# Decentralized Polling With Respectable Participants[☆,☆☆]

Rachid Guerraoui[a], Kévin Huguenin[b,*], Anne-Marie Kermarrec[c], Maxime Monod[a,1], Ýmir Vigfússon[d,e]

[a] *EPFL, LPD, School of Computer and Communication Systems, EPFL, 1015 Lausanne, Switzerland.*
[b] *Université de Rennes I / IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France.*
[c] *INRIA Rennes – Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France.*
[d] *Reykjavik University, School of Computer Science, Menntavegur 1, 101 Reykjavík, Iceland.*
[e] *IBM Research, Haifa University Campus, Mount Carmel, 31905 Haifa, Israel.*

**Abstract**

We consider the polling problem in a social network: participants express support for a given option and expect an outcome reflecting the opinion of the majority. Individuals in a social network care about their reputation: they do not want their vote to be disclosed or any potential misbehavior to be publicly exposed. We exploit this social aspect of users to model dishonest behavior, and show that a simple secret sharing scheme, combined with lightweight verification procedures, enables private and accurate polling without requiring any central authority or cryptography.

We present DPol, a simple and scalable distributed polling protocol in which misbehaving nodes are exposed with positive probability and in which the probability of honest participants having their privacy violated is traded off with the impact of dishonest participants on the accuracy of the polling result. The trade-off is captured by a generic parameter of the protocol, an integer $k$ called the privacy parameter. In a system of $N$ nodes with $B$ dishonest participants, the probability of disclosing a participant's vote is bounded by $(B/N)^{k+1}$, whereas the impact on the score of each polling option is at most $(3k+2)B$ with high probability when dishonest users are a minority (i.e., $B < N/2$), assuming nodes are uniformly spread across groups used by the system. When dishonest users are few (i.e., $B < \sqrt{N}$), the impact bound holds deterministically and our protocol is asymptotically accurate: there is negligible difference between the true result score of the poll and the outcome of our protocol.

To demonstrate the practicality of DPol, we report on its deployment on 400 PlanetLab nodes. The relative error of the polling result is less than 10% when faced with the message loss, crashes and delays inherent in PlanetLab. Our experiments show that the impact on the score of each polling option by dishonest nodes is $(2k+1)B$ on average, consistently lower that the theoretical bound of $(3k+2)B$.

*Keywords:* Distributed polling; Social networks; Overlay networks; Fault tolerance; Security.

## 1. Introduction

The past few years have seen explosive interest in on-line social networks and the number of users of such networks is still growing regularly by the day, with Facebook alone currently boasting more than 750 million active users. Many of these users regularly share images and videos as well as discuss various social and political matters. They do so with close friends, but also with people they hardly know.

A particularly useful task in social networks is *polling*, and often people would want their responses to be private. For instance, Facebook recently conducted a system-wide poll about their terms of service [25]. Or as a hypothetical example, the organizers of a Saturday night party may also want to ask guests whether partners should be invited too. In general, a poll seeks to determine which of $d \geq 2$ *options* is preferred by the greatest number of participants, typically by allowing each participant to submit a single *vote* to indicate her preference. To be meaningful, a polling protocol should tolerate dishonest participants trying to bias the outcome or discover other participants' votes.

One can consider different approaches to conduct a polling in a social network.

*Centralized.* A straightforward solution for polling is to use a central server (e.g., Facebook Poll [19] and Doodle [9]). Each participant sends its vote to a central entity, which subsequently aggregates all votes and computes the outcome. Besides not being scalable, this solution does not ensure privacy because participants might generally not want their vote (and maybe even the subject of the poll and the result) to be known by a central entity, be it trusted or not [33].

*Distributed Aggregation.* Performing the polling through a distributed *aggregation* is a simple, yet naïve, alternative to avoid a central server. Participants aggregate votes in such a way that it is impossible to know the vote of a specific participant. Such an aggregation scheme, however, is vulnerable to attacks. First, participants may significantly bias the result

☆ This article is a revised and extended version of a paper that appeared in the Proceedings of the 13th International Conference on Principles of Distributed Systems (OPODIS 2009) [13].

☆☆ This work has been partially supported by the ERC Starting Grant GOSSPLE number 204742.

* Corresponding author

*Email addresses:* rachid.guerraoui@epfl.ch (Rachid Guerraoui), kevin.huguenin@irisa.fr (Kévin Huguenin), anne-marie.kermarrec@inria.fr (Anne-Marie Kermarrec), maxime.monod@epfl.ch (Maxime Monod), ymir@ru.is (Ýmir Vigfússon)

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

by corrupting intermediate results. Second, even with aggregation, the initial votes needed for bootstrapping are revealed. To prevent the initial bootstrapping votes to be known, each vote could be split according to a homomorphic secret sharing scheme. However, a dishonest participant can still create an invalid initial set of shares, for instance by voting for an arbitrary large value, and thus bias the result.

*Secure Multi-Party Computation.* Assuming a minority of dishonest users, the distributed polling problem may be solved with complete privacy and no bias on the outcome by using heavy machinery from cryptography using protocols for secure multi-party computation (MPC) between $N$ mutually distrustful users [24]. Following two decades of theoretical advances, MPC protocols to address efficiency and scalability concerns have only recently been proposed [6, 5]. The line of work is promising, but even the time and communication complexity of the state-of-the-art protocol (a polynomial in $N$) [5] are too large to be practical.

