.0	0.55	0.32	0.21	0.16	0.13	0.11	0.11
	$c_1^{(t)}[\{2\}]$	1.0	0.05	0.08	0.09	0.09	0.1	0.1	0.1
	$c_1^{(t)}[\{3\}]$	1.0	0.5	0.5	0.5	0.5	0.5	**0.5**	**0.5**
$c_2^{(t)}$	$c_2^{(t)}[\emptyset]$	**0.0**	**0.37**	**0.28**	**0.22**	**0.18**	**0.15**	0.14	0.12
	$c_2^{(t)}[\{1\}]$	0.0	0.55	0.32	0.21	0.16	0.13	0.11	0.11
	$c_2^{(t)}[\{2\}]$	0.0	0.05	0.08	0.09	0.09	0.1	0.1	0.1
	$c_2^{(t)}[\{3\}]$	0.0	0.5	0.5	0.5	0.5	0.5	**0.5**	**0.5**
$c_3^{(t)}$	$c_3^{(t)}[\emptyset] = c_3^{(t)}[\{1\}] = c_3^{(t)}[\{2\}] = c_3^{(t)}[\{3\}]$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

TABLE I: Illustrative example of the proposed method to reach resilient consensus, with network of agents $\mathcal{G}$, attacked node $\mathcal{A} = \{3\}$ and $\varepsilon = 0.05$. The state of $x_u^{(t)}$ is the value in bold of the vector $c_u^{(t)}$.

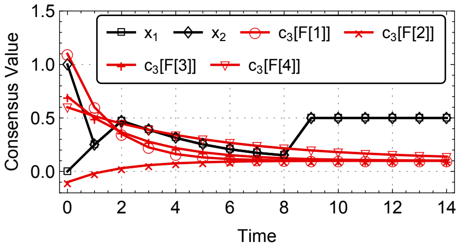


Fig. 2: State evolution of normal agents ($\mathcal{V} \setminus \mathcal{A}$) and of the vector shared by the attacked agent, $c_3^{(t)}$, using Alg. 1 and a discrete-time and synchronous communication CA, with $\mathcal{G}$ and $\mathcal{A} = \{3\}$.

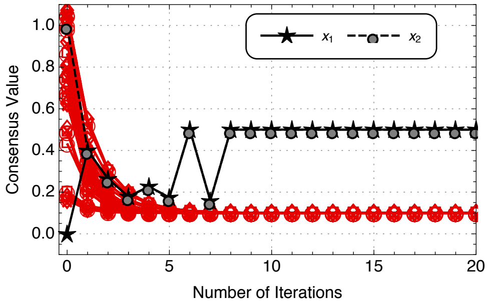


Fig. 3: State evolution of normal agents ($\mathcal{V} \setminus \mathcal{A} = \{1, 2\}$), and evolution of the shared by the attacked agents vector, $\mathcal{A} = \{3, 4, 5\}$, using Alg. 1 and a discrete-time and synchronous communication CA, with complete network of agents, $\mathcal{G}$.

of $x_1^{(0)}$ and $x_2^{(0)}$, see Fig. 3. The values in red correspond to the different entries of the vectors $c_v^{(t)}$, $t \in \mathbb{N}_0$, for $v \in \mathcal{A}$. If the goal of the method is just detecting the existence of attacked agents, instead of which are attacked nodes and without state correction, then we show a simplified and more efficient version of Alg. 1. In this case, we do not need to have a parameter f, and we can consider the set $\mathcal{F}$ to be $\mathcal{F} = \{\emptyset, \{1\}, \ldots, \{|\mathcal{V}|\}\}$, see Alg. 2.

Alg. 2: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a network of agents, C be a CA, and $N \in \mathbb{N}$ be the number of time steps for which we run C. Let $\varepsilon > 0$ and $\mathcal{F} = \{\emptyset, \{1\}, \ldots, \{|\mathcal{V}|\}\}$, and hence $|\mathcal{F}| = |\mathcal{V}| + 1 = n + 1$. The detection algorithm consists in the following three steps:

1. **(State Vector Initialization)** Each agent u sets $c_u^{(0)}[\mathcal{F}[i]] = x_u^{(0)}$;
2. **(State Vector Update)** For each time instance, $t \leq N$, each agent $u \in \mathcal{V}$ communicates a vector $c_u^{(t)}$ with size $|\mathcal{F}| = |\mathcal{V}| + 1$, where the i-th entry $c_u^{(t)}[\mathcal{F}[i]]$ is the result of algorithm C for agent u, utilizing the i-th entries of the last received neighbor vectors $c_v^{(t-1)}$, $v \in \mathcal{N}_u$;
3. **(Detection of Attacked Agents from the Vector State)** For $t \leq N$ every agent u checks if

