Decentralized Polling With Respectable Participants

Rachid Guerraoui, Kévin Huguenin, Anne-Marie Kermarrec, Maxime Monod

Decentralized Polling With Respectable Participants

Rachid Guerraoui[1], Kévin Huguenin[2],
Anne-Marie Kermarrec[3], and Maxime Monod[1⋆]

[1] EPFL
[2] Université de Rennes 1 / IRISA
[3] INRIA Rennes - Bretagne Atlantique

Abstract. We consider the polling problem in a social network where participants care about their reputation: they do not want their vote to be disclosed nor their misbehaving, if any, to be publicly exposed. Assuming this reputation concern, we show that a simple secret sharing scheme, combined with verification procedures, can efficiently enable polling without the need for any central authority or heavyweight cryptography.

More specifically, we present DPol, a simple and scalable distributed polling protocol where misbehaving nodes are exposed with a non-zero probability and the probability of dishonest participants violating privacy is balanced with their impact on the accuracy of the polling result. The trade-off is captured by a generic parameter of the protocol, an integer k we call the *privacy parameter*, so that in a system of N nodes with $B < \sqrt{N}$ dishonest participants, the probability of disclosing a participant's vote is bounded by $(B/N)^{k+1}$, whereas the impact on the polling result is bounded by $(6k+2)B$.

We report on the deployment of DPol over 400 PlanetLab nodes. The polling result suffers a relative error of less than 10% in the face of message losses, crashes and asynchrony inherent in PlanetLab. In the presence of dishonest nodes, our experiments show that the impact on the polling result is $(4k+1)B$ on average, consistently lower that the theoretical bound of $(6k+2)B$.

1 Introduction

Social networks are growing exponentially, and one of the most celebrated examples, Facebook, currently boasts more than 200 million active users. Many of these users regularly share images and videos as well as discuss various social and political matters. They do so both with close friends and people they hardly know. A particularly important task in such networks is *polling*, such as the recent one about the terms of service of Facebook – initiated by Facebook managers [1], or the organizers of a Saturday night party asking in a social group

⋆ Maxime Monod has been partially funded by the Swiss National Science Foundation with grant 20021-113825.

whether partners should be invited too. In many cases, such a polling can be expressed in a *binary* form: each participant starts with $+1$ or -1, expressing the answer to a question, and the goal is to compute the sum of the initial values. To be meaningful, a polling protocol should tolerate dishonest participants trying to bias the polling or discover other participants' votes.

An easy way to conduct a poll is to use a central server (e.g., Facebook Poll [2]). Each participant sends its vote to a central entity, which subsequently aggregates all votes and computes the outcome. Beside the non-scalability of this solution however, privacy is not ensured as participants might generally not want their vote (and maybe even the subject of the poll or the result) to be seen by a central entity, be it trusted or not [3]. Distributed *aggregation* is a simple, yet naive, alternative to avoid a central server: participants aggregate votes so that once these are summed up, it is impossible to know the vote of a participant. However, since participants contribute to the outcome's computation, they may bias the result by corrupting intermediary results. To prevent the initial bootstrapping vote to be known, every vote could be split (homomorphic secret sharing). However, a dishonest participant can still create an invalid initial set of shares (e.g., voting for an arbitrary large value) and bias the result.

Not surprisingly, devising a distributed polling protocol that ensures privacy while tolerating dishonest participants that might want to bias the votes is very challenging. This is particularly true if the goal is to devise a practical, hence simple, peer-to-peer protocol that does not rely on any heavyweight cryptography (e.g., asymmetric cryptography). The motivation of this work is to address this challenge by exploiting the special nature of social networks. A defining characteristic of such networks is the one to one mapping between social network identities and real ones (as opposed to virtual world platforms such as SecondLife). Participants in such networks do care about their *reputation*: information related to a user is intimately considered to reflect the associated *real* person. We leverage this concern and propose an approach which, instead of masking (e.g., BFT [4]) or preventing (e.g., cryptography) dishonest behaviors, dissuades such behaviors. This is achieved by executing, in addition to the polling algorithm, a distributed verification protocol which tags the profiles of the participants. For instance, if the testimonies of Alice and Bob demonstrate that Mallory misbehaved, their profiles are tagged with "Alice and Bob jointly accused Mallory" and the profile of Mallory is tagged with "Mallory has been accused by Alice and Bob". No participant would like to be tagged as dishonest (by a protocol that does not wrongly accuse participants as we will describe below). Moreover, assuming a system with a large majority of honest participants, the risk for a participant to be caught wrongly accusing others is high. For instance, if a participant is accused only by users that are related in the social network (i.e., friends forming a coalition), the accusation would be suspect and thus not be taken into account and this would eventually backfire on the accuser. In a social network, this kind of attacks can indeed be easily detected by a human reader or by an automated graph analysis tool inspired from SybilLimit [5].

We consider a system of both honest and dishonest participants. The former follow the protocol assigned to them while the latter might not. Should they deviate from the protocol, they never do anything that will jeopardize their reputation with certainty (i.e., with probability 1). In this context, we present DPol, a simple decentralized polling protocol. In a nutshell, DPol works as follows. Participants, clustered in fully connected groups, known as *offices*, make use of a simple secret sharing scheme to encode their vote. Then they send the shares of their vote (*ballots*) to proxies, belonging to another group (an office). Each office computes a partial tally that is further broadcast to all other groups. Each participant eventually outputs the same tally. DPol is fully decentralized and does not assign specific roles to any participant. This results in a simple, scalable, and easy to deploy protocol. The spatial complexity of DPol is $\mathcal{O}(\sqrt{Nk})$ and the number of messages sent is $\mathcal{O}(\sqrt{Nk})$, k being the *privacy parameter* and N the number of participants.

