# Attack resistant Leader Election in Social Overlay Networks by Leveraging Local Voting

Martin Byrenheid
TU Dresden
martin.byrenheid@tu-dresden.de

Thorsten Strufe
TU Dresden
thorsten.strufe@tu-dresden.de

Stefanie Roos
Delft University of Technology
s.roos@tudelft.nl

## ABSTRACT

Current leader election algorithms fail in the presence of Sybil attacks, i.e., one malicious entity inserting many nodes, network dynamics, and restricted knowledge about the graph. However, social overlay networks, i.e., peer-to-peer networks with links corresponding to social relationships, face all of the above challenges. Social overlay networks naturally offer privacy, as they avoid connections with strangers, and furthermore prevent a Sybil attacker from controlling a large number of links in the graph. As recent ideas for scalable communication in such overlays rely heavily on attack resistant leader election, solving leader election for such overlays opens the door for decentralized, privacy-preserving, and secure communication at a large scale.

In this work, we propose a novel leader election algorithm based on three-majority voting that utilizes timestamps and cryptographic signatures to detect leader faults in an attack resistant manner. We evaluate our algorithm with simulations on real-world as well as synthetic network topologies. Our results indicate that in networks whose degree sequence follows a power law, our leader election algorithm quickly achieves consensus for more than 80% of all nodes. Furthermore, attackers are unlikely to become leaders as long as the number of connections they establish with honest nodes is low.

## CCS CONCEPTS

• **Networks** → **Network simulations**; **Peer-to-peer networks**; *Network dynamics*; • **Computer systems organization** → *Reliability*.

## KEYWORDS

leader election, peer-to-peer networks, social overlay networks, dynamic networks

**ACM Reference Format:**
Martin Byrenheid, Thorsten Strufe, and Stefanie Roos. 2020. Attack resistant Leader Election in Social Overlay Networks by Leveraging Local Voting. In *21st International Conference on Distributed Computing and Networking (ICDCN 2020), January 4–7, 2020, Kolkata, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3369740.3369770

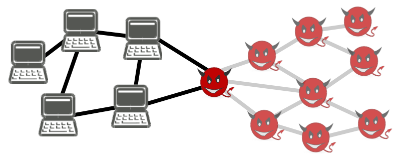

**Figure 1: Because the set of participants is unknown to the correctly operating nodes, a malicious node can pretend to be connected to an arbitrary number of fake nodes.**

## 1 INTRODUCTION

The problem of leader election is one of the most researched topics in the area of distributed computing. Because of the high communication overhead required by traditional solutions, alternative notions have emerged to cover large-scale networks such as peer-to-peer overlays [1, 23]. However, these solutions still require that more than 2/3 of all nodes operate correctly, restricting their use to networks that can prevent participants from creating an arbitrary number of fake nodes.

One type of peer-to-peer network without centralized admission control that is likely to benefit from leader election are social overlay networks. These networks represent a promising idea for the realization of scalable, attack resistant and privacy-preserving distributed services over the Internet [9, 27, 35]. In contrast to other peer-to-peer networks, social overlay networks restrict connectivity to nodes whose users share a mutual trust relationship in the real world. This approach improves privacy by limiting the exposure of identifying information, such as the node's IP address, to trusted participants.

Due to the lack of centralized admission control, social overlay networks are highly vulnerable to the insertion of an arbitrary number of fake nodes, also known as *Sybil attack* [15] (as illustrated by Figure 1). Not only do Sybil attacks render the 2/3-majority assumption unrealistic, they also make seemingly weak assumptions such as the existence of a reliable estimate of the network size problematic. However, an adversary that aims to insert nodes under his control into a social overlay network first needs to perform social engineering to trick participants into accepting a connection from its nodes. Thus, social overlay networks might contain an arbitrary number of malicious participants but the number of links between malicious and honest participants is likely to be small in comparison to the total number of links.

Leader election comes into play when considering communication in the network. Recent approaches suggest that spanning tree-based routing algorithms will drastically increase the performance of social overlays [30]. Electing an honest root node is key for the success of these routing algorithms. However, despite its importance, the problem of leader election in social overlays remains unsolved.

Solving leader election in social overlays requires overcoming three challenges. In addition to the key challenge of Sybil attacks, scalability and the ability to deal with network dynamics are of utmost importance. Networks may consist of thousands of nodes and participants join and leave frequently.

In this work, we investigate whether distributed voting algorithms [4, 10, 21] can serve as a foundation for scalable and attack resistant leader election in social overlay networks. These algorithms rely only on local interaction between directly connected nodes and do not have any built-in assumptions about global network properties (e.g., the number of nodes or the mixing time [25]), making them a promising substrate for leader election.

Concretely, we design a leader election protocol based on three-majority voting [4]. In contrast to the original protocol, our algorithm can deal with network dynamics, including the failure of a leader. We furthermore leverage digital signatures to prevent malicious parties from impersonating leaders.

Our extensive simulation study reveals that our algorithm converges fast on various synthetic and real-world social networks. However, it heavily relies on the power-law distribution of a social network and shows a drastically reduced convergence speed for graphs with a uniform degree distribution. In terms of attack resistance, we evaluate the likelihood of an attacker to be elected leader. As long as the attacker establishes at most twice as many connections as an average honest node, its chance to become the leader amounts to 1.9% in our real-world data set.

## 2 RELATED WORK

In the following, we survey the state of the art of Byzantine leader election and explain which assumptions render existing works unsuitable for social overlay networks. Afterwards, we present the necessary background on distributed voting protocols, which we build upon in the following sections.

### 2.1 Byzantine leader election

Traditional Byzantine leader election algorithms [17, 19, 22, 32] guarantee consensual choice of a single leader amongst *all* nodes, even if a fraction of nodes behaves incorrectly. Furthermore, traditional algorithms guarantee that with constant probability, the chosen leader is a correctly-behaving node. Nodes deviating from the correct behavior are called *Byzantine nodes*.

However, algorithms for traditional Byzantine leader election assume a static clique network where the total number of participating nodes is known to all nodes and all communication is done by broadcast, thus limiting their use to networks with a few hundred nodes. To enable Byzantine leader election in large networks such as peer-to-peer networks, King et al. [23] relaxed the requirement that *all* non-Byzantine nodes agree on the same leader and proposed a solution without broadcast communication. While the solution of King et al. only works on static networks, Augustine et al. [1] consider Byzantine leader election in large, dynamic networks. Similar to King et al., Augustine et al. allow a fraction of non-Byzantine nodes to select a different leader. Unfortunately, both works critically rely on the assumption that less than one third of all nodes behaves incorrectly. As outlined in Section 1, this is unrealistic for social overlay networks.

Several algorithms related to Byzantine leader election have also been proposed for wireless sensor networks (WSNs) [14, 33, 37]. These solutions employ cryptographic techniques to keep Byzantine nodes from increasing their chance of being elected. However, all works in the area of WSNs assume that every node holds a list with the identifiers of all other nodes. Because we allow attackers to create an arbitrary number of fake identifiers, they can easily compromise such schemes.