Our solution to the polling problem does not rely on cryptography for ensuring privacy or accuracy. This is for three reasons. First, we are concerned that the practicality and scalability of the protocol will be impacted by the complexity of cryptographic techniques. Accordingly, we wish to explore the space of protocols where we give up some bounded degree of privacy and accuracy without compromising scalability and performance. Second, there is a small risk that the unproven assumptions underlying modern cryptography techniques (e.g., impracticality of factoring large numbers or inverting higher mathematical functions) may be broken. To avoid a full dependence on such techniques, it is prudent to study alternatives. Third, as advocated in [20] and [26], there is scientific value in determining if traditional problems in distributed computing (e.g., computation in general and polling in particular) can be solved without cryptography and if so, how efficiently. These investigations help us understand the *crux* of the individual problems.

Instead of using cryptography, we exploit the social nature of the participants involved in the polling protocol, specifically the one-to-one correspondence between social network identities and real ones. The key insight is that participants in social networks care about their *reputation*: information related to a user ultimately reflects on the associated *real* person. Therefore, their misbehavior is rather restricted and not fully Byzantine. We leverage this concern and propose an approach which *dissuades* dishonest behaviors instead of masking their impact (e.g., as in BFT [4]) or preventing them (e.g., by using cryptography [1]). In addition to running a polling algorithm based on a simple secret sharing scheme, we execute a distributed verification protocol which tags the profiles of the participants based on collected testimonies. A reputation system like EigenTrust [18] can be used to manage the blames in a robust manner. Social relationships between users further help preventing colluding users from submitting wrongful reports.

To illustrate the idea behind tagging, consider a situation where the testimonies of Alice and Bob demonstrate that Mallory misbehaved. In this situation, Alice's and Bob's profiles are then tagged with "Alice and Bob jointly accused Mallory" and the profile of Mallory is tagged with "Mallory has been accused by Alice and Bob". In a social network, no participant would like to be tagged as dishonest by a protocol that does not wrongly accuse participants, as we will describe below. Our protocol does not wrongfully blame participants, but dishonest participants may [36].

Assuming a system with a large majority of honest participants, the risk for a participant to be caught when 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 allegation would be suspicious and thus not taken into account and the claim would eventually backfire on the accuser. It is important to note that Sybil identities can be detected by analyzing the specific characteristics of social graph (e.g., SybilGuard [39], SybilLimit [38]). Without Sybil identities, the problem of dishonesty boils down to the case where a coalition of *real* users try to affect an honest user's profile by wrongfully blaming her, attempt to spam the system with a large number of blames, or blame one another as a smokescreen. By leveraging the acquaintanceship between users (e.g., the social graph – be it explicit or inferred), several practical systems have been proposed and successfully applied to on-line massively multi-player games [12, 17], spam mitigation [22, 32] and recommendation systems [35, 8]. Most of these techniques require a consensus of an unaffiliated jury to expel a user and renounce blames originating from friends that are considered suspicious.

In devising our protocol, we have considered a system with both honest and dishonest participants. The honest ones follow the protocol assigned to them whereas the dishonest ones might not, in order to promote their opinion beyond what is allowed. Should dishonest nodes deviate from the protocol, we assume that they never do anything that will jeopardize their reputation with certainty (i.e., with probability 1). We believe that our model for dishonest users is more reflective of real human behavior than e.g. Byzantine users as it accounts for social aspects of the participants, and is interesting in its own right. It is an intriguing direction to consider how to solve problems under this model.

*Contributions*

We present DPol, a scalable polling protocol that leverages special properties of social networks. 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 to proxies that belong to another group (an office). In the context of polling, the shares of a vote are referred to as *ballots*. The key idea is that participants can retain privacy by submitting ballots for their chosen candidate as well as ballots for the candidates they *didn't* vote for, making sure to send one more ballot for the true choice than the other ones. 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. Our scheme results in a simple and scalable protocol that is easy to deploy.

*Complexity.* The time complexity of DPol is $\mathcal{O}(\sqrt{Nk})$, the spatial complexity is $\mathcal{O}(\sqrt{Nk})$ and the number of messages sent is $\mathcal{O}(\sqrt{Nk})$, where $k$ is the *privacy parameter* and $N$ the number of participants. We point out that other decentralized protocols have explored the $\mathcal{O}(\sqrt{N})$ trade-off between time and message complexity (e.g., Kelips [14] and [11]).

*Privacy and accuracy.* The trade-off between privacy and accuracy arises naturally in our settings: improving accuracy means verifying the participant's actions, which in turn compromises privacy. We bound the impact on the polling result by dishonest participants and balance this with the level of privacy ensured. More specifically, in a system of $N$ participants with $B$ colluding dishonest participants (assuming a social network with a limited number of Sybil identities), 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 their impact on the outcome of each of the $d$ options is bounded by $(3k+2)B$ with high probability when $B < N/2$ for large $N$, assuming that dishonest nodes have no control over the assignment of nodes to groups (more specifically that nodes are assigned to groups uniformly at random). For $B < \sqrt{N}$, the accuracy guarantee of $(3k+2)B$ always holds for any value of $N$ and for any assignment of nodes to groups, and the relative error on the poll outcome is negligible when $N$ is large. This is due to the ability of our underlying simple secret sharing scheme to expose, with certainty, dishonest participants that affect the outcome by more than $3k+2$ with only public verifications, i.e., without requiring the participants' votes to be revealed. We also show that private verifications expose, with positive probability, dishonest participants who try to affect the outcome below the limit (i.e., even if their impact on each option is less than $3k+2$), but require inspection of the contents of a subset of ballots. Our results also imply that a coalition of a minority of the nodes cannot influence the outcome of a poll if the most popular option has a lead of more than $(6k+4)B$ votes over the runner up.