We bound the impact of dishonest participants and balance this with the level of privacy ensured. More specifically, in a system of N participants with $B < \sqrt{N}$ dishonest participants (which is a reasonable assumption in a social network with a limited number of sybil identities [5]), we can choose any integer k such that the probability for a given participant to have its vote recovered by dishonest participants is bounded by $(B/N)^{k+1}$ and the impact on the result is bounded by $(6k+2)B$. This is due to our underlying simple secret sharing scheme enabling to expose, with certainty, dishonest participants affecting the outcome more than $6k+2$ with only public verifications, i.e., without requiring to reveal the participants' vote. As we show in the paper, private verifications expose, with a non-zero probability, dishonest participants further (i.e., even if their impact is less than $6k+2$), but require to inspect the content of a subset of ballots.

Consider for illustration a system of $10,000$ participants with 99 dishonest participants ($\lceil\sqrt{N}\rceil - 1$) and assume a proportion α of participants voting $+1$. For instance, setting $k = 1$ ensures privacy with probability 99.99% and for any $\alpha > 0.54$, 100% of the participants compute the right binary decision (i.e., the sign of the outcome). While e-voting requires stronger guarantees, this amply fits polling applications requirements.

DPol is indeed easy to deploy and we report on its deployment over 400 PlanetLab participants. The polling result suffers a relative error of less than 10% in the face of message losses, crashes and asynchrony inherent in PlanetLab. In the presence of dishonest participants, our experiments show that the impact on the polling result is $(4k+1)B$ on average, consistently lower that the theoretical bound of $(6k+2)B$.

The rest of the paper is organized as follows: Section 2 describes our system model. Section 3 gives a detailed description of DPol together with its formal analysis. The impact of dishonest participants is presented in Section 4. Experimental results from PlanetLab are reported in Section 5. Section 6 reviews related work and Section 7 concludes the paper and proposes perspectives on future work.

2 System Model

We consider a system of N uniquely identified nodes representing participants. Each node p votes for a binary value $v_p \in \{-1, +1\}$ and the expected output of the polling algorithm is $\sum_p v_p$. Each participant in the social network is assigned a profile that can be tagged by DPol.

Nodes can either be honest or dishonest. Honest nodes strictly follow the protocol and contribute to the verifications as long as their privacy is not compromised. More specifically, honest nodes always collaborate with verification procedures that do not require them to reveal their ballots (i.e., public verifications). However, they may refuse to reveal their ballots for a verification procedure (i.e., private verification). Dishonest nodes may misbehave either to promote their opinion or reveal the opinion of honest nodes. Yet, they are rational in the sense that they never behave in such a way that their reputation is tarnished with certainty, i.e., they do not perform attacks that can be detected with probability 1 by means of public verification procedures. Dishonest nodes do not wrongfully blame honest nodes as it is rather easy for a human reader, when looking at other users' profiles, to distinguish between legitimate and wrongful blames, e.g., a single node blaming at random a very large number of nodes or a group of nodes always blaming together. We consider colluding dishonest nodes as a single coalition $\mathcal{B}$ ($|\mathcal{B}| = B$). When dishonest nodes collaborate to bias the outcome of the poll, they are assumed to share the same opinion. Still, they act selfishly in the sense that they prefer protecting their own reputation rather than covering up their suspected accomplices. For the sake of readability, we consider that the coalition always promotes -1 in the rest of the paper. A single coalition represents the worst case scenario for both discovering a node's vote and biasing the result of the polling.

Theoretical analysis (Section 3.3) assumes reliable channels and non-faulty nodes. We revisit these assumptions by measuring the impact of message losses and crashes in the implementation (Section 5).

In order to make DPol scalable and to allow for efficient verifications we assume a structured overlay which could be provided by the social network infrastructure. Note that the overlay is independent from the social graph. The N nodes are clustered into r ordered groups, from g_0 to g_{r-1}. A node p in group g_i maintains two sets of nodes: a set $\mathcal{P}_o$ of *officemates* containing all nodes belonging to the same group ($\mathcal{P}_o = g_i \backslash \{p\}$) and a fixed-size set $\mathcal{P}_p$ of *proxies*, containing nodes in the next group ($\mathcal{P}_p \subseteq g_{i+1 \bmod r}$). Therefore, all groups virtually form a ring, g_1 being the successor of g_r. Each group g_i is a clique. We define a *client* of a node p, a node for which p acts as a proxy. Every node maintains a list of its clients in the previous group ($\mathcal{P}_c \subseteq g_{i-1}$). A node discards every message originating from a node that is not in $\mathcal{P}_c \cup \mathcal{P}_o$. We assume a random uniform distribution of nodes across the r groups and nodes in the successor groups are distributed uniformly at random as proxies in the predecessor groups.

3 The Polling Protocol

We first give an overview of DPol. Then we prove its correctness and provide a theoretical analysis of its complexity, considering only honest nodes. Finally, we analyze how the protocol resists a privacy attack, i.e., dishonest nodes recovering the vote of a node. Attacks on the outcome of the poll, i.e., dishonest nodes biasing the result, are discussed in Section 4.