In summary, all of the existing solutions for Byzantine leader election assume that only a minority of nodes in the network is Byzantine. Thus they cannot meet their security guarantees in networks where an attacker can introduce an arbitrary number of malicious nodes.

### 2.2 Distributed voting algorithms

Distributed voting algorithms resemble the spread of opinions and the forming of consensus in social systems. Such algorithms are interesting in the context of social overlays as they operate without any knowledge about global network properties, such as the size of the network or the current state of all nodes.

Common to all distributed voting algorithms is that every node has a $vote$-variable that holds an element from a set $\mathcal{V}$ of possible votes and that each node adjusts its $vote$-value at regular intervals. In the context of leader election, the set $\mathcal{V}$ contains the unique identifier of each node in the network. Furthermore, the network may start in a state where each node's $vote$-value holds its own identifier.

In the most basic variant, also called the *voter model*, each node periodically requests the current $vote$-value of a randomly chosen neighbor and writes the response into its own $vote$-register [5]. While arguably very simple, this algorithm suffers from slow convergence, as the time needed for global consensus grows linearly with the network size [5].

To reduce convergence time, recent algorithms favor popular votes, i.e., those present in the $vote$-variable of many nodes, more strongly. Cooper et. al. [10] proposed *two-sample voting*, where every node periodically requests the $vote$-value of two randomly chosen neighbors. If both neighbors respond with the same value, then this value is adopted. Otherwise, the requesting node keeps its current $vote$-value. We consider two-sample voting to be unsuited for leader election, where every node may start with a different $vote$-value. In this case, nodes with a high number of neighbors are unlikely to change their $vote$-value until a sufficient number of neighbors have already adopted a common $vote$ value, thus slowing down the election.

Currently, the most promising opinion dynamics for leader election is three-majority voting [4], where every node $u$ periodically requests the $vote$-value of three randomly chosen neighbors. If at least two of the received responses contain the same $vote$-value $v$, then the node adopts $v$ as its new $vote$-value. Otherwise, i.e., all three responses contain different values, it adopts a random vote out of the three responses. Thus, if all nodes start with a different $vote$-value, three-majority voting initially behaves similar to the voter model but as soon as a $vote$-value is adopted by a sufficient number of nodes, three-majority voting will converge more quickly [6].

So far, existing work on the convergence speed and attack resistance of three-majority voting focuses exclusively on complete network topologies [3, 4, 6, 18]. Thus, it remains an open question how well three-majority voting performs in non-complete networks.

## 3 MODEL AND NOTATION

In the following, we present a simple formal model for social overlay networks and describe our threat model.

### 3.1 System model

We model a social overlay network at a fixed point in time as a connected, undirected graph $S = (V, E), |V| = n$, where each edge corresponds to a bidirectional connection between a pair of participants. In the following, $N(u) = \{v \in V \mid \{u, v\} \in E\}$ denotes the *neighbors of* $u$.

Since social overlay networks are dynamic, nodes may join or leave the system and connections between nodes are established or torn down over time. Leaving includes the case that a node notifies its neighbors prior to its departure as well as that the node crashes abruptly. We do not make any assumptions about the number of nodes being affected by an arrival or departure of a node. It is therefore possible that a large number of nodes joins the overlay at once (e.g., because a connection between two separated overlays has been established) or leaves the overlay (e.g., because the network got partitioned).

We do not assume that any node knows the overall number of participants or other global properties at any point in time. The only assumption we make regarding knowledge about the network structure is that there exists a globally known upper bound $D$ on the diameter of the network. At every point in time, the actual diameter of the overlay never exceeds $D$ but can be much lower than $D$. We consider this assumption to be realistic, as existing work on online social networks indicates that even networks with millions of nodes have a diameter below 30 [26].

To perform leader election, each node may send messages to its neighbors and process received messages. We assume that the overlay network operates *synchronously*, meaning that

- The clocks of any pair of nodes differ at most by a constant $\Delta_C$.
- The time needed to process received messages and to send new messages is bounded by a constant $\Delta_E$.
- The delay between the sending of a message and its arrival at the corresponding neighbor is bounded by a constant $\Delta_D$.

The constants $\Delta_C$, $\Delta_E$ and $\Delta_D$ are known to all nodes.

Based on the aforementioned assumptions, we design our algorithm such that all nodes operate in *synchronous rounds*, where each round lasts for $\Delta_R = \Delta_E + \Delta_D$ time units. Within each round, every node processes incoming messages and sends new messages exactly once.

### 3.2 Adversary model

In this work, we consider an adversary that controls a set $B$ of colluding *malicious* (or *adversarial*) nodes and is able to set up a bounded number of connections between nodes under his control and non-malicious nodes (called *honest nodes*) $H$.

We consider this adversary to be relevant because the setup of connections in social overlays requires large-scale social engineering, which we deem to be difficult in practice. Thus, the adversary will only be successful for a subset of the overlay's participants.

The adversary aims to maximize the chance that a malicious node is elected as leader, which may serve as starting point for attacks on the algorithms building upon the leader election. To bias the election towards themselves, adversarial nodes may actively interfere with the leader election process by deviating arbitrarily from the leader election algorithm. However, we assume that the adversary does not know all honest nodes and their connections between each other a priori. As a consequence, he cannot choose which honest nodes will connect to malicious nodes. We consider this assumption to be realistic, because social overlay networks are large-scale and dynamic distributed systems with participants from multiple countries.

Furthermore, we assume that the adversary is limited to polynomial-time attacks and hence unable to break secure cryptographic primitives. This is in accordance with the Dolev-Yao model [13], which has been accepted as realistic for all practical purposes.

## 4 VOTING-BASED ELECTION

As explained in Section 2.1, existing leader election algorithms fail if an attacker can create a large number of fake nodes. In the following, we thus propose a novel leader election algorithm based on three-majority voting [4]. Three-majority voting operates solely on information of each node's neighborhood, which renders it suitable for large-scale networks. Furthermore, it is not affected by the number of malicious nodes but only by the number of edges between honest and malicious nodes.

### 4.1 Adaptation of three-majority voting

To enable leader election based on three-majority voting, each node needs to have a unique identifier. Assigning unique identifiers in a privacy-preserving manner is thus the first step of our protocol. As a consequence, IP addresses are unsuitable, as they reveal sensitive information about the leader.

Instead, we let every node generate a public/secret key pair $ID_u, SK_u$ of an asymmetric cryptosystem once upon startup. Each pair of keys is generated locally by each node, without reliance on a third party, such as a public key infrastructure. We use the term *ID of node u* synonymously for the public key $ID_u$ of $u$. For a given public key $ID$, we say that $u$ *owns* $ID$ if $SK_u$ is the correct secret key for $ID$. Given that the length of each public key is $m$ bits, $\mathcal{ID} = \{0, 1, .., 2^m - 1\}$ denotes the set of *possible* identifiers.

To indicate which identifier is considered to be the leader's identifier, every node $u$ has a $leader_u$ variable that holds a value from $\mathcal{ID}$. Upon startup of a node $u$, $leader_u$ holds $ID_u$. As stated above, nodes then change their leader value to the most common value within a set of three values picked uniformly at random from its neighbors. Since nodes in social overlay networks may have less than three neighbors, the neighbors are selected randomly with replacement. If at least two of the chosen neighbors have the same $leader$ value $ID'$, $u$ sets its own $leader$ value to $ID'$. If all three neighbors hold a different $leader$ value, $u$ randomly selects one of these values and adopts it as its $leader$ value.