(i) if $\underset{\mathcal{S} \in \mathcal{F}, \mathcal{S} \neq \emptyset}{\forall} \left| c_u^{(t)}[\emptyset] - c_u^{(t)}[\mathcal{S}] \right| < \varepsilon$ or

(ii) $\underset{\mathcal{S} \in \mathcal{F}, \mathcal{S} \neq \emptyset}{\exists!} \left| c_u^{(t)}[\emptyset] - c_u^{(t)}[\mathcal{S}] \right| \geq \varepsilon$, then it **detects** the existence of attacked agents and outputs TRUE;

(iii) else it **does not detect** attacked nodes and outputs FALSE. •

Next, we show that Alg. 2 detects the existence of attacked nodes, and the soundness and completeness of Alg. 1. That is, if there are several attacked agents, not exceeding f, then we detect them and achieve the consensus value discarding the malicious agents, and that if detection occurs with robustness parameter f, then there are at most f attacked agents. Next, we show that any normal agent outputs TRUE, if there are attacked agents and FALSE, otherwise.

Proposition 1: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a digraph with n nodes, C be a CA and $\mathcal{A} = \{v_1, \ldots, v_k\} \subset \mathcal{V}$ be a set of k nodes attacked by a malicious agent making each of their values converge to a value a, verifying Assumption 1, and $\varepsilon > 0$ be the computational precision used. Using Alg. 2 with robustness $f \geq |\mathcal{A}|$, each agent $v \in \mathcal{V} \setminus \mathcal{A}$ identifies after some time steps if there are attacked agents. ○

Proof: If there are attacked agents ($|\mathcal{A}| = k > 0$) the consensus of each agent in $\mathcal{G}$ using C converges to a. Each $v \in \mathcal{V} \setminus \mathcal{A}$ has a state vector with: a consensus value of considering all the nodes (first entry), all nodes except one (for a fixed ordering of the nodes) in the next n entries (for each set of nodes of the form $\mathcal{V} \setminus \{u\}$, $u \in \mathcal{V}$). If $k > 0$, after some time steps, for each normal node, either there is at most one of the consensus vector values that is different by at least ε (there is one attacked agent) or every consensus value is equal (they converged to the attacker value), step 3.(i). In both cases, $v \in \mathcal{V} \setminus \mathcal{A}$ outputs TRUE. If $\mathcal{A} = \emptyset$, by Assumption 1, the consensus value that each node computes for the entire graph is different (by at least ε) from the value for each subgraph without one of the nodes, and each $v \in \mathcal{V} \setminus \mathcal{A}$ does step 3.(iii), outputting FALSE. ■

A core issue when evaluating a resilience method is its added complexity. As discussed, methods removing smaller and larger received values have constant additional complexity, but do not detect some attacking cases. Next, we state the computational complexity of the Alg. 2. As a trade-off of the increased detection capabilities, our proposal increases linearly with the number of nodes the complexity.

Proposition 2: If the computational complexity of the CA C to run for $N \in \mathbb{N}$ time steps is $C(N)$, then Alg. 2 has time complexity of $\mathcal{O}(nC(N))$. ○

Proof: Let $C(N)$ be the computational complexity of the CA C to run for $N \in \mathbb{N}$ time steps and let $|\mathcal{V}| = n$, then Alg. 2 is equivalent to run C $n+1$ times, in step 2. (i.e., the number of entries of c_u for $u \in \mathcal{V}$), and the remaining steps (1. and 3.) have lower time complexity, more specifically $\mathcal{O}(n)$. Totaling a time complexity of $\mathcal{O}(nC(N))$. ■

A similar analysis can be performed for Alg. 1 which, besides the detection, allows each normal agent to recover and to converge to the correct steady state.

Theorem 1: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a digraph with n agents, C be a CA, and $\mathcal{A} = \{v_1, \ldots, v_k\} \subset \mathcal{V}$ be a set of k agents attacked by a malicious entity which makes these agents share values converging to a. Let $\varepsilon > 0$ be the precision utilized to do comparisons between values. In this scenario, Alg. 1 with robustness $f \geq k$ identifies, after a number of time steps, the attacked agents in $\mathcal{A}$, and the agents $v \in \mathcal{V} \setminus \mathcal{A}$ converge to the consensus value of $\mathcal{G} \setminus \mathcal{A}$ from the CA C.