3.1 Polling in a nutshell

DPol is composed of three phases: *(i)* voting, *(ii)* counting and *(iii)* broadcasting. During the voting phase, a node generates a set of ballots reflecting its opinion and sends each ballot to one of its proxies. In the counting phase, each node in a group computes the sum of the votes of the nodes in the previous group (local tally). This is achieved by having each proxy summing up the ballots it has received and broadcasting the result to its officemates. Finally, the local tallies are forwarded along the ring so that all nodes eventually compute the final outcome.

Algorithm 1 DPol at node p in group g_i, $i \in \{1, \ldots, r\}$

Input: a vote $v \in$ {-1, +1}
Variables: an individual tally $t'' = 0$
a local tally $t' = 0$
an array of local tally sets $\mathcal{S}[\{1, \ldots, r\} \to \varnothing]$
a local tally array $\mathcal{T}[\{1, \ldots, r\} \to \perp]$
Output: the global tally $\hat{t}$

Polling Algorithm

1: vote(v, $\mathcal{P}_p$)
2: local_count(t'', $\mathcal{P}_p$)
3: $t' = t' + t''$
4: local_tally_broadcast(i, t'', $\mathcal{P}_p$)
5: $\hat{t} = \sum_i \mathcal{T}[i]$

Voting phase

procedure vote(v,$\mathcal{P}_p$) **is**
6: $b = v$
7: **for each** $proxy \in \mathcal{P}_p$ **do**
8: send [BALLOT, b] ($proxy$)
9: $b = -b$
10: **end for**

upon event ⟨ receive | [BALLOT, b] ⟩ **do**
11: $t'' = t'' + b$

Intermediate Counting phase

procedure local_count(t'',$\mathcal{P}_o$) **is**
12: **for each** $officemate \in \mathcal{P}_o$ **do**
13: send [INDIVIDUALTALLY, t''] ($officemate$)
14: $b = -b$
15: **end for**

upon event ⟨ receive | [INDIVIDUALTALLY, t] ⟩ **do**
16: $t' = t' + t$

Local Tally Broadcasting & Forwarding phase

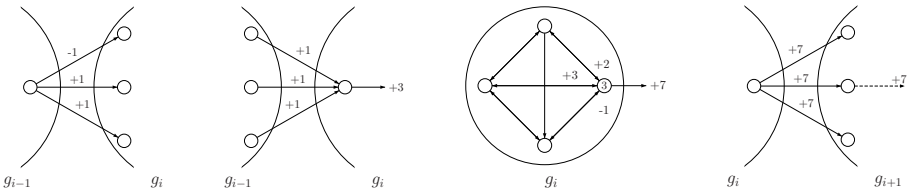
procedure local_tally_broadcast(i, t'', $\mathcal{P}_p$) **is**
17: **for each** $proxy \in \mathcal{P}_p$ **do**
18: send [LOCALTALLY, i, t'] ($proxy$)
19: **end for**

upon event ⟨ receive | [LOCALTALLY, i_{group}, t] ⟩ **do**
20: $\mathcal{S}[i_{group}] = \mathcal{S}[i_{group}] \cup \{t\}$
21: **if** ($\mathcal{S}[i_{group}] = |\mathcal{P}_c|$) **then**
22: $\mathcal{T}[i_{group}] =$ choose($\mathcal{S}[i_{group}]$)
23: local_tally_broadcast(i,$\mathcal{T}[i_{group}]$)
24: **end if**

function choose($\mathcal{A}$) **returns** local tally **is**
25: **return** the most represented local tally in $\mathcal{A}$

3.2 Description

Voting The ballot generation is inspired by the simple secret sharing scheme introduced in [6] and shares similarities with the Vote/Anti-Vote/Vote system [7]. In order to vote for a given value $v \in$ {-1, +1}, a node generates $2k + 1$ *ballots* $b_1, \ldots, b_{2k+1} \in$ {-1, +1} representing its vote, where k is an integer called privacy parameter. The intuition is to create $k + 1$ ballots of a given tendency (positive or negative) and k opposite ballots, resulting, when summed, in a single vote



(a) Voting (b) Intermediate counting (c) Individual tally sharing (d) Local tally broadcasting & forwarding

Fig. 1. Key phases of DPol. (a) A node in g_{i-1} generates 3 ($k = 1$) ballots $\{-1, +1, +1\}$ and sends them to its proxies in g_i. (b) A node in g_i collects its received ballots $\{+1, +1, +1\}$ and sums them to $+3$ (individual tally) that it shares with its officemates in g_i as depicted in (c). (c) A node receives every expected individual tallies $\{-1, +3, +2\}$, computes and sends the local tally ($+7$) to its proxies in the next group g_{i+1} as depicted in (d). (d) These proxies forward the local tally to theirs in g_{i+2}.

$v' = v$. Each ballot is defined as $b_j = v$ if j is odd and $b_j = -v$ if j is even, so that $v' = \sum_{j=1}^{2k+1} b_j = v$.

Once a node has generated its $2k + 1$ ballots, it sends each of them to a different proxy. The number of proxies is to be chosen accordingly, $|\mathcal{P}_p| = 2k+1$. Lines 6–10 in Algorithm 1 detail the voting phase. Figure 1(a) depicts a node sending its $2k+1$ ballots (e.g., $\{-1, +1, +1\}$) to its assigned proxies. Once every node in the system has received one ballot from each of its clients, the voting round is over.