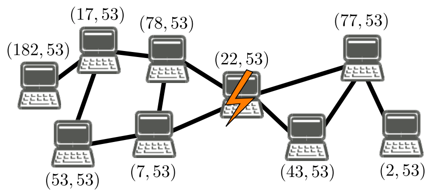

**Figure 2: Example for invalid *leader* values due to a node fault. The tuples next to each node denote its respective *ID* and *leader* value. Because node 22 crashed, node 43, 2 and 77 now form a separate overlay in which there is no node with identifier 53.**

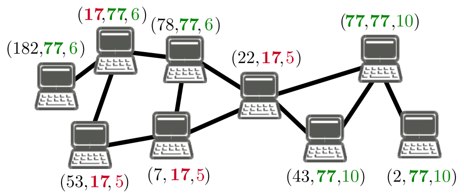

**Figure 3: Example for erroneous expiration of timestamps. The tuples next to each node denote its *ID*, *leader* and *ts* value. Since node 22 considers node 17 to be the leader, it does not propagate timestamp value 10 from node 77 to node 78. Furthermore, node 17 also considers node 77 to be the leader and thus node 53 does not receive more recent timetamps.**

In contrast to the original algorithm [4], our version does not include a node's own current leader value when choosing a new value. This modification is motivated by the fact that two-sample voting has been shown to converge fast when not including a node's own current leader [11].

## 4.2 Handling leader faults

Three-sample voting cannot be used as-is in dynamic networks, because it does not adapt to node faults. Figure 2 shows an example for an overlay in which a node with identifier 53 has been elected as leader. Due to a node fault, the overlay is separated into two partitions. Without further measures, all nodes belonging to the partition without node 53 will keep considering node 53 as leader.

In the following, we say that an identifier is *invalid* if none of the nodes currently participating in the social overlay owns this ID.

To eventually recognize and remove invalid identifiers, each node $u$ attaches a timestamp $ts_u$ to its own identifier. During the election procedure, this timestamp will then be propagated together with $u$'s ID. At time $t$, an identifier is considered to be invalid if $t - ts_u > \Delta_C + D\Delta_R$, where $\Delta_C$, $\Delta_R$ and $D$ are the globally known bounds from Section 3.1.

We favor a timeout-based approach over an active propagation of notifications about node failures because of its lower complexity. An active propagation scheme must be made robust against fake notifications as otherwise, malicious nodes can deliberately cause honest nodes to drop popular identifiers.

To keep malicious nodes from altering the timestamp associated with an identifier, each node $u$ that considers itself to be the leader also attaches a digital signature $sig_u$ of $ts_u$, computed using $u$'s secret key, to its identifier. Upon reception of a message, each node then checks if the provided signature is correct and drops the message in case the check fails.

Given that social overlays are large-scale dynamic networks, maintaining a global list of currently active identifiers shared by all nodes would incur a high overhead. By creating a large number of fake identifiers, malicious nodes could furthermore exploit such a mechanism to exhaust the memory and bandwidth of honest nodes. Instead, we embed the detection of invalid identifiers into the voting process such that if a node $u$ considers itself to be the leader, it periodically generates and propagates fresh timestamps for its own identifier. Otherwise, it propagates the most recent timestamp received for the identifier currently in its *leader*-variable. As illustrated by Figure 3, our approach might cause honest nodes to wrongly consider a valid identifier to be invalid if i) the owning node does not consider itself to be the leader or if ii) there is no path to the leader with $ID$ on which all nodes have $leader = ID$. However, our evaluation in Section 5 shows that in networks whose degree distribution follows a power law, most nodes still quickly reach agreement on a single valid identifier.

The three-majority voting proposed by Becchetti et al. [4] implements a pull-based approach where nodes actively request the current *leader*-value of the randomly chosen neighbors. While this approach has a low communication overhead, it is unsuited for the propagation of new timestamps in a timely manner, as a node might choose to contact the neighbor with the most recent timestamp just after several rounds have passed. Furthermore, if a neighbor of the requesting node crashed or deliberately drops messages, the latter will not receive a sufficient number of responses and thus be unable to update its state accordingly.

To avoid the aforementioned problems, we instead rely on a push-based approach where every node broadcasts its current *leader*-value, including the most recent timestamp and the corresponding signature, to all its neighbors in every round. Each node then keeps the most recently received values for each neighbor, such that the local voting procedure can be performed locally, without having to send additional requests.

Due to the local broadcast, nodes propagate new timestamps early, such that nodes with a high distance to the leader also receive new timestamps in a timely manner. Furthermore, the broadcast messages also implicitly serve as a heartbeat mechanism that can be used to detect if a neighbor has crashed.

Because existing works [24, 26] indicate that social networks are sparse such that nodes have a low degree and each node only needs to send a $(leader, ts, sig)$-tuple to each neighbor, we consider the overhead for this approach to be acceptable.

## 4.3 Leader election algorithm

Algorithm 1 displays the pseudocode of our leader election algorithm. Upon start, every node writes its own $ID$ into its $leader$ variable, computes the corresponding timestamp together with a signature and sends its state to all its neighbors (Line 3-7). The function *sign* uses the secret key given as first argument to produce a digital signature for the timestamp given as second argument. Afterwards, using the globally known constants given in Section 3.1, each node estimates the number of passed rounds and sets a timer for the next round (Line 8-9).

**Algorithm 1:** Leader Election at node $u$

```
State: u's public key $ID_u$ and secret key $SK_u$, u's leader value
       $leader_u$, an array $neigh_u$ holding the most recent
       $(leader, ts, sig)$-tuple from each neighbor
upon start:
begin
    $T := time()$
    $leader_u := ID_u$
    $ts := T$
    $sig := sign(SK_u, ts)$
    $broadcast(leader_u, ts, sig)$
    $passed := \lfloor \frac{T}{\Delta_R} \rfloor$
    $setTimer(\Delta_C + (passed + 1)\Delta_R)$
upon timer:
begin
    $T := time()$
    $validNeigh := getValidEntries(neigh, T)$
    if $|validNeigh| > 0$ then
        $votes := pickUniform(validNeigh, 3)$
        $leader_u := majorityVote(votes)$
        $mrt := getMostRecent(leader_u, validNeigh)$
        $ts := mrt.t$
        $sig := mrt.sig$
    else
        $leader_u := ID_u$
    if $leader_u = ID_u$ then
        $ts := T$
        $sig := sign(SK_u, ts)$
    $broadcast(leader_u, ts, sig)$
    $setTimer(T + \Delta_R)$
upon message $(leader, ts, sig)$ from neighbor $i$:
begin
    $neigh.i = (leader, ts, sig)$
```