Proof: If $k = 0$, then each non attacked agent $u \in \mathcal{V} \setminus \mathcal{A}$ holds a vector $c_u^{(t)}$ converging to a vector with entries all distinct from each other, since from Assumption 2 the consensus of each subgraph of $\mathcal{G}$ with $n-1$ nodes is distinct and different from the consensus of all nodes. Thus, agent u selects $x_u^{(t)} = c_u^{(0)}[\mathcal{F}[1]]$, where $\mathcal{F}[1] = \emptyset$. If $|\mathcal{A}| = k$ and $0 < k \leq f$, then there is $\mathcal{A}' \in \wp(\mathcal{V}, k)$ that is precisely the set of attacked agents, i.e., $\mathcal{A}' = \mathcal{A}$. That is, there is a smallest set of agents that, if excluded from the consensus computations, leads to a consensus value different from the attacked agents' values by at least ε, after a certain number of time steps. Further, every other subset in $\bigcup_{j=0}^{k} \wp(\mathcal{V}, j)$ has some attacked agent, implying that the normal agents converge to a. Hence, every agent $v \in \mathcal{V} \setminus \mathcal{A}$ outputs the consensus of the subgraph $\mathcal{G} \setminus \mathcal{A}$, which is that results of using C for a certain number of time steps, and v correctly identifies $\mathcal{A}$ as the set of attacked agents. ■

By Theorem 1, any normal node identifies and corrects its state. So, we check if the detection may yield false positives.

Proposition 3: Consider the digraph of n agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a CA C. By using Alg. 1 with $f \geq k = |\mathcal{A}|$ if, after some time steps, a node $v \in \mathcal{V}$ finds $k \leq f$ attacked nodes, then there exist k attacked nodes. ○

Proof: Let $v \in \mathcal{V}$ identify $k \leq f$ attacked nodes. Then, in step 3 of Alg. 1 the else occurs with $i = k$ and there is one and only one subgraph with k agents, $\mathcal{V}' \subseteq \mathcal{V}$, that converges to a consensus distinct from the network consensus. The other subgraphs have attacked nodes. Thus, if agent v finds k attacked nodes, then $\mathcal{G}$ has exactly k attacked nodes. ■

Next, we do the computational complexity analysis of Alg. 1. For each agent to be able not only to detect the existence of attacked nodes but also to identify which are the attacked agents and correct the consensus value, Alg. 1 has time complexity worse than the Alg. 2. However, it has polynomial time complexity, for a fixed f.

Proposition 4: Let the CA C have a computational complexity of $C(N)$ to run for $N \in \mathbb{N}$ time steps, then the worst-case time complexity of Alg. 1 is $\mathcal{O}(C(N)n^f)$. ○

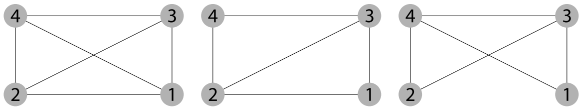
Proof: Suppose that the CA C has time complexity $C(N)$ to run for $N \in \mathbb{N}$ time steps. Thus, in step 3, Alg. 1 calls $\sum_{i=0}^{f} \binom{n}{i}$ times C. Since $\binom{n}{i} = \frac{1}{i!}\prod_{j=0}^{i-1}(n-j) = \mathcal{O}(n^i)$, it follows that $\mathcal{O}(\sum_{i=0}^{f} \binom{n}{i}) = \mathcal{O}(\max\{n, n^2, \ldots, n^f\}) = \mathcal{O}(n^f)$. As it is the larger computational complexity step, Alg. 1 has $\mathcal{O}(C(N)n^f)$ complexity for N time steps. ■

Proposition 4 says that Alg. 1 has polynomial complexity in the number of nodes, if C has polynomial complexity.

Corollary 1: Let the C have a computational complexity of $C(N)$ to run for $N \in \mathbb{N}$ time steps, and let $k \leq f$ be the number of attacked agents. Then, the time complexity of Alg. 1 is $\mathcal{O}(C(N)n^k)$ to run for N time steps.

IV. Illustrative Examples

In the following illustrative examples, we use the networks in Fig. 4 and $\varepsilon = 0.05$. In order to simplify the visualization of the examples, in what follows, the attacker shares a vector with all entries having the same value. This simplification is not needed, as we depicted in Fig. 2, but it would overload the figures with information. Also, the choosen CA C simply updates the value of node $v \in \mathcal{V}$ as $x_v = \frac{1}{|\mathcal{N}_v|}\sum_{u \in \mathcal{N}_v} x_u$.