For illustration, consider a poll with two options, Yes and No (a *binary* poll), in a system of $10,000$ participants. Further, assume there are 99 colluding dishonest participants ($\lceil \sqrt{N} \rceil - 1$), and assume that a fraction $\alpha$ of the participants vote Yes. By setting $k = 1$, for instance, we ensure privacy with probability 99.99% and when $\alpha > 0.54$, every participant computes the right decision (i.e., Yes). While e-voting requires stronger guarantees, this amply fits polling applications requirements.

*Practicality.* DPol is easy to deploy and we report on its deployment on 400 PlanetLab nodes. The result of a binary poll suffers an average 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 $(6k+4)B$. Further, we back up various theoretical results with experimental evaluation throughout the paper.

*Roadmap*

The rest of the paper is organized as follows. Section 2 presents the model and introduces the terms and notation used throughout the paper. We present DPol in Section 3 and we report on the deployment of its binary version on PlanetLab. We analyze the correctness and complexity of the protocol. We derive upper and lower bounds on the impact of dishonest nodes in Section 4 by considering worst case scenarios. Section 5 revisits these bounds in practical scenarios. Probabilistic results are illustrated with simulations and experiments on PlanetLab. Section 6 reviews related work and we conclude in Section 7.

## 2. System Model

We consider a system of $N$ uniquely identified nodes that represent participants of a social network. Each node $p$ votes for a value $v_p \in V$ and the expected output of the polling algorithm is a vector containing the proportions of nodes voting for each value in $V$. The set of possible votes $V$ is $\{1,\ldots,d\}$ where $d$ is the number of options. Each participant in the social network has an assigned profile which may be tagged by DPol.

We assume that nodes are able to communicate by message-passing, specifically that they can receive messages from any other node in the system. To make this assumption work in the real world, the communication takes place using UDP and efficient NAT traversal techniques ("hole punching") such as STUN [29]. Other complementary techniques, for instance the Internet Gateway Device Protocol (through Universal Plug'n'Play), are supported by modern routers and can be used to let devices behind the NAT to dynamically add translation rules thus allowing them to receive incoming messages. We used UDP communication in our implementation and deployment on PlanetLab.

In our model, nodes are either 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. They are, however, 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 are guaranteed to be detected by public verification procedures. As such they are less powerful than Byzantine users. As motivated in the introduction, dishonest nodes do not wrongfully blame honest nodes since it is rather easy for a human reader or an automatic tool aware of social relationship between users, to distinguish between legitimate and wrongful accusations. Consider, for example, a single participant who blames a large number of nodes and the case where a group of related participants all blame an identical set of nodes. Several existing systems manage to filter wrongful blames or at least limit their impact. For instance Digg [8] does not allow related users to rate each other's articles. SumUp [35] allows nodes to vote only over the edges of the social network thus limiting the impact of coalitions of connected users. Ostra [22] bounds the emission rate of a node by the number of trust relationships it has. SocialFilter [32] complements trustworthiness by the

notion of credibility which weights the reports nodes send. Finally, EigenTrust [18] uses an iterative algorithm inspired by PageRank to determine node reputation in a robust way.

We consider participants who collude to form a single coalition $\mathscr{B}$ ($|\mathscr{B}| = B$). When dishonest nodes collaborate to bias the outcome of the poll, they are assumed to share the same opinion. While honest nodes vote for *one* option, dishonest nodes may want to promote a set of options. Still, they act selfishly in the sense that they prefer to protect their own reputation to covering up their suspected accomplices. A single coalition represents the worst case scenario for both discovering a node's vote and for biasing the result of the polling.

DPol relies on a structured overlay, independent of the social graph, which provides scalable dissemination and facilitates verifications. This overlay could be provided by the social network infrastructure or be built in a decentralized fashion. 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 $\mathscr{P}_o$ of *officemates* containing all nodes belonging to the same group ($\mathscr{P}_o = g_i \backslash \{p\}$) and a fixed-size set $\mathscr{P}_p$ of *proxies*, containing nodes in the next group ($\mathscr{P}_p \subseteq g_{i+1 \bmod r}$). Therefore, all groups virtually form a ring with $g_0$ being the successor of $g_{r-1}$. Each group $g_i$ is a clique. We define the *client* of a node $p$ to be a node for which $p$ acts as a proxy. Every node maintains a list of its clients in the previous group ($\mathscr{P}_c \subseteq g_{i-1}$). Each node discards messages that originates in nodes which are not in the set $\mathscr{P}_c \cup \mathscr{P}_o$. Figure 1 depicts the overlay used by DPol. Several prominent overlay construction protocols make use of group-based overlays using gossip, the most commonly known of which is Kelips [14]. These protocols can further be made robust to dishonest nodes and provide mechanisms that we will extend for polling. Fireflies [16], for instance, builds a randomized intrusion-tolerant overlay on which Byzantine nodes have only a limited control, and is resilient to the eclipse attack (i.e., no node has only Byzantine neighbors [36]). Brahms [3] provides unbiased uniform random peer-sampling in the presence of Byzantine nodes. Finally, AVMON [23] builds a pseudo-random overlay based on the hashes of the nodes' IP addresses (on which malicious nodes have very little, if any, control) thus reducing the chances of colluding nodes being connected. We further discuss the overlay construction and how to make it resilient to dishonest nodes in our probabilistic analysis which assumes a uniform random distribution of nodes among groups (Section 5).