When the timer of a node $u$ fires, $u$ first computes those neighbors whose most recently reported $(leader, ts, sig)$-tuple is valid (Line 12-13). For the given set *neigh* of tuples and the given timestamp $T$, the function *getValidEntries* returns only those tuples from *neigh* which (1) hold a correct signature over $ts$ for the given leader ID and (2) whose timestamp does not differ by more than $\Delta_C + M\Delta_R$ time units from $T$. We call $M$ the *expiry parameter* and discuss suitable values for $M$ in Section 4.4. If none of the neighbor tuples satisfies both aforementioned conditions, $u$ will reset its *leader* variable to its own identifier (Line 21). If at least one neighbor reported a valid tuple, $u$ will locally perform three-majority voting (Line 15-16). Using the function *pickUniform*, $u$ will pick three out of all valid tuples uniformly at random. Since it is possible that a node only has one or two neighbors in the social overlay network, *pickUniform* may pick a neighbor multiple times. For a given set *votes* containing three $(leader, ts, sig)$-tuples, the function *majorityVote* then checks whether at least two tuples hold the same *leader*-value $ID'$. If this is the case, *majorityVote* returns $ID'$ and otherwise, *majorityVote* picks one of the three tuples uniformly at random and returns the corresponding *leader* value. After node $u$ has updated its *leader* variable, it looks up the most recent timestamp and corresponding signature for this identifier among all its valid neighbor tuples (Line 17). Given an identifier *leader* and a set $\mathcal{N}$ of neighbor tuples, the function *getMostRecent* returns a tuple $(leader, t, sig)$ from $\mathcal{N}$ such that $t = \max_{(l,ts,sig)\in\mathcal{N}}\{ts \mid l = leader\}$.

If at least one neighbor of $u$ considers $u$ to be the current leader node, it is possible that *majorityVote* returns $u$'s own identifier. In this case, $u$ must not use the timestamp collected by the *getMostRecent* function but instead propagate a new timestamp. Thus, after the actual voting procedure, $u$ checks if it considers itself to be the leader and if so, sets the timestamp to be propagated to its current clock time (Line 22-24). Subsequently, $u$ sends its current *leader* value together with a corresponding timestamp and signature to all its neighbors.

In contrast to the existing algorithms for Byzantine leader election described in Section 2.1, our algorithm does not terminate execution. However, in the absence of changes to the network topology, it will *eventually* converge to a stable state.

## 4.4 Eventual Convergence

In this section, we prove that our algorithm eventually converges to a stable state, assuming a non-bipartite graph. The key idea is to show that the evolution of the network under Algorithm 1 corresponds to an absorbing Markov chain. Absorbing Markov chains are guaranteed to converge towards an absorbing state, i.e., a state $s$ for which the transition probability to any state $s'$ is 1 for $s' = s$ and 0 otherwise [20].

Theorem 1. *Let $G$ be a static graph that is non-bipartite, i.e., there exists a cycle of length $2k+1$ for some natural number $k$. If the expiry parameter satisfies $M \geq 2D + k - 1$, Algorithm 1 eventually reaches a state with $leader_u = leader_v$ for all nodes $u$ and $v$.*

In the context of social overlays, we assume $k = 1$, i.e., there is at least one triangle, as social networks exhibit strong clustering [28].

Proof. For simplicity, we first consider a version of the algorithm that does not use timestamps, which are purely included to deal with topology changes. Afterwards, we explain why the result holds even in the presence of timestamps.

We show that the Markov chain corresponding to the simplified version is time-homogeneous, has absorbing states, and is aperiodic. It then follows that the Markov chain is absorbing and the algorithm converges. For modelling the algorithm, we first enumerate the nodes in the network as $v_1, \ldots, v_n$. We define the state set $S$ as the set of vectors of length $n$ with elements in $\mathcal{ID}$. As indicated by the timers, Algorithm 1 proceeds in rounds with each node adapting its leader value once per round. Let $(S_t)_{t\in\mathbb{N}_0}$ with $S_t \in S$ be the random process representing the changes of the leader values.

All nodes change their leader values using three-sample voting based solely on the leader value of the neighbors in the previous round. As w assume that the topology and hence the sets of neighbors remain the same, we have for all $s, s' \in S$ and all $t > 0$ that

$$P(S_t = s'|S_{t-1} = s) = P(S_1 = s'|S_0 = s).$$

In other words, the probability distribution of $S_t$ only depends on $S_{t-1}$ and do not change over time. So, $(S_t)_{t\in\mathbb{N}_0}$ is indeed a time-homogeneous Markov chain.

The Markov chain has absorbing states, namely all states when all nodes have the same leader. It remains to argue that every non-absorbing state can reach an absorbing state in a finite number of steps, i.e., that the Markov chain is aperiodic.

We first consider a state $s$ such that at least two neighboring nodes $u$ and $v$ have the same leader value $ID$. With a non-zero probability, $u$ and $v$ both choose the other's value as their new leader and hence keep $ID$ as their leader value. Also, all neighbors of $u$ and $v$ choose $ID$ as their leader value with non-zero probability. Given that all these nodes now have at least one neighbor with leader value $ID$, it inductively follows that within $D$ rounds all nodes can have $ID$ as their leader value with non-zero probability.

Now, let $s'$ be a state such that neighboring nodes always have distinct leader values. By assumption, there is at least one circle of length $2k+1$. Let $u_1, \dots, u_{2k+1}$ be nodes on such a circle: $u_i$ and $u_{i+1}$ are neighbors as well as $u_{2k+1}$ and $u_1$. With non-zero probability, $u_k$ and $u_{k+2}$ choose the leader value $ID_{k+1}$ of $u_{k+1}$ in the next round. In the following round, $u_{k-1}$ and $u_{k+3}$ then can choose $ID_k$. By induction, we get that after $k$ rounds, $u_1$ and $u_{2k+1}$ have the leader value $ID_{k+1}$ with non-zero probability. So, there is a non-zero probability to go from $s'$ to a state $s$ where two neighboring nodes have the same leader value within $k$ rounds. It follows that there is a non-zero probability to go from $s'$ to an absorbing that within $D + k$ rounds for all states.

The inclusion of timestamps requires further assumptions to allow for absorbing states. Timestamps need to be part of the state information as an expired timestamp affects the transition probabilities. In order to have absorbing states, we need to assume that latencies are constant and rounds always of the same duration. Then, we can replace each timestamp with a decreasing counter indicating the rounds until it expires.

Now, the state space for Algorithm 1 consists of vectors in $\hat{S} = \mathcal{ID} \times R$ with $R = \{0, \dots, M\}$ being the number of rounds until a timestamp expires. Thus, the Markov chain $(\hat{S}_t)_{t \in \mathbb{N}_0}$ records the leader value and rounds until expiry for the corresponding timestamp of each node. As for the simplified version, $(\hat{S}_t)_{t \in \mathbb{N}_0}$ is time-homogeneous as the timestamp values of each round follow from the values of the previous round.

It is not immediately obvious why $(\hat{S}_t)_{t \in \mathbb{N}_0}$ has absorbing states. Consider a state $s = ((l_1, r_1), \dots, (l_n, r_n))$ such that $l_i = l_j$ for all $i, j$. We claim that $s$ is an absorbing state if $r_i$ is the difference of the maximal value $m$ of the timestamp and the hop distance $h$ of $v_i$ to the leader. By Line 17 of Algorithm 1, each node selects the most recent timestamp in its neighborhood. As a consequence, the leader will always choose its own timestamp, which has the maximal value. The claim follows by induction on the distance to the leader.