(a) Network $\mathcal{G}_2$ (b) Network $\mathcal{G}_{[t<9]}$ (c) Network $\mathcal{G}_{[t\geq 9]}$

Fig. 4

Attacked Agents: Next, we explore the digraph of agents $\mathcal{G}_2 = (\mathcal{V}_1, \mathcal{E}_1)$. We consider two agents under attack, $\mathcal{A}_2 = \{1, 3\}$, sharing the values, at time t, given by the functions $7 - 2^{-0.3t}$ and $7 + 2^{-0.3t}$, respectively. Alg. 2 with $f \geq 2$ allows each agent to correctly detect the presence of network corrupted nodes. We depict the value of the unresilient consensus ($f = 0$) in Fig. 5 (a). When each agent $u \in \mathcal{V}_2 \setminus \mathcal{A}_2$ uses Alg. 1 with $f \geq 2$, it pinpoints exactly 1 and 3 as the attacked agents, converging to the correct consensus of the subnetwork $\mathcal{G}_2 \setminus \mathcal{A}_2$, see Fig. 5 (b).

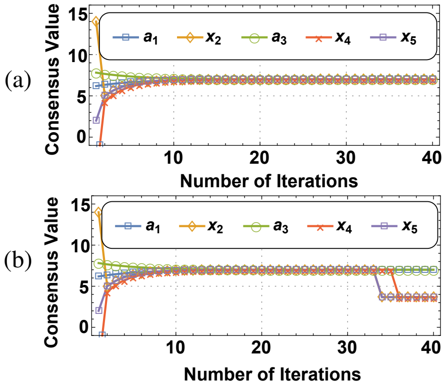


Fig. 5: (a) Consensus of C, with the network $\mathcal{G}_2$ and $\mathcal{A} = \{1, 3\}$; each agent converges to the attackers' values. (b) Output consensus of Alg. 1, with CA C, network $\mathcal{G}_2$ and $\mathcal{A} = \{1, 3\}$; after some time, each normal agent detects the attacked nodes 1 and 3 (a_1 and a_3), and corrects the consensus value.

Attacked Agents & Dynamic Network: Here, we demonstrate the main results in the scenario where there is a fixed set of nodes $\mathcal{V}$ in the network that evolve over time. We explore a set of nodes with a dynamic network that changes between two digraphs. When $t < 9$ the network is $\mathcal{G}(t) = \mathcal{G}_{[t<9]}$, see Fig. 4(b), and the network is $\mathcal{G}(t) = \mathcal{G}_{[t\geq 9]}$, see Fig. 4(c), otherwise. We render, in Fig. 6, the outcome of the consensus in the cases: (a) unresilient consensus ($f = 0$); (b) consensus using Alg. 1, with $f = 1$.

Alg.1 vs. Benchmark Resilient CAs: The conventional path to attain resilience to f attacked nodes in CAs builds upon the idea that each node rejects the set of the f largest and smallest of its neighbor states. We can easily note that an attacker in the former approach can still effectively tamper with the consensus value and drift it to the desired value or close to it. The attacker only needs to keep its state within the range of values of other nodes. This observation

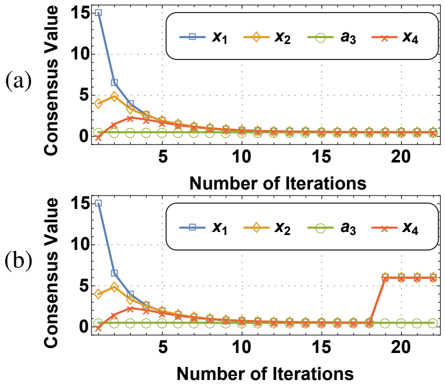


Fig. 6: (a) Consensus result of C, with the network of agents $\mathcal{G}(t)$ and attacked node 2; each agent converges to the attackers' values. (b) Output consensus of Alg. 1 and with CA C, network $\mathcal{G}(t)$ and $\mathcal{A} = \{2\}$; after some time, each normal agent detects the attacked agent 2 (a_2), choosing the correct consensus value.

is depicted with the network of nodes $\mathcal{G}_3 = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, 5\}$ and $\mathcal{E} = \{(i,j) : i,j \in \mathcal{V} \text{ and } i \neq j\}$. In $\mathcal{G}_3$, the attacker selects node 5 ($\mathcal{A} = \{5\}$) to make it perpetually forward the value 1. In this case, see in Fig. 7 (a) the state-of-the-art methods result, with robustness parameter $f = 1$. Thus, the attacker successfully deviated the consensus state of the network. The proposed method, Alg. 1, also with $f = 1$, allows the normal agents to detect the attacked node and correct their states, see Fig. 7 (b).