Intermediate Counting A group acts as a voting office for the preceding group on the ring. The officemates collect ballots from their clients (Figure 1(b)) and share intermediate results (Figure 1(c)). To this end, a proxy sums the ballots it received into an *individual tally* t'', as described in Algorithm 1, line 11. Once a node has received the expected number of ballots from its clients, it broadcasts the computed individual tally to its officemates, as depicted in Figure 1(b) (lines 12–15 in Algorithm 1). The officemates aggregate the received data, i.e., they sum each others' individual tallies and store the result into a *local tally* t' as shown in Figure 1(c) (line 16 in Algorithm 1).

Local Tally Forwarding Once the intermediate counting phase is over, i.e., all the officemates have computed a local tally, each node sends its local tally to its proxies (lines 17–19 in Algorithm 1). Upon reception of a message containing a local tally, a proxy adds it to the set $\mathcal{S}[i]$ of possible values for g_i. When a proxy has received the expected number $|\mathcal{P}_c|$ of local tallies for a given group g_i, it decides on a local tally by choosing the most represented value in $\mathcal{S}[i]$ and stores it in $\mathcal{T}[i]$. When a local tally $\mathcal{T}[i]$ is assigned, it is further forwarded (Figure 1(d)) to the next group using the proxies (lines 20–24 in Algorithm 1).

Local tallies are then forwarded in the system along the ring. When a local tally reaches its source (the group that emitted it), it is no longer forwarded. The global tally is computed at each node by simply summing the local tallies of all groups: $\hat{t} = \sum_{i=1}^{r} \mathcal{T}[i]$ (line 5 of Algorithm 1).

3.3 Analysis

We analyze here the correctness and complexity of DPol assuming only honest nodes. We then consider the impact of the dishonest nodes on privacy. The impact of dishonest attacks on the accuracy of the polling is presented in Section 4.

Theorem 1 (Correctness). *Assume a system where each node p starts with a binary value v_p. The polling algorithm terminates and each node eventually outputs $\sum_p v_p$.*

Proof. (Accuracy) We first prove that the local tally computed in every group g_i reflects the vote of all nodes in g_{i-1}. The local tally computed in a group is the sum of the ballots received by its members. Each node p in g_{i-1} sends each of its ballots $b_{1,p}, \ldots, b_{2k+1,p}$ to one distinct proxy in g_i. Similarly, each proxy p' in g_i receives a set of ballots $\mathcal{B}_{p'}$ from its clients. Since we assume only honest nodes, the set of ballots sent by the nodes in g_{i-1} is the set of ballots received in g_i. Therefore, the local tally computed by each member of g_i is $t' = \sum_{p' \in g_i} \sum_{b \in \mathcal{B}_{p'}} b = \sum_{p \in g_{i-1}} \sum_{j=1}^{2k+1} b_{j,p} = \sum_{p \in g_{i-1}} [(k+1) \cdot v_p + k \cdot (-v_p)] = \sum_{p \in g_{i-1}} v_p$. Note that this follows from the homomorphic property of the simple secret sharing scheme. Since nodes do honestly forward the local tallies along the ring and the messages are eventually received, each node ends up with the correct values for the local tallies of every group, thus the correct global tally.

(Termination) A node knows the number of messages it is supposed to receive in each phase. Since every node sends the required number of messages and every message eventually arrives, each phase completes. As the algorithm is a finite sequence of phases, it is guaranteed to eventually terminate.

In a realistic scenario, crashes and message losses do occur and may affect correctness and termination. Failures of nodes and message losses may *(i)* affect the accuracy of the global tally, and *(ii)* prevent nodes from detecting the end of a phase or deciding on a local tally.

Proposition 1 (Spatial complexity). *The asymptotic size S of the state maintained at each node in group g_i is $\mathcal{O}(r \cdot k + |g_i|)$.*

Proof. A node maintains a set of proxies ($2k+1$), the set of its officemates ($|g_i|$) and the list of its clients (at most $|g_{i-1}|$). Additionally, a node stores a set of $2k+1$ possible values for each of the r local tallies to perform global counting, that is $S = \mathcal{O}(k) + \mathcal{O}(|g_i|) + \mathcal{O}(|g_{i-1}|) + \mathcal{O}(r \cdot k) = \mathcal{O}(r \cdot k + |g_i|)$.

Proposition 2 (Message complexity). *The asymptotic average number of messages M sent by a node in group g_i is $\mathcal{O}(r \cdot k + |g_i|)$.*

Proof. A node sends messages during the voting phase ($2k + 1$ ballots), the intermediate counting phase ($|g_i| - 1$ individual tallies), and the global counting phase which involves the dissemination of r local tallies along the ring using its $2k + 1$ proxies, that is $M = \mathcal{O}(k) + \mathcal{O}(|g_i|) + \mathcal{O}(r \cdot k) = \mathcal{O}(r \cdot k + |g_i|)$.

Note that the parameters are not independent: the sizes of the groups are related and bound to the number of groups by the relation $\sum_{i=1}^{r} |g_i| = N$. The two quantities M and S are minimized when $r = \sqrt{N/k}$ and $|g_i| = \sqrt{Nk}$, and thus $M = S = \mathcal{O}(\sqrt{Nk})$.