## 3. The Polling Protocol

In this section, we present DPol and prove that the protocol is correct; we then analyze its spatial and message complexity and present experimental results of a binary poll on PlanetLab. We complement these experiments with an analysis of DPol in the presence of crashes and message losses.

### 3.1. Polling in a nutshell

DPol is composed of three phases: *(i)* voting, *(ii)* counting and *(iii)* broadcasting. A node's vote goes to *one* option

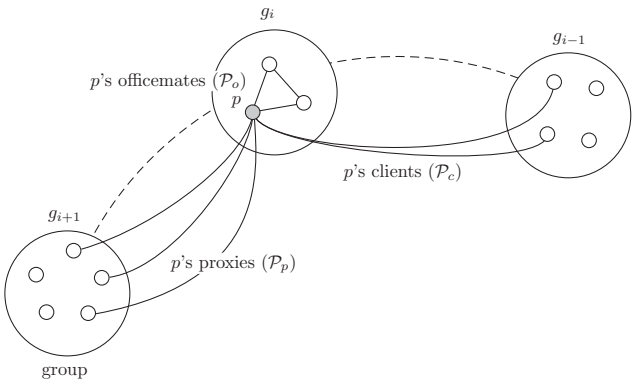



Figure 1: The structure of DPol. Node $p$ in $g_i$ acts as a proxy for its clients in the preceding group $g_{i-1}$, and has proxies of its own in the successive group $g_{i+1}$. It also communicates with its officemates – the nodes in $g_i$.

$v_p \in \{1,\ldots,d\}$. During the voting phase, a node generates a set of ballots in the form of binary vectors of size $d$, $\vec{b} \in \{0,1\}^d$ reflecting its vote (when aggregated) and sends each ballot to one of its proxies. A ballot with only the first two bits set equally promotes the first two options over the $d-2$ other options. 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. Tallies are natural numbers vectors of size $d$, $\vec{t} \in \mathbb{N}^d$ and the component with the maximum value in the final tally vector determines the winner of the poll.

### 3.2. Description

We now give the algorithm details of each phase of the DPol protocol.

*Voting.* The ballot generation method is inspired by the simple secret sharing scheme introduced in [7] and shares similarities with the Vote/Anti-Vote/Vote system [27]. To vote for a given value $v \in \{1,\ldots,d\}$, a node generates $2k+1$ *ballots* $\mathbf{b}_1,\ldots,\mathbf{b}_{2k+1} \in \{0,1\}^d$ representing its vote, where $k$ is an integer called the privacy parameter. To be valid, a ballot must contain at least one 1 and at least one 0.

In the binary case, a valid ballot is $(1,0)$ or $(0,1)$. In this case, the intuition is to create $k+1$ ballots towards a given target (first or second option) and $k$ ballots for the other option, such that when the ballots are summed they result in a vote for the chosen option $v_p$. In general, we assign the $i$-th option with a vector $c_i = (0,\ldots,0,1,0\ldots,0)$ with 1 in the $i$-th position and its complementary vector $\overline{c}_i = (1,\ldots,1,0,1,\ldots,1)$ with a 0 at the $i$-th position. In the case where a node is allowed to vote for a single option, a valid vote therefore equals $c_i + k \cdot (1,\ldots,1)$ where $k$ is the privacy parameter. The ballots emitted by a node must sum to a valid vote. For instance, in the case $d = 3$ and $k = 1$ the set of ballots $\{(1,0,0),(0,1,1),(1,0,0)\}$ is valid and promotes the first option.

**Algorithm 1** DPol at node $p$ in group $g_i$, $i \in \{0,\ldots,r-1\}$

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

**Polling Algorithm**

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

**Voting phase**

**procedure** vote($v$,$\mathscr{P}_p$) **is**
6: $\{\mathbf{b}_1,\ldots,\mathbf{b}_{2k+1}\} =$ share($v$)
7: **for each** $proxy \in \mathscr{P}_p$ **do**
8: send [BALLOT, $\mathbf{b}_i$] ($proxy$)
9: **end for**
**upon event** ⟨ receive | [BALLOT, $\mathbf{b}$] ⟩ **do**
10: $\mathbf{t}'' = \mathbf{t}'' + \mathbf{b}$
**function** share($v$) **returns** a set of ballot $\mathscr{B}$ **is**
11: $\mathscr{B} = \{c_v\}$
12: **for each** $i \in \{1,\ldots,k\}$ **do**
13: $w =$ random number in $\{1,\ldots,d\}$
14: $\mathscr{B} = \mathscr{B} \cup \{c_w\} \cup \{\overline{c}_w\}$
15: **end for**
16: **return** $\mathscr{B}$

**Intermediate Counting phase**

**procedure** local_count($\mathbf{t}''$,$\mathscr{P}_o$) **is**
17: **for each** $officemate \in \mathscr{P}_o$ **do**
18: send [INDIVIDUALTALLY, $\mathbf{t}''$] ($officemate$)
19: **end for**
**upon event** ⟨ receive | [INDIVIDUALTALLY, $\mathbf{t}$] ⟩ **do**
20: $\mathbf{t}' = \mathbf{t}' + \mathbf{t}$