Concerning aperiodicity, first consider states $s_t$ such that at least one node $v$ has its own ID as leader value. Then the corresponding time until expiry of $ID_v$ is $M$. As above, let $u_1, \dots, u_{2k+1}$ be nodes on a circle of length $2k + 1$. We distinguish two cases: i) all nodes $u_i$ are at distance $D$ to $v$ and ii) at least one node $u_i$ has distance at most $D - 1$ to $v$.

For the first case, there is a non-zero chance that all nodes on the circle have the leader value $ID_v$ after exactly $D$ rounds as the $i$-th node on their path to $v$ can have after $i$ rounds. As detailed for the simplified algorithm, $ID_v$ can be the globally agreed-upon leader after another $D$ rounds. In particular, after at most $2D \leq M$ rounds, $v$ can have its own ID as leader value again and start to generate new timestamps. Once the optimal timestamps spread to all nodes, an absorbing state is reached.

For the second case, the node $u_i$ can have $ID_v$ as its leader value within at most $D - 1$ steps. After $k$ more rounds, two neighboring nodes on the circle can have $ID_v$ as their leader value, as for the simplified version. From then onward, all nodes can have $ID_v$ as their leader value within $D$ rounds. As $D - 1 + k + D = 2D + k - 1 = M$, the timestamp does not expire until $v$ starts generating new timestamps. So again, the process can reach an absorbing state with non-zero probability in a finite number of rounds.

In summary, the Markov chain $(\hat{S}_t)_{t \in \mathbb{N}_0}$ is also absorbing and hence Algorithm 1 converges. □

Note that Theorem 1 does not provide any bounds on the time of convergence. Indeed, previous research on voting algorithms indicates the existence of metastable phases [12]. In a metastable phase, there are multiple leader values active in the graph but the majority of nodes does not change their value for a long time. As shown in the next section, metastable phases are common for communities with few connections between them. In such a case, each community sticks to one leader value and there is no global consensus. While the above argument shows that Algorithm 1 will eventually result in consensus and hence overcome the metastable phase, the probability of overcoming such a state can be very low and hence it can take a long time to escape it. Thus, for practical reasons, there is a chance that the algorithm remains (seemingly) stuck in such a state.

In the presence of malicious nodes that do not follow Algorithm 1 correctly, e.g., by never adopting the identifier of an honest node, all honest nodes will eventually adopt the identifier propagated by a malicious node, electing it as global leader. Due to the possibility of reaching a metastable state however, it may actually take a very long time until all honest nodes adopt such an identifier.

## 5 EMPIRICAL RESULTS

In previous work, three-majority voting has only been considered for complete graphs. Thus, there exists no knowledge on how it performs on realistic network topologies, in particular social graphs. To evaluate the scalability and attack resistance of Algorithm 1, we performed a simulation study that addresses the following research questions:

(1) How does the topology of the network affect the convergence of the election and the emergence of metastable phases?
(2) How does the likelihood of electing a malicious node relate to the number of edges between honest and malicious nodes as well as the network topology?

### 5.1 Data sets

To date, there are no snapshots of real-world social overlay networks like the Freenet [9] or Briar[1] available that can be used for simulation. Since social overlay networks are explicitly designed to prevent collection of information about its participants and their connections, it is difficult to obtain topological information.

[1]https://briarproject.org/, 2019-07-10

**Table 1: Properties of the graphs used for simulation, including average degree $\overline{deg}$, diameter $D$, average shortest path length $\overline{spl}$ and modularity $M$.**

| Graph | n | $\overline{deg}$ | D | $\overline{spl}$ | M |
|---|---|---|---|---|---|
| Facebook | 63,392 | 25.8 | 15 | 4.32 | 0.62 |
| SPI | 9,222 | 10.58 | 12 | 4.67 | 0.62 |
| Brightkite | 56,739 | 7.5 | 18 | 4.92 | 0.66 |
| Ripple | 67,149 | 2.9 | 15 | 3.82 | 0.69 |
| Rand. Facebook | 63,392 | 25.8 | 8 | 3.58 | 0.15 |
| Barabási-Albert (BA) | 63,392 | 25.9 | 5 | 3.32 | 0.17 |
| Erdös-Renyi (ER) | 63,392 | 26 | 5 | 3.74 | 0.14 |

As social overlays represent social relationships, we instead utilized data sets from popular online social networks. The most important characteristics of our data sets are summarized in Table 1. To quantify the occurrence of community structure, Table 1 includes an approximation of the modularity measure for each graph using the algorithm of Blondel et al. [7]. Informally, a high modularity indicates a higher prevalence of weakly interconnected communities, i.e. communities with few edges between them.

*Facebook* denotes the largest connected component of a graph obtained by crawling a part of the Facebook social network [36]. Similarly, *SPI* represents the largest connected component of the users of an university online social network [29]. *Brightkite* denotes a graph obtained from the Brightkite network, a location-based online social network [8]. In all of these graphs, each node represents a user account and each edge represents a friendship between two users. The *Ripple* data set denotes a snapshot of the Ripple payment network [31], where each node corresponds to a user account and each edge represents a credit link between two accounts.

All of the aforementioned networks exhibit a significant community structure, as indicated by their high modularity value. To evaluate the effect of community structure, we thus generated a graph that has the same degree sequence as the Facebook graph, but randomly chosen endpoints for each edge (thus called *Randomized Facebook* in the following). The resulting graph has a much lower modularity value, indicating the absence of community structure. To investigate the impact of the network's degree sequence, we furthermore generated two graphs with the same number of nodes and approximately the same number of edges as the Facebook graph. The first graph, denoted as *Erdös-Renyi* (ER) in the following, has normal distributed degrees [16]. The degree sequence of the second graph, generated according to the Barabasi-Albert (BA) model [2], follows a power law. In contrast to the real-world graphs, the minimum degree of the BA graph is 13, giving it overall better connectivity. Note that all graphs but the ER graph have a power-law degree distribution.

## 5.2 Metrics

The *agreement ratio* quantifies the degree of consensus in a network: Given a social overlay $S = (V, E)$, let $n_{ID}(t)$ be the number of nodes that have set their *leader* value to *ID* at time $t$. The agreement ratio at time $t$ is then $\max_{ID \in \mathcal{ID}} \frac{n_{ID}(t)}{|V|}$, where $\mathcal{ID}$ denotes the set of all possible identifiers, as defined in Section 4. Informally, the agreement ratio measures the largest fraction of nodes that consider the same node as leader at time $t$. In the following, we call the term $\frac{n_{ID}(t)}{|V|}$ the *popularity* of *ID* at time $t$. If the agreement ratio is higher than 0.5, we say that the node owning the most popular ID is the leader at that point in time.