Theorem 2 (Privacy). *The probability, for a given node, to have its vote recovered by a coalition of B dishonest nodes is bounded by $(B/N)^{k+1}$.*

Proof. The vote of a node is recovered by the dishonest nodes if and only if the $k + 1$ proxies that received the $k + 1$ ballots containing the most represented value collude. This event occurs with probability $\binom{B}{k+1}/\binom{N}{k+1}$. For all k, B and N, this probability is bounded by $(B/N)^{k+1}$.

4 Confining the impact of dishonest nodes

In this section, we analyze the impact of dishonest nodes on DPol, assuming companion verification schemes to detect attacks and identify dishonest nodes. Detection is performed by the nodes themselves relying on some of them being honest, and verifications are performed by an external entity (e.g., the social network infrastructure), polling nodes to identify the dishonest ones. We distinguish two types of verifications: *(i)* public verifications that leverage only information that does not compromise the nodes' privacy (i.e., the content of the ballots), such as the individual tallies received from their officemates, and *(ii)* private verifications that may leverage all information including the content of the ballots.

To dissuade nodes from misbehaving, verifications must affect the profiles of the involved nodes. When an attack is detected and reported, the neighbors of the accused nodes (i.e., the nodes it communicates with, typically clients and proxies) are polled for the messages they exchanged. If the testimonies of p_1 and p_2 demonstrate that p_0 misbehaved, their profile is tagged with "p_1 and p_2 jointly accused p_0" and the profile of p_0 is tagged with "p_0 has been accused by p_1 and p_2".

We consider an overlay of $\sqrt{N}$ groups of size $\sqrt{N}$ ($\sqrt{N} \in \mathbb{N}$) and a perfect client/proxy matching, i.e., each node has exactly the same number $(2k + 1)$ of clients and proxies.

In a first step, we assume that honest nodes do not want to disclose any of the ballots they sent or received (i.e., public verifications). In this context, we study the impact of colluding dishonest nodes.

In a second step, we assume that honest nodes are willing to sacrifice privacy for the sake of accuracy by revealing some ballots (i.e., private verification) and we prove that colluding nodes compromising the global tally within the bound may be caught with a non-zero probability.

4.1 Impact of a dishonest coalition under public verifications

Theorem 3. *For $B < \sqrt{N}$, each member of a dishonest coalition may affect the global tally up to $6k + 2$.*

Proof (Proof (structure)). The proof relies on the facts that *(i)* honest nodes always tell the truth and strictly follow the protocol (including verifications), and *(ii)* dishonest nodes do not behave in such a way that their reputation is decreased with certainty. Effectively, showing that the attacks with unbounded impact are detected by the honest nodes with probability 1 proves the theorem. A dishonest node may bias the protocol at all three phases. Lemmas 1-4 encompass all possible attacks, propose a detection scheme relying on honest nodes, and bound the impact of those that cannot be detected with probability 1. In addition, if an attack is detected, we prove that the dishonest node is exposed by the public verification. Summing the impacts of all these attacks (Lemmas 1 and 2) for each dishonest node gives a maximum impact $(2k + 2(2k + 1)) = (6k + 2)$ on the global tally.

Note that the proof relies on the assumption $B < \sqrt{N}$ to ensure that the dishonest coalition can neither "control" (there is at least one honest node in each group) nor "fool" an entire group without being detected (there are not enough dishonest nodes to both perform and cover dishonest actions). In fact, the weakest assumption needed is that $\forall i, |g_i \cap \mathcal{B}| + |g_{i+1} \cap \mathcal{B}| < \sqrt{N}$.

Lemma 1 (Voting). *A dishonest node can affect the global tally up to $2k$, when voting.*

Proof. Due to the overlay structure, a node can only send ballots to the proxies it is assigned (otherwise the ballots are discarded), i.e., a maximum of $2k + 1$ ballots. Therefore a dishonest node may affect the global tally by either *(i)* sending less ballots than it is supposed to or by *(ii)* sending more than $k + 1$ -1-ballots. In the worst case, the dishonest node sends $2k + 1$ -1-ballots, i.e., $-2k - 1$. Since a node voting -1 is supposed to send $k + 1$ -1-ballots and k +1-ballots, its maximum impact is $2k$.

Lemma 2 (Computing individual tallies). *There exists a public verification scheme so that, if a dishonest node modifies its individual tally by more than $2(2k + 1)$, the attack is detected with probability 1 and the node is exposed.*

Proof. The considered overlay structure ensures that all nodes have exactly $2k+1$ clients and thus receives $2k+1$ ballots during the voting phase. A dishonest node can modify its individual tally by inverting the +1-ballots it received, i.e., it turns them in -1-ballots, decreasing its individual tally accordingly. In addition, a dishonest node can also forge ballots. The latter attack is identified by its honest officemates if the individual tally is out of the range $[-(2k + 1), 2k + 1]$. Therefore, not to be publicly detected, a node corrupting or forging ballots must output an individual tally in that range. Consequently, the worst case occurs when a dishonest node receives $2k + 1$ +1-ballots and inverts them all when summing them, leading to a maximum impact of $2(2k + 1)$.