**Local Tally Broadcasting & Forwarding phase**

**procedure** local_tally_broadcast($i$, $\mathbf{t}''$, $\mathscr{P}_p$) **is**
21: **for each** $proxy \in \mathscr{P}_p$ **do**
22: send [LOCALTALLY, $i$, $\mathbf{t}'$] ($proxy$)
23: **end for**
**upon event** ⟨ receive | [LOCALTALLY, $i_{group}$, $\mathbf{t}$] ⟩ **do**
24: $\mathscr{S}[i_{group}] = \mathscr{S}[i_{group}] \cup \{\mathbf{t}\}$
25: **if** ($\left|\mathscr{S}[i_{group}]\right| = |\mathscr{P}_c|$) **then**
26: $\mathscr{T}[i_{group}] =$ choose($\mathscr{S}[i_{group}]$)
27: **if** ($i_{group} \neq (i+1) \mod r$) **then**
28: local_tally_broadcast($i_{group}$,$\mathscr{T}[i_{group}]$)
29: **end if**
30: **end if**
**function** choose($\mathscr{A}$) **returns** local tally **is**
31: **return** the most represented local tally in $\mathscr{A}$

In DPol, a node generates $k$ ballots associated to $k$ random candidates and their complementary ballots and a single ballot for the candidate $v$ it wants to promote (lines 11–16 in Algorithm 1). Such a set of ballots is valid and sums to $c_v + k \cdot (1,\ldots,1)$. 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, $\left|\mathscr{P}_p\right| = 2k+1$. Lines 6–9 in Algorithm 1 detail the voting phase. Figure 2(a) depicts a node sending its $2k+1$ ballots (e.g., $\{(0,1),(1,0),(1,0)\}$) 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 2(b)) and share intermediate results (Figure 2(c)). To this end, a proxy sums the ballots it received into an *individual tally* $\mathbf{t}''$ (line 10 in Algorithm 1). 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 2(b) (lines 17–19 in Algorithm 1). The officemates aggregate the received data, i.e., they sum each others' individual tallies (line 14 in Algorithm 1) and store the result summed with their individual tally into a *local tally* $\mathbf{t}'$ as shown in Figure 2(c) (line 3 in Algorithm 1).

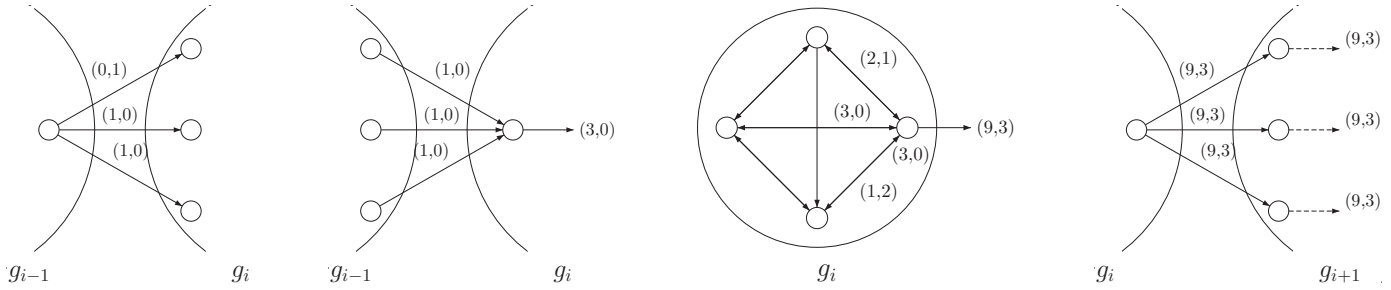
*Local Tally Forwarding.* Once the intermediate counting phase is over, i.e., all the officemates have computed a local tally, each node sends the local tally of its group to its proxies (lines 21–23 in Algorithm 1). Upon reception of a message containing a local tally, a proxy adds it to the set $\mathscr{S}[i]$ of possible values for $g_i$ (line 18 in Algorithm 1). When a proxy has received the expected number $|\mathscr{P}_c|$ of local tallies for a given group $g_i$, it decides on a local tally by choosing the most represented value in $\mathscr{S}[i]$ and stores it in $\mathscr{T}[i]$. When a local tally $\mathscr{T}[i]$ is assigned, it is further forwarded (Figure 2(d)) to the next group using the proxies (lines 25–30 in Algorithm 1). Local tallies are then forwarded in the system along the ring. When a node receives the local tally corresponding to its own group, the tally is no longer forwarded (lines 27–29). The global tally is computed at each node by simply summing the local tallies of all groups: $\hat{\mathbf{t}} = \sum_{i=0}^{r-1} \mathscr{T}[i]$ (line 5 of Algorithm 1).

The nodes in group $g_i$ assume a special role for the members of the preceding group $g_{i-1}$: they collect and count ballots from $g_{i-1}$ and initiate the dissemination of the resulting local tally over the ring. Each group has a special role for the preceding one, and all nodes execute the exact same protocol.

### 3.3. Analysis

In this section, we analyze the correctness and complexity of DPol assuming an ideal setting (i.e., reliable channels and non-faulty nodes) and only honest nodes. We later revisit these assumptions by measuring the impact of message loss and crashes both analytically and experimentally in Section 3.4. We will consider the impact of dishonest nodes on privacy and accuracy in the worst case in Section 4 and on average in Section 5.

**Theorem 1** (Correctness)**.** *Consider a system of size N where each node p votes for the $v_p$-th option. The polling algorithm terminates and each node eventually outputs $N \cdot k(1,\ldots,1) + \sum_p c_{v_p}$.*




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

Figure 2: Key phases of DPol with $d = 2$. (a) A node in $g_{i-1}$ generates 3 ($k = 1$) ballots $\{(1,0),(1,0),(0,1)\}$ and sends them to its proxies in $g_i$. (b) A node in $g_i$ collects its received ballots $\{(1,0),(1,0),(1,0)\}$ and sums them to $(3,0)$ (individual tally) and shares the tally with its officemates in $g_i$ as depicted in (c). (c) A node receives all expected individual tallies $\{(3,0),(2,1),(1,2)\}$, then computes and sends the local tally $((9,3))$ to its proxies in the next group $g_{i+1}$ as depicted in (d). (d) The proxies in $g_{i+1}$ forward the local tally to their proxies in $g_{i+2}$.