As described at the end of Section 4.4, an attacker aiming to establish a malicious node as leader may fail to do so in a reasonable amount of time if the system enters a metastable phase. However, the set of honest nodes that resists adoption of the malicious node's identifier may actually be very small. We thus consider the attacker to be successful if at the end of the simulation, more than 50% of all honest nodes hold the identifier of a malicious node in their *leader* variable. If this is the case, we say that the election *failed* in this run. Otherwise, we say that the election was *successful.* For a given number $n$ of simulation runs on a given network and a fixed set of victim nodes, we approximate the probability that the majority of honest nodes quickly adopts a malicious node as leader by the failure ratio $\frac{n_f}{n}$, where $n_f$ denotes the number of failed runs.

## 5.3 Simulation model

Our simulation model is based on the OMNeT++ [34] framework. We implemented Algorithm 1 and the above metrics pertaining to its evaluation.

At the begin of a simulation run, we first create a network of nodes according to the given network topology. Afterwards, each node once generates a random 32-bit identifier that also corresponds to its initial *leader* value. We then run Algorithm 1, excluding cryptographic operations. As we consider a computationally bounded adversary, the adversary is unable to break a computationally secure digital signature scheme. Thus, we disregard attacks on the cryptographic algorithms in our simulation and hence omit cryptographic operations as they are irrelevant for the metrics of interest.

When simulating an attack, we modelled the attacker as a single malicious node to indicate collusion. After picking $g$ nodes, we connected all of them to the attacker. In the following, we call an honest node that is connected to a malicious node a *victim node.* The degree of victim nodes can potentially affect the effectiveness of the attack as connecting to high-degree nodes indicates a more central position in the network. Thus, our simulations evaluated two strategies for choosing victim nodes: i) uniformly at random without replacement and ii) nodes with the highest degree. If there was more than one possible set for the $g$ nodes with the highest degree for the considered graph, we only used a single set for our simulations due to time constraints for our study.

During the simulation, the attacker node periodically broadcasts a single adversary-chosen random identifier value with a fresh timestamp. We consider this behavior to be realistic, because we focus on an adversary that aims to maximize the chance that a malicious node is elected quickly. Propagating different identifiers over different links or changing the propagated identifier over time causes these identifiers to compete with each other, which is unlikely to improve over the aforementioned strategy.

## 5.4 Parameters

For all simulation experiments, we used 40 as the expiry parameter $M$ introduced in Section 4.3. All nodes update their *leader* value

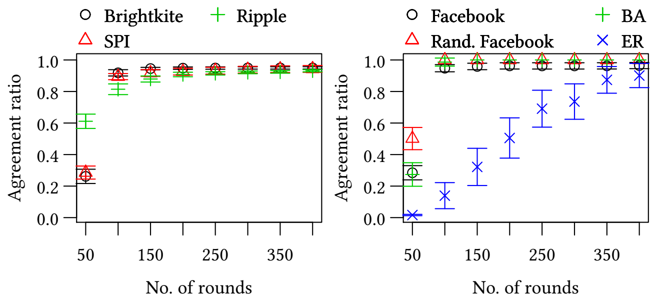

**Figure 4: Obtained mean value of agreement ratio at different round numbers. The bars around each point represent 99% confidence intervals.**

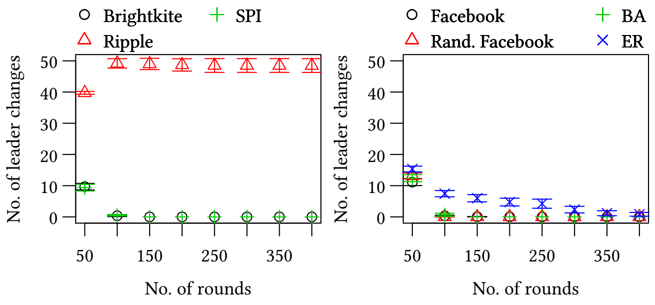

**Figure 5: Obtained mean number of leader changes at consecutive intervals of 50 rounds. The bars around each point represent 99% confidence intervals.**

at fixed intervals, which are chosen such that between two consecutive updates, all other nodes have updated their *leader* value exactly once. Consequently, the time needed to reach consensus depends linearly on the constants $\Delta_C$ and $\Delta_R$. We thus focused on the behavior of the system related to the number of synchronous rounds and set $\Delta_C$ and $\Delta_E$ to zero and $\Delta_D$ to one.

When evaluating attack resistance, we chose values from $\{50, 100, 150, 200, 250\}$ as the number of attack edges $g$.

The simulation was terminated after 400 simulated rounds and for the results in Section 5.5 averaged over 100 runs. The results in Section 5.6 are averaged over 50 runs.

## 5.5 Impact of network structure

In the following, we first present our results regarding the impact of the network's degree sequence on the convergence of the election. Afterwards, we discuss the effect of community structure.

*Degree sequence.* Figure 4 depicts the agreement ratio in relation to the number of rounds. If the degree sequence follows a power law, the election quickly achieves consensus among a large fraction of nodes. After 100 rounds, all power-law graphs but the Ripple graph on average reached an agreement ratio of above 90%. The Ripple graph also achieved a high agreement ratio of 81.4%. In contrast to networks with a power-law degree distribution, the leader election proceeded much slower for the ER graph with normal distributed degrees. After 200 rounds, just around 50% of nodes on average consider the same node as leader.

The reason for the slow convergence of the ER graph lies in the frequency of expiring timestamps. Shortly after the beginning of the election, even the node $u$ owning the most popular ID was likely to change its *leader* value to a different ID. While the former occurred on all graphs, the election generally progressed faster on the power-law graphs, such that $u$ re-adopted its own ID as *leader* value before the corresponding timestamps expired. Because the *leader* value of a node with high degree more strongly increases in popularity than the *leader* value of a node with low degree, the existence of few nodes with very high degree in power-law graphs results in a strong bias towards the *leader* value held by latter nodes, quickly ruling out other identifiers. Due to the lower variance in node degrees, the election on the ER graph requires more time until the most popular ID becomes sufficiently more popular than all other identifiers. Thus it was unlikely that the node owning the most popular ID re-adopts its identifier from a neighbor soon enough to prevent expiration of timestamps. Since social networks generally have a power-law degree distribution, in practice they are not affected by the slow convergence in networks with normal distributed degrees.

Although the agreement ratio on the Ripple graph increased nearly as fast as on the other power-law graphs, the most popular identifier changed almost every round throughout the simulation. For consecutive intervals of 50 rounds, Figure 5 shows the mean number of times the most popular ID changed in each interval for the different settings. Except for the Ripple graph, the number of leader changes dropped within the first 100 rounds, meaning that the leader chosen by the majority of nodes does not change anymore over time.

The oscillating behavior of the Ripple graph results from its strong Hub-and-Spoke structure where 71.25% of all nodes have a degree of 1 and three nodes have a degree higher than 10,000. If a node $u$ with an ID $ID$ as leader value has many neighbors with degree 1 that in turn have a different ID $ID'$ as *leader* value, it is highly likely that $u$ will adopt $ID'$ as *leader* value in the next round. At the same time however, given that $u$'s current $(leader, ts, sig)$-tuple is valid, every neighbor of $u$ with degree 1 will adopt $ID$ in the next round with probability 1. Given a high enough number of nodes with degree 1, the node owning $ID$ now replaced $ID'$ in its role as the leader. In the next round, the same behavior again causes the node owning $ID'$ to become the leader and so on.