Lemma 3 (Broadcasting individual tallies). *There exists a public verification scheme so that, a dishonest node broadcasting inconsistent copies of its individual tally to honest nodes, i.e., sending different values to its honest officemates, is detected with probability* 1 *and the node is exposed.*

Proof. Before deciding on a local tally, every node broadcasts to its officemates the set of individual tallies it received. This way, an honest officemate trivially detects the inconsistency. Dishonest nodes are exposed when their neighbors are asked for the individual tallies they received from these nodes.

Note that broadcasting different individual tallies can help a dishonest node to impose an arbitrary value for the local tally. For instance, assume a proxy has 5 clients, i.e., $k = 2$, only two of them being dishonest. In that case, there is a majority of honest nodes. Consequently, if the honest nodes send the same local tally, it will be the one chosen by the proxy. However, if the dishonest nodes send different values as their individual tallies then honest nodes will compute different local tallies. The proxy will then decide on the arbitrary local tally sent by the dishonest nodes.

Lemma 4 (Forwarding local tallies). *There exists a public verification scheme so that, a group forwarding inconsistent copies of a local tally, i.e., nodes sending different values to their proxies, is detected with probability* 1 *and the dishonest nodes are exposed.*

Proof. An inconsistent local tally forwarding is detected assuming the following: before deciding on a local tally, a node broadcasts the set of received local tallies to its officemates. An inconsistency is detected if at least one of the following conditions is satisfied: (C1) an honest node received different local tallies from its clients, (C2) an honest node received different local tallies than its officemates. Consider j dishonest nodes concentrated in a group g_i forwarding an incorrect local tally to their proxies. Because of (C1), the clients of an honest node in g_{i+1} must all be dishonest. Since the number of clients of all nodes equals their number of proxies $(2k + 1)$, a coalition of j dishonest nodes can corrupt a maximum of j proxies. Therefore, the $\sqrt{N} - j$ remaining proxies in g_{i+1} must collude with the coalition in g_i to circumvent (C2). To conclude, not to be detected, such an attack requires j dishonest nodes in g_i and $\sqrt{N} - j$ dishonest nodes in g_{i+1}, that is a minimum number of $\sqrt{N}$ dishonest nodes in $g_i \cup g_{i+1}$. Assuming $B < \sqrt{N}$, either a dishonest node in g_i is exposed by a public verification scheme (since it broadcast a local tally that does not correspond to the sum of individual tallies it received) or a dishonest node in g_{i+1} is exposed by a public verification scheme (since it has broadcast a different local tally from the one it received).

4.2 Private verifications

So far we only considered public verifications, i.e., where the content of the ballots is never disclosed. Assume now that the nodes accept, with a non-zero probability, to relax privacy for the sake of verifications and reveal *a subset*

of the ballots they sent and/or received. Then, this partial information can be leveraged to detect the dishonest behaviors described in Lemmas 1 and 2.

Theorem 4. *There is a non-zero probability to detect a dishonest node when misbehaving, even if its impact is less than* $6k + 2$.

Proof. Regarding the voting phase, a dishonest node that sent $k+1+j$ -1-ballots and only $k-j$ +1-ballots ($1 \leq j \leq k$) is unable to provide, for both kinds, the id of $k-j+1$ proxies to which it sent a +1-ballot. Therefore, a simple verification is to ask the suspected node to provide a list of proxies that can testify it sent at least j' ballots of each kind, for a random value j' ranging from 1 to k.

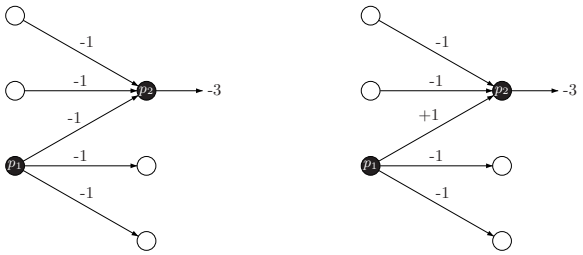


Fig. 2. Dishonest nodes have no gain in covering up each other.

Note that an inspected node can disclose $j' = k$ without revealing its vote. Regarding the ballot corruption attack (Lemma 2), partial information about the ballots received by the inspected node can be leveraged to refine the bound on its individual tally: assume the inspected node received n_b ballots, if we further know that it received at least n_b^+ +1-ballots and n_b^- -1-ballots, then the range is refined from $[-n_b, n_b]$ to $[-n_b + 2n_b^+, n_b - 2n_b^-]$.

In both the aforementioned verifications, a dishonest node has no interest in covering another one up. Consider the examples depicted in Figure 2 where a dishonest node p_1 is the client of a dishonest node p_2. In Figure 2(a), if p_1's vote is verified and p_2 covers p_1 up, i.e., it testifies p_1 sent a +1-ballot, it exposes itself to a private verification on its individual tally (note that a node has to be consistent from one verification to another, thus, if the vote of p_2 is further verified, p_2 must stick to its first version about the ballots it received). The same situation occurs in Figure 2(b): if p_2's individual tally is verified and p_2 covers p_1 up, i.e., it testifies it sent a +1-ballot to p_2, it puts itself at risk, should it be subject to a private verification on its vote. Since dishonest nodes are selfish, they never cover each other up when privately verified. In conclusion, relaxing privacy ensures that every dishonest node has a non-zero probability to be exposed.

5 Implementation

The goal of the evaluation is to assess DPol with respect to the presented theoretical bound, and the impact of relaxing the assumptions made in the analysis. Our experiments show that, in a practical setting, DPol suffers as low as 10% in accuracy, and the average impact of dishonest nodes is around $(4k+1)B$.