*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 $\mathbf{b}_{1,p},\ldots,\mathbf{b}_{2k+1,p}$ to distinct proxies in $g_i$. Similarly, each proxy $p'$ in $g_i$ receives a set of ballots $\mathscr{B}_{p'}$ from its clients. Since all nodes are honest by assumption, the set of ballots sent by the nodes in $g_{i-1}$ equals the set of ballots received in $g_i$. Therefore, each member of $g_i$ computes the local tally to be:

$$\begin{aligned}
\mathbf{t}' &= \sum_{p'\in g_i}\sum_{\mathbf{b}\in\mathscr{B}_{p'}}\mathbf{b} \\
&= \sum_{p\in g_{i-1}}\sum_{j=1}^{2k+1}\mathbf{b}_{j,p} \\
&= \sum_{p\in g_{i-1}}\left[k\cdot(1,\ldots,1)+c_{v_p}\right] \\
&= |g_{i-1}|\cdot k\cdot(1,\ldots,1)+\sum_{p\in g_{i-1}}c_{v_p}
\end{aligned}$$

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, and 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. Because the algorithm has a finite number of phases, it is guaranteed to eventually terminate. □

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

*Proof.* A node maintains the 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 (a node has $2k+1$ clients on average) for each of the $r$ local tallies to perform global counting, that is $S = \mathscr{O}(k)+\mathscr{O}(|g_i|)+\mathscr{O}(|g_{i-1}|)+\mathscr{O}(r\cdot k) = \mathscr{O}(r\cdot k+|g_i|)$. □

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

*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 $M = \mathscr{O}(k)+\mathscr{O}(|g_i|)+\mathscr{O}(r\cdot k) = \mathscr{O}(r\cdot k+|g_i|)$. □

Note that the parameters are not independent: the sizes of the groups are related and bound to the number of groups by the relation $\sum_{i=0}^{r-1}|g_i| = N$. The two quantities $M$ and $S$ are minimized when $r = \sqrt{N/k}$ and $|g_i| = \sqrt{Nk}$, and thus $M = S = \mathscr{O}(\sqrt{Nk})$.

**Proposition 3** (Time complexity)**.** *Under the assumption of a synchronous system where time evolves in rounds,* DPol *operates in $\mathscr{O}(\max|g_i|+rk)$ rounds.*

*Proof.* The voting phase operates in $2k+1$ rounds. The counting phase requires each node to send its individual tally to its $|g_i|-1$ officemates. All the nodes in the system send their individual tally in parallel, therefore the time complexity of this phase is $\mathscr{O}(|g_i|)$. The local tally broadcast phase operates in $\mathscr{O}(kr)$ rounds as it requires $2k+1$ rounds for a local tally to be forwarded from one group to the next one and the ring is composed of $r$ groups. □

Using the values of the parameters $r$ and $\{g_i\}_{i=1}^{r}$ specified above, the time complexity of DPol is $\mathscr{O}(\sqrt{Nk})$.

### *3.4. Evaluation*

We report on the deployment of DPol, in the case of binary polling, on a PlanetLab testbed of 400 nodes and analyze the practical performance of DPol. The message loss rates, crashes and asynchrony inherent to PlanetLab allow us to experiment with the algorithm in tough real-world settings. 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 which keeps 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 by UDP which may suffer message loss on the communication channels. For instance, we observed loss rates on PlanetLab ranging from 5% to 15%. In addition, PlanetLab nodes are unreliable, causing expected messages to be lost due to sender 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 $\gamma \cdot |\mathscr{P}_c|$, the node gets $\Delta t$ seconds to make the decision for this particular local tally. The two other phases are simply bounded in time. In our implementation, $\gamma$ is set to 0.5 and $\Delta t$ to 5 seconds.

*Experimental results.* Figure 3 shows the accuracy of DPol among 400 PlanetLab nodes with $k = 2$. Figure 3(a) considers the value of the outcome while Figure 3(b) considers the sign of the outcome. By outcome we mean the difference between the number of votes for the first option (i.e., corresponding to the first component of the tally) and the number of votes for the second option. More specifically, the outcome equals $(1,-1)^T \cdot \mathbf{t}''$. Without loss of generality, we vary the proportion $\alpha$ of node voting for the first option between 0.5 and 1. In Figure 3(a), we plot the standard deviation on the computed outcome for $\alpha$ in that range. For each run, we compute the average of the error when computing the outcome (this is the difference between the outcome 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 $\alpha$ is close to 0.5. This is because the closer the tally is to 0.5, the lower the impact of message losses on the outcome. Effectively, the two components of the individual and local tallies are close and therefore their impact on the outcome is close to zero.

Figure 3(b) displays the fraction of nodes deciding on the winning option (among the nodes that were able to decide on a global tally) as a function of $\alpha$. Effectively, even if the standard deviation is relatively small, some nodes may decide incorrectly on the winning option. Consider the organizers of a Saturday night party asking their friends in a social network whether partners should be excluded or not. 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 who compute negative outcome would come with their partners. Figure 3(b) (solid 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 $\gamma$). We observe that this fraction remains very low (less than 4%) and is independent of $\alpha$.