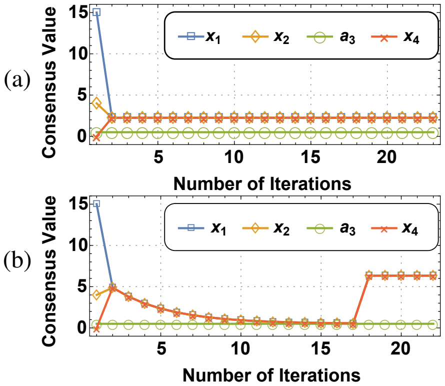


Fig. 7: (a) Consensus of using the **state-of-the-art** for resilient consensus, with $\mathcal{A} = \{5\}$ and network $\mathcal{G}_3$. Normal agents do not reach the correct consensus. The attack succeeds. (b) Consensus with Alg. 1 and CA C, attacked node 5 and network $\mathcal{G}_3$; after some time, each normal agent detects the attacked node 5 (a_5) and corrects the consensus value.

V. Conclusions & Future Research

We approached the problem of resilient discrete-time and synchronous communication consensus, when a malicious entity attacks a set of agents to make each of them converge to an attacker's desired value. We devised a general algorithm to tackle this problem that is suitable for an extensive range of CAs, which may be utilized for static and dynamic networks of agents. Our method enables normal nodes of a network to distinguish the set of attacked nodes. Each normal node, using our approach, is able to correct its state to the value of the consensus of the subnetwork without attacked agents. Furthermore, the algorithm we proposed possesses polynomial-time complexity. Future research involves investigating the concept behind the presented algorithm and improve resilience to different sorts of attacks.

References

[1] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Broadcast and gossip stochastic average consensus algorithms in directed topologies," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 474–486, June 2019.

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

[3] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *47th IEEE Conference on Decision and Control (CDC)*, Dec 2008, pp. 4185–4190.

[4] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[5] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, 2006.

[6] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[7] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *46th IEEE Conference on Decision and Control (CDC)*, Dec 2007, pp. 5492–5498.

[8] A. Haseltalab and M. Akar, "Approximate byzantine consensus in faulty asynchronous networks," in *American Control Conference (ACC)*, July 2015, pp. 1591–1596.

[9] ——, "Convergence rate analysis of a fault-tolerant distributed consensus algorithm," in *54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 5111–5116.

[10] D. Silvestre, P. Rosa, R. Cunha, J. P. Hespanha, and C. Silvestre, "Gossip average consensus in a byzantine environment using stochastic set-valued observers," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 4373–4378.

[11] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Finite-time average consensus in a byzantine environment using set-valued observers," in *American Control Conference*, June 2014, pp. 3023–3028.

[12] ——, "Stochastic and deterministic fault detection for randomized gossip algorithms," *Automatica*, vol. 78, pp. 46 – 60, 2017.

[13] H. Y. Öksüz and M. Akar, "Distributed resilient consensus: a nonparametric approach," *Transactions of the Institute of Measurement and Control*, p. 0142331218785673, 2018.

[14] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Transactions on Automatic Control*, pp. 1–1, 2018.

[15] S. M. Dibaji and H. Ishii, "Consensus of second-order multi-agent systems in the presence of locally bounded faults," *Systems & Control Letters*, vol. 79, pp. 23–29, 2015.

[16] Y. Kikuya, S. M. Dibaji, and H. Ishii, "Fault tolerant clock synchronization over unreliable channels in wireless sensor networks," *IEEE Transactions on Control of Network Systems*, pp. 1–1, 2018.

[17] S. Sundaram, "Ignoring extreme opinions in complex networks: The impact of heterogeneous thresholds," in *55th IEEE Conference on Decision and Control (CDC)*, Dec 2016, pp. 979–984.

[18] J. Usevitch and D. Panagou, "Resilient leader-follower consensus to arbitrary reference values in time-varying graphs," *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1755–1762, 2019.

[19] D. Saldana, A. Prorok, S. Sundaram, M. F. M. Campos, and V. Kumar, "Resilient consensus for time-varying networks of dynamic agents," in *American Control Conference (ACC)*, May 2017, pp. 252–258.

[20] S. M. Dibaji, H. Ishii, and R. Tempo, "Resilient randomized quantized consensus," *IEEE Transactions on Automatic Control*, 2017.

[21] Y. Chen, S. Kar, and J. M. F. Moura, "Attack resilient distributed estimation: A consensus+innovations approach," in *American Control Conference (ACC)*, June 2018, pp. 1015–1020.

[22] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184.

[23] S. M. Dibaji and H. Ishii, "Resilient consensus of second-order agent networks: Asynchronous update rules with delays," *Automatica*, vol. 81, pp. 123 – 132, 2017.