5.1 Experimental setup

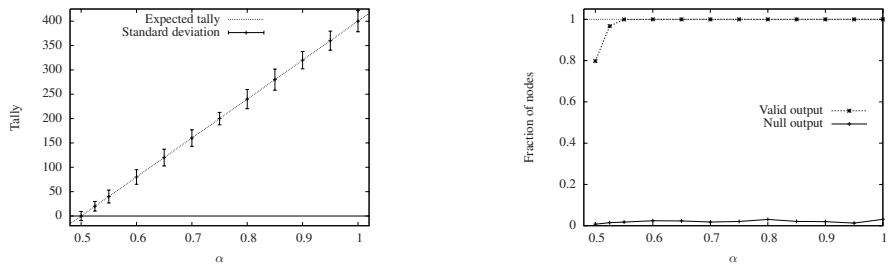
In this section we report on the deployment of DPol on a 400 PlanetLab nodes testbed. This enables to stretch the algorithm in a real world setting, *(i)* in the presence of message losses, crashes and asynchrony inherent in PlanetLab, and *(ii)* when attacking the protocol by introducing dishonest nodes. We evaluate our algorithm with two different privacy parameter values $k = 1$ and $k = 2$.

Overlay The cluster-ring-based overlay is built using a centralized bootstrapping entity keeping track of the whole set of nodes, assigning each node to a random group. Nodes have exactly $2k + 1$ proxies in the next group and the number of clients of a node is $(2k + 1)\,|g_{i-1}|\,/\,|g_i|$ on average.

Communication and Asynchrony Nodes communicate through UDP leading to message losses on the communication channels (with PlanetLab, we observed a loss rate ranging from 5% to 15%). In addition, PlanetLab nodes are unreliable, leading to expected messages to be lost due to senders crashes. Therefore, phase terminations cannot be detected by simply counting the number of received messages. In the local tally forwarding phase, when the number of possible values for a local tally grows beyond a given threshold $\gamma \cdot |\mathcal{P}_c|$, the node makes the decision for this particular local tally in Δt seconds. In our implementation, γ is set to 0.5 and Δt to 5 seconds. The two other phases are simply bounded in time.

5.2 Polling in practice

Accuracy Figure 3 depicts the accuracy of DPol among 400 PlanetLab nodes with $k = 2$. Figure 3(a) considers the value of the tally while Figure 3(b) considers the sign of the tally. Without loss of generality, we consider a proportion α of node voting +1 ranging from 0.5 to 1. In Figure 3(a), we plot the standard deviation on the computed tally for α in that range. For each run, we compute the average of the error when computing the tally (this is the difference between the tally on each node and the real one) over all nodes. Each point represents the average of this value over 20 independent runs. Note that the accuracy increases when α is close to 0.5. This is due to the fact that the closer the tally to 0.5, the fewer message losses impact the outcome: the closer the number of -1-ballots and +1-ballots, the closer to 0 the individual and local tallies.



(a) (b)

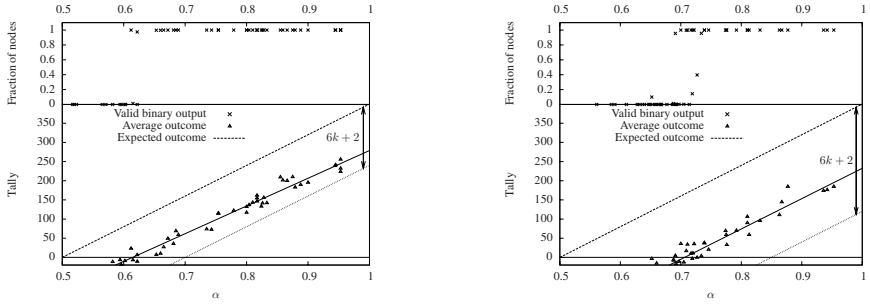
Fig. 3. Accuracy of the algorithm in presence of asynchrony, message losses and failures ($N = 400$ and $k = 2$).

Figure 3(b) displays the fraction of nodes deciding on the correct *sign* of the tally function of α. Effectively, even if the standard deviation is relatively small, some nodes may decide incorrectly on the sign of the outcome. Consider the organizers of a Saturday night party asking their friends in a social network whether partners should be excluded. As depicted in Figure 3(b), for $\alpha = 52.5\%$, some nodes would compute a different answer than the majority. This means that a minority of participants computing a negative result would come with their partners... Figure 3(b) (plain line) also shows the proportion of nodes that are unable to decide on a global tally (because their set of possible values never reach the threshold γ). We observe that this fraction remains very low (less than 4%) and is independent from α.

Attacks We consider the worst case: dishonest nodes do every possible attack that does not compromise their reputation with probability 1, i.e., every dishonest node *(i)* sends $2k + 1$ -1-ballots, and *(ii)* inverts every +1-ballot it receives. Figure 4 displays for $k = 1$ and $k = 2$, the resulting tally (sign on the upper part of the figure and value on the lower part), compared to the real one (dashed line), for $B = 19$ dishonest nodes ($B = \lceil\sqrt{N}\rceil - 1$) in a system of 400 nodes.