*Analysis.* Crashes and message loss do arise and affect the correctness and termination of the protocol. Crashes can impact the system in two different ways. First, nodes may crash independently before sending *unique* information. The information refers to data they received that did not yet get replicated, typically initial shares of votes (i.e., ballots). Losing such data affects the global tally. Second, several nodes may be crashed at given time. This may result in other nodes being unable to decide on a local tally and thus on the global tally from a lack of corroborating pieces of information. In the following, we analyze the impact of the first type of crashes on the outcome. Finally, by relaxing the decision condition from $\left|\mathscr{S}[i_{group}]\right| = |\mathscr{P}_c|$ to $\left|\mathscr{S}[i_{group}]\right| \geq \gamma \cdot |\mathscr{P}_c|$ (as described above), we compute the probability of a node failing to decide on the global tally.

**Proposition 4** (Impact of crashes on accuracy)**.** *An individual crash can affect the score of an option up to* $3k+2$*.*

*Proof.* Consider a node that crashes before broadcasting its individual tally to its officemates. This individual tally is lost, and represents the sum of the $|\mathscr{P}_c| = 2k+1$ ballots sent by its clients. The impact of such a crash on each option score is bounded by $2k+1$ since ballots are binary vectors. Moreover, if a node crashes while sending its ballots, it affects each option score by up to $k+1$. The maximum impact of an individual crash is thus $2k+1+k+1 = 3k+2$. □

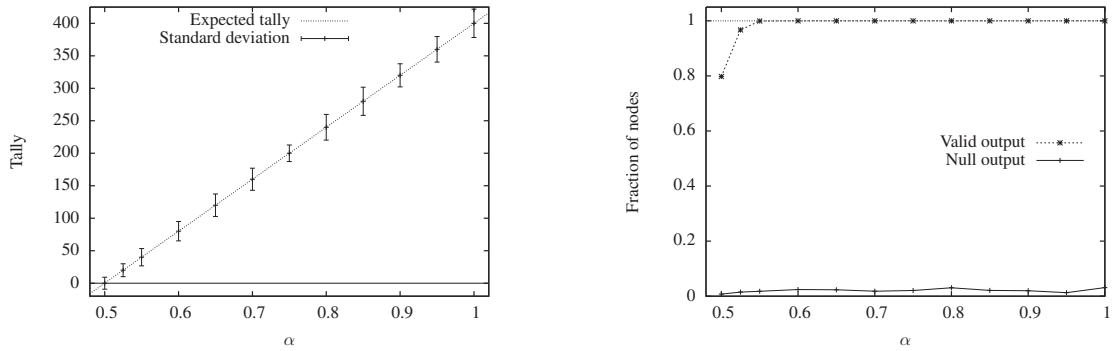
We assume that nodes crash with probability $c$ and never recover from a crash. There is a probability $\overline{d}_i$ for a node $p \in g_i$ not to decide on the local tally of group $r$. This can happen if more than $(1-\gamma)\,|\mathscr{P}_c| = (1-\gamma)(2k+1)$ clients fail to forward the local tally because they either crashed or have themselves not decided on the local tally. We define $\overline{e}_i$ as the probability for a node in $g_i$ to fail to forward a local tally. We have $\overline{e}_i = c + (1-c)\overline{d}_i$ where

$$\overline{d}_i = \sum_{j=0}^{\gamma(2k+1)-1} \binom{2k+1}{j} (1-\overline{e}_{i-1})^j \overline{e}_{i-1}^{2k+1-j} \text{ and } \overline{d}_0 = 0$$

A node does not decide on the global tally if it has not has decided on at least one local tally, that is $d_r$.

*Discussion.* DPol is a decentralized peer-to-peer protocol deployed on the Internet, where nodes may leave and join the system dynamically. This behavior, referred to as churn, is both common and widespread, and may further disrupt the function of the protocol beyond message loss and node crashes. Fortunately, because our protocol is lightweight and executes quickly, the impact of churn is at the same manageable level as we observed in our PlanetLab experiments (i.e., only a few nodes among the 400 nodes participating in the poll crashed or left during the course of the protocol) as we detail below.

As we explained at the beginning of this Section, the phases of DPol are bounded in time for fault-tolerance. During our experiments with 400 nodes, we set the time-out $\delta$ to 5 seconds, thus giving an execution time of less than 20 seconds. Using Proposition 3, which states that the time complexity of DPol grows as $\sqrt{N}$, we can make a projection for a poll involving 100,000 participants (recall that DPol targets polls within




(a) Outcome of the poll.

(b) Fraction of node deciding correctly on the winner.

Figure 3: Accuracy of the algorithm in presence of asynchrony, message loss and failures ($N = 400$ and $k = 2$). [400 nodes PlanetLab tested]

groups rather than in the entire social network): the algorithm is expected to complete in approximately five minutes.

To put this duration in perspective, several trace-driven analysis of churn in various peer-to-peer systems and applications, including file-sharing (BitTorrent, eDonkey [34]), Voice over IP (Skype [37]) and on-line social networks (Facebook, Hi-Five [30]), report an average session time of at least a few minutes. They further show that this duration is significantly increased when considering peers who *already* spent some time in the system, and initial duration of 5 minutes are typical for social networks. For instance, it is shown in [34] that more than 95% of the peers who already spent an hour in the system stay at least one more hour in the system. Therefore, by assuming (or even requiring) that peers have spent a reasonable time online to participate to the poll, the impact of churn is brought down to a level in which we ensure stability for a correct execution of the protocol. This corresponds to the outcome of our experiments, in which only a handful of nodes were unable to determine the outcome of the poll due to crashes.