However, since the Ripple graph represents a payment network, we believe that most of the nodes with only one neighbor represent inactive users that just performed one single transaction. For more popular overlay networks, we consider such a strong Hub-and-Spoke-structure to be unlikely and thus leave a corresponding adaptation of the algorithm for future work.

*Community structure.* Although the agreement ratio initially grows quickly for all networks whose degree sequence follows a power law, there are notable differences regarding its long-time behavior between the graphs with high modularity and those with low modularity. On the graphs with low modularity, namely the randomized Facebook graph and the BA graph, the agreement ratio reached 1.0 after at most 105 and 124 rounds, respectively for all 100 runs. On some runs on the graphs with high modularity, the election entered a *metastable phase* as described in Section 4.4.

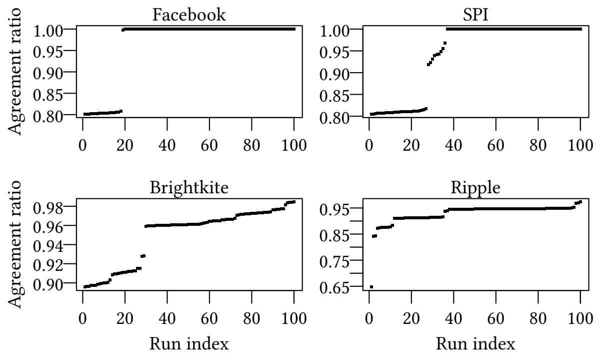

**Figure 6: Agreement ratio after 400 rounds for the different networks, ordered by agreement ratio.**

Figure 6 shows the agreement ratio at the end of each simulation run, i.e. after 400 rounds, for all graphs with high modularity. The results for the different runs are ordered according to the agreement ratio. Out of all 100 runs for the Facebook graph, the system entered a metastable phase at an agreement ratio around 0.8 in 18 runs. In all other runs, the agreement ratio at the end of the simulation was at least 0.998. In 20 runs, the agreement ratio reached 1.0 before the end of the simulation. The fact that there are no runs with an agreement ratio between 0.81 and 0.998 on the Facebook graph suggests that there is a community consisting of around 20% of all nodes which only has few edges to nodes outside this community. The runs that reached an agreement ratio between 0.998 and 1.0 furthermore indicate the presence of very small communities with few edges to the rest of the graph.

Similar to the Facebook graph, there is a notable of set of runs on the SPI graph where the network only reached an agreement ratio of 0.8, indicating a community with around 20% of all nodes. In 9 runs however, the agreement ratio was between 0.91 and 0.97 at the end of the simulation. During these runs, the agreement ratio quickly increased to a value of approximately 0.8 within the first 100 rounds and afterwards grew slowly, meaning that no notable metastable phase occurred. However, because most of the remaining nodes already agreed on a different leader, it took more time until they changed their *leader* value to the most popular identifier.

While for the SPI graph, the agreement ratio reached 1.0 in 63 out of 100 runs, it did not reach 1.0 in any of the runs on the Ripple and the Brightkite graph. As suggested by their higher modularity score, the Brightkite and the Ripple graph contain a notably stronger community structure than the Facebook graph and the SPI graph. However, our results show that still more than 80% of all nodes on these graphs are part of communities with sufficiently many edges between them, such that the weakly interconnected communities are comparatively small. For the Brightkite graph, the agreement ratio at the end of the simulation was at least 0.89 and the gap between 0.92 and 0.96 indicates that the largest community with few edges to other communities contains around 4% of all nodes. All other communities contain less than 1% of all nodes.

Similarly, all but one run on the Ripple graph reached an agreement ratio of at least 0.84 and the gaps displayed in Figure 6 suggest that the communities that failed to adopt the most popular identifier are rather small. In the run where the system only reached an agreement ratio of around 0.65, the timestamps related to the most popular identifier expired near the end of the simulation, causing a large number of nodes to reset their state.

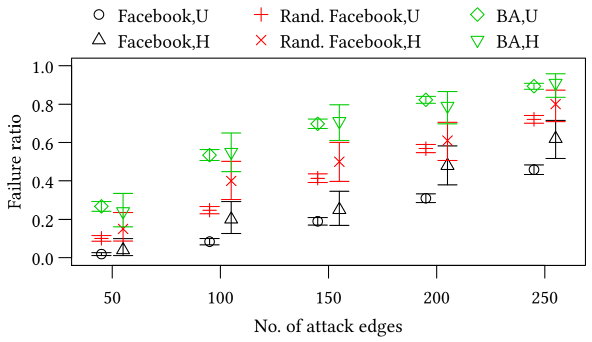

**Figure 7: Obtained mean failure ratios for the Facebook graph, its randomized version and the BA graph, for a varying number of attack edges and uniformly (U) as well as highest-degree (H) victims. The bars below and above each mean value represent** 95% **confidence intervals.**

*Summary.* Our results show that the structure of the network has a strong impact on the convergence of our leader election algorithm. In particular, a higher heterogeneity regarding node degrees drastically reduces the time needed until the majority of nodes agrees on a common leader. However, in the extreme case that around 70% of all nodes have only one neighbor, the actual leader identifier is likely to oscillate heavily.

Furthermore, the presence of weakly interconnected communities may cause the election to enter a metastable phase. While the community structure of the datasets used for our study often kept our algorithm from reaching consensus among *all* nodes, still more than 80% of all nodes quickly reached consensus on a single leader.

## 5.6 Attack resistance

Now, we address the attack resistance of Algorithm 1 when malicious nodes aim to bias the election towards choosing a malicious node as the leader. Our evaluation focuses on the Facebook graph, its randomized version as well as the BA graph. We excluded Ripple and the ER graph as Algorithm 1 is unsuitable for these graphs even in the absence of malicious nodes, as indicated by Section 5.5.

Figure 7 shows the obtained failure ratios for a varying number $g$ of victim nodes. For randomly selected victim nodes, each point denotes the mean value of the failure ratio over 30 different randomly chosen sets of $g$ nodes. For each of these sets, we performed 50 simulation runs to approximate the failure ratio for this particular set of victim nodes. For maximum-degree victim nodes, there is only one set of $g$ nodes and we performed 100 runs to approximate the failure ratio for this set. The difference in the confidence between the two cases hence follows from the different total number of runs, as the decisive factor of randomness seems to be in the voting process rather than in the selection of the victim set.

Irrespective of how the victim nodes were chosen, there is a notable difference regarding the mean failure ratio between the different graphs. For every simulated number of attack edges, the mean failure ratio for the Facebook graph was significantly lower than for the two synthetic graphs. However, the decrease of the failure ratio over the randomized Facebook graph due to the presence of community structure is negligible. In the scenario that victim nodes were selected randomly, the system entered a metastable phase where the popularity of the malicious node's ID remained close to 0.2 in at most 18 out of the total of 1500 runs for each number of edges. When the highest degree nodes were chosen as victim, such a metastable phase occurred only in at most 3 out of the 100 runs for each number of edges. From these results, we conclude that the presence of community structure does not significantly contribute to the attack resistance of our algorithm.