We observe that the dishonest coalition affects the outcome of the poll within the theoretical bound derived in the analysis (dotted lines in Figure 4). However, the average impact of the coalition is less than $6k+2$ (considering the worst case where the dishonest proxy receives only +1-ballots and inverts them all). The theoretical bound is never reached as the average impact of a dishonest node depends on the actual number of ballots it can invert, this, in turn, depends on α. Effectively, a dishonest proxy receives $k + \alpha$ +1-ballots out of $2k + 1$ on average. Therefore, its impact is $2k + 2(k + \alpha) = 4k + 2\alpha$ on average. For $k = 2$, fitting our 55 data point cloud with a least-squares regression line (plain line in Figure 4) $a(\alpha - 0.5) + b$ gives $a = 791$ and $b = -163$. This is close to the expected parameter values $a = 2(N - B) = 760$ and $b = -B(4k + 1) = -180$. We use this analysis to make a projection on larger scale systems. For $k = 1$ (Figure 4(a)), every node of the poll outputs a valid binary results when $\alpha > 0.62$, which is

to be compared to $\alpha > 0.55$ observed in Figure 3(b) (without dishonest nodes). On average, we can analytically derive that with $N = 10,000$ and $B = 99$, the proportion α for which the nodes decide correctly is $\alpha > 0.52$.



(a) $k = 1$ (b) $k = 2$

Fig. 4. Accuracy of the poll in the presence of dishonest nodes: with $N = 400$ and $B = 19$, dishonest nodes manage to confuse the majority of the nodes for *(a)* $\alpha < 0.62$ when $k = 1$, and *(b)* $\alpha < 0.73$ when $k = 2$.

6 Related Work

We discuss here related distributed voting protocols with a particular attention to those that are not based on the intractability of mathematical computations. Similarly to most protocols without cryptography, our work ensures privacy via secret sharing techniques. Also, our solution distances itself from related work in the sense that no participant has a special role (following the peer-to-peer paradigm), resulting in an increased scalability and robustness.

A large amount of work on secret sharing schemes (introduced by Rivest *et al.* in [8]) has been published in the late 80's. In [9], Benaloh proposed a scheme for sharing secrets privately based on polynomials. Since this scheme is an homorphism with respect to addition, it can be used for polling. However, a dishonest participant can easily corrupt the shares, thus potentially significantly impacting the final outcome.

Assuming a majority of honest participants, Rabin and Ben-Or extended Benaloh's secret sharing and proposed *verifiable secret sharing scheme* (VSS) [10]. Based on VSS, they proposed a multiparty protocol to compute privately the sum of the participants' inputs with an exponentially small error on the output. Beyond the fact that these techniques assume a fully connected network, synchronous links and broadcast channels, they involve higher mathematics. Moreover, since there is no control over the inputs themselves (contrarily to DPol where the ballot are in $\{-1, 1\}$ and therefore the vote is at max $\pm(2k + 1)$), a dishonest participant may still share an arbitrarily high value and can thus affect the outcome in a potentially unbounded way. The strength of this class of protocols is to ensure strong privacy to participants, including the dishonest

ones but this makes such schemes hardly suitable for polling applications. Note that this also applies to complex secret sharing scheme and private multiparty computation such as AMPC [11].

In [12], Malkhi *et al.* proposed an e-voting protocol based on AMPC and enhanced check vectors. While powerful, this protocol differs from our work since participants have distinct and predefined roles (dealers, talliers, and receivers). This may result in decreased scalability and robustness if specific nodes fail.

7 Conclusion

We considered DPol, a simple decentralized polling protocol and proved that it can be made private and accurate in a social network, where participants are concerned over their reputation, by means of verification procedures, opening the way to a novel and promising way to perform secure distributed computations. In addition we believe that our work can be extended to distributed applications that are not critical (i.e., that are not sensitive to small deviation on their outcome). A natural and interesting perspective is the extension of our polling protocol to n-ary inputs providing doodle-like services [13]. Also, designing an automated tool to help users of a social network to evaluate the reputation of a participant by cross-checking information such as tags and social graphs is definitely a very interesting and challenging problem.

References

1. Richmond, R.: Facebook Tests the Power of Democracy (April, 23$^{\text{rd}}$ 2009)
2. Kremsa Design, Inc.: Facebook Poll http://www.facebook.com/apps/application.php?id=20678178440.
3. Stelter, B.: Facebook's Users Ask Who Owns Information (February, 17$^{\text{th}}$ 2009)
4. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance and Proactive Recovery. TOCS **20**(4) (2002) 398–461
5. Yu, H., Gibbons, P., Kaminsky, M., Xiao, F.: SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In: SP. (2008) 3–17
6. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: Secretive Birds: Privacy in Population Protocols. In: OPODIS. (2007) 329–342
7. Rivest, R., Smith, W.: Three Vvoting Protocols: ThreeBallot, VAV, and twin. In: EVT. (2007) 16
8. Rivest, R., Shamir, A., Tauman, Y.: How to Share a Secret. CACM **22** (1979) 612–613
9. Benaloh, J.: Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. In: CRYPTO. (1986) 251–260
10. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: STOC. (1989) 73–85
11. Malkhi, D., Pavlov, E.: Anonymity without 'Cryptography'. In: FC. (2001) 108–126
12. Malkhi, D., Margo, O., Pavlov, E.: E-Voting without 'Cryptography'. In: FC. (2002) 1–15
13. Doodle: Easy Scheduling: http://www.doodle.com.