Instead, the mean failure ratio shows a strong negative correlation with the average shortest path length of the network. A high average shortest path length causes a slower spread of the malicious ID and this might indeed be related to the attack resistance.

When the number of attack edges was at most 200, the selection of high degree nodes as victims did not result in a significant increase for the mean failure ratio over random selection. While the difference for 250 attack edges is significant, there is no general rule that preferring high-degree nodes as victims increases the strength of the attack. We furthermore consider it to be unrealistic in practice that an attacker can compromise such a number of the highest degree nodes.

*Summary.* The second part of our evaluation showed that the presence of community structure does not contribute significantly to the attack resistance of our algorithm. Our results suggest that the probability that a malicious node becomes the leader is positively correlated with the average shortest path length of the network. For the investigated real-world graph, an adversary without knowledge of the network structure needs to establish at least 250 attack edges to achieve a 50% chance that one of his nodes will be elected as leader. Furthermore, it turned out that the degree of the victim nodes does not play a decisive role for the chance that a malicious node becomes the leader.

## 6 CONCLUSION

In this work, we proposed a leader election algorithm for large-scale dynamic networks that utilizes three-majority voting to achieve consensus among the majority of nodes. Rather than requiring a low number of malicious nodes, our solution works if the number of links between honest and malicious nodes is low. Our algorithm employs cryptographically signed timestamps to react to failed leaders and prevent impersonation. An extensive simulation study indicates fast consensus for the majority of nodes in social networks as well as a high resistance to attacks.

An interesting area for future work is to combine our leader election with a distributed spanning tree protocol, which can be then be leveraged for privacy-preserving, efficient, and secure routing [30].

## 7 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. This work has in parts been funded by the German Research Foundation (DFG) Grant STR 1131/2-2 and as part of Germany's Excellence Strategy – EXC 2050/1 CeTI – Project ID 390696704. We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for allocations of computer time.

## REFERENCES

[1] John Augustine, Gopal Pandurangan, and Peter Robinson, *Fast byzantine leader election in dynamic networks*, Symposium on Distributed Computing, 2015.
[2] Albert-László Barabási and Réka Albert, *Emergence of scaling in random networks*, Science (1999), 509–512.
[3] Luca Becchetti et al., *Stabilizing consensus with many opinions*, Symposium on Discrete algorithms, 2016.
[4] ———, *Simple dynamics for plurality consensus*, Distributed Computing (2017), 293–306.
[5] Petra Berenbrink et al., *Bounds on the voter model in dynamic networks*, Colloquium on Automata, Languages and Programming, 2016.
[6] ———, *Ignore or comply?: On breaking symmetry in consensus*, Symposium on Principles of Distributed Computing, 2017.
[7] Vincent D Blondel et al., *Fast unfolding of communities in large networks*, Journal of statistical mechanics: theory and experiment (2008), P10008.
[8] Eunjoon Cho et al., *Friendship and mobility: user movement in location-based social networks*, Knowledge discovery and data mining, 2011.
[9] Ian Clarke, Oskar Sandberg, Matthew Toseland, and Vilhelm Verendel, *Private communication through a network of trusted connections: The dark freenet*, https://freenetproject.org/assets/papers/freenet-0.7.5-paper.pdf, 2010.
[10] Colin Cooper, Robert Elsässer, and Tomasz Radzik, *The power of two choices in distributed voting*, Colloquium on Automata, Languages, and Programming, 2014.
[11] Colin Cooper, Tomasz Radzik, Nicolás Rivera, and Takeharu Shiraga, *Fast plurality consensus in regular expanders*, Symposium on Distributed Computing, 2017.
[12] Emilio Cruciani et al., *Distributed community detection via metastability of the 2-choices dynamics*, Conference on Artificial Intelligence, 2019.
[13] Danny Dolev and Andrew Yao, *On the security of public key protocols*, Transactions on Information Theory (1983), 198–208.
[14] Qi Dong and Donggang Liu, *Resilient cluster leader election for wireless sensor networks*, Sensor, Mesh and Ad Hoc Communications and Networks, 2009.
[15] John R Douceur, *The sybil attack*, Workshop on peer-to-peer systems, 2002.
[16] P Erdös and A Rényi, *On random graphs i*, Publ. Math. Debrecen (1959), 290–297.
[17] Uriel Feige, *Noncryptographic selection protocols*, FOCS, 1999.
[18] Mohsen Ghaffari and Johannes Lengler, *Nearly-tight analysis for 2-choice and 3-majority consensus dynamics*, Principles of Distributed Computing, 2018.
[19] Ronen Gradwohl, Salil Vadhan, and David Zuckerman, *Random selection with an adversarial majority*, International Cryptology Conference, 2006.
[20] Charles M Grinstead and J Laurie Snell, *Markov chains*, Introduction to probability (1997).
[21] Yehuda Hassin and David Peleg, *Distributed probabilistic polling and applications to proportionate agreement*, Information and Computation (2001), 248–268.
[22] Bruce M Kapron et al., *Fast asynchronous byzantine agreement and leader election with full information*, Transactions on Algorithms (2010), 68.
[23] Valerie King et al., *Towards secure and scalable computation in peer-to-peer networks*, Symposium on Foundations of Computer Science, 2006.
[24] Jure Leskovec et al., *Statistical properties of community structure in large social and information networks*, Conference on World Wide Web, 2008.
[25] David A Levin and Yuval Peres, *Markov chains and mixing times*, vol. 107, American Mathematical Soc., 2017.
[26] Alan Mislove et al., *Measurement and analysis of online social networks*, Conference on Internet Measurement, 2007.
[27] Prateek Mittal, Matthew Caesar, and Nikita Borisov, *X-vine: Secure and pseudonymous routing in dhts using social networks.*, NDSS, 2012.
[28] Mark EJ Newman and Juyong Park, *Why social networks are different from other types of networks*, Physical review E (2003), 036122.
[29] Thomas Paul et al., *The students portal of ilmenau: A holistic osns user behaviour model*, Tech. report, Palestine Polytechnic University, 2016.
[30] Stefanie Roos, Martin Beck, and Thorsten Strufe, *Anonymous addresses for efficient and resilient routing in f2f overlays*, INFOCOM, 2016.
[31] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg, *Settling payments fast and private: Efficient decentralized routing for path-based transactions*, Networks and Distributed Systems Security, 2018.
[32] Alexander Russell and David Zuckerman, *Perfect information leader election in $\log^* n$+o(1) rounds*, Journal of Computer and System Sciences (2001), 612–626.
[33] Michael Sirivianos et al., *Non-manipulable aggregator node election protocols for wireless sensor networks*, Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007.
[34] Andras Varga, *OMNeT++ Discrete Event Simulator*, https://omnetpp.org/, November 2018.
[35] E. Vasserman et al., *Membership-concealing Overlay Networks*, CCS, 2009.
[36] Bimal Viswanath et al., *On the evolution of user interaction in facebook*, Workshop on Online social networks, 2009.
[37] Gicheol Wang and Gihwan Cho, *Securing cluster head elections in wireless sensor networks*, Journal of Advances in Computer Networks (2014), 243–247.