## 4. Impact of dishonest nodes

In this section, we analyze the maximum impact of dishonest nodes on DPol. For the sake of the analysis, we assume an overlay with $\sqrt{N}$ groups of size $\sqrt{N}$ ($\sqrt{N} \in \mathbb{N}$), such as Kelips [14], and that each node has exactly the same number $(2k+1)$ of clients and proxies. We consider a polling scheme based on secret sharing to preserve privacy and we assume auxiliary verification schemes to detect attacks and identify dishonest nodes. We distinguish between two types of verifications: *(i)* public verifications that leverage only information which 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 affect the user 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 asked for the messages they exchanged. If the testimonies of $p_1$ and $p_2$ demonstrate that $p_0$ misbehaved, their profiles are 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 $p_2$". These tags can then be used to determine abnormal behaviors.

### 4.1. Preserving privacy in the presence of dishonest nodes

We derive a theoretical upper bound on the impact of a coalition of dishonest nodes on the nodes' privacy, that is the maximum number of votes the dishonest coalition can disclose.

**Theorem 2.** *A coalition of $B$ malicious nodes can disclose the votes of at most $\lfloor B \cdot \frac{2k+1}{k+1} \rfloor \leq 2B$ honest users.*

*Proof.* This theorem follows from the secret sharing scheme used in DPol, which is to divide a node's vote into $2k+1$ ballots. Among them, $k+1$ ballots have the bit corresponding to the node's vote set and only $k$ ballots have the bit corresponding to each competing candidate set. We first prove that a node's vote is disclosed by dishonest nodes if and only if its $k+1$ proxies that received the ballots corresponding to its vote (i.e., the ballots in which the $v_p$-th bit is set) belong to the coalition. It is clear that if $k+1$ ballots received by the dishonest have a bit set at the $v_p$-position, the vote is recovered with certainty. We now prove the contrapositive statement by contradiction. Let $\vec{b}$ be the sum of the ballots sent by $p$ to dishonest proxies. We suppose that all the components of $\vec{b}$ are strictly lower than $k+1$ and assume that the dishonest coalition recovers the client's vote. Not all of the $2k+1$ proxies can be dishonest, otherwise, $\vec{b}$ would simply have been the vote of $p$, that is $c_{v_p} + k \cdot (1, \ldots, 1)$ (whose $v_p$-th component is equal to $k+1$). The best case (from the standpoint of the coalition) is when $2k$ proxies are dishonest. In that case, the components of $\vec{b}$ are either equal to $k-1$ or $k$. The vote of $p$ is recovered with certainty if and only if a single component of $\vec{b}$ is equal to $k$. This implies that the missing ballot contains only ones, which is in contradiction with the definition of a valid ballot.

Since all nodes, including dishonest ones, have exactly $2k+1$ clients, the dishonest coalition collects a total of $B \cdot (2k+1)$ ballots which in turn may recover at most $\lfloor B \cdot \frac{2k+1}{k+1} \rfloor$ votes. □

### 4.2. Confining the impact of dishonest nodes

First, 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.

Next, we assume that honest nodes are willing to sacrifice privacy for accuracy by revealing some of their ballots (i.e., private verification) and we explore how this information can be used to catch dishonest nodes that cheat without being detected by public verifications.

#### 4.2.1. Impact of a dishonest coalition under public verifications

To anthropomorphize the discussion, suppose that votes are being cast for $d$ distinct *candidates* representing the different options of the poll. Recall that the global tally is the $d$-ary vector $\sum_p c_{v_p} + N \cdot k \cdot (1,\ldots,1)$, whose components correspond to the tallies for each candidate in the poll.

**Theorem 3.** *For $B < \sqrt{N}$, every member of a dishonest coalition may affect each candidate score up to $3k+2$ without being detected by public verifications.*

*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 tarnished with certainty. Effectively, showing that the attacks with an impact greater than $3k+2$ are detected by the honest nodes with certainty proves the theorem. A dishonest node may bias the protocol at all three phases. Lemmas 4-7 encompass all possible attacks, propose a detection scheme relying on honest nodes, and bound the impact of those that cannot be detected with certainty. 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 4 and 5) for each dishonest node gives a maximum impact of $k+1+(2k+1) = 3k+2$ on each component of the global tally.

Note that the proof relies on the assumption $B < \sqrt{N}$ , where $N$ is the size of the system and the size of a group is $\sqrt{N}$, to ensure that the dishonest coalition can neither "control" (there is at least one honest node to report a misbehavior inside each group) nor "fool" an entire group without being detected (there are not enough dishonest nodes to both perform and cover dishonest actions). This security property holds deterministically under the assumption $B < \sqrt{N}$ irrespective of how dishonest nodes are distributed among the groups. Indeed, in the worst case scenario where all dishonest nodes are concentrated in one or two consecutive groups, the fact that the number of dishonest nodes is strictly smaller than the size of a group guarantees that there is at least one honest node in each group and that there is a majority of honest node in every pair of successive groups.

The weakest assumption needed is that two consecutive groups contain less than $\sqrt{N}$ dishonest nodes, formally that $|g_i \cap \mathscr{B}| + |g_{i+1} \cap \mathscr{B}| < \sqrt{N}$ for all $i$. In Section 5, we provide probabilistic analysis of this bound in a setting where dishonest nodes are distributed randomly among the groups and prove that the system is not compromised (dishonest nodes do not form a majority in any pair of consecutive groups) with high probability when $N$ tends to infinity for $B < N/2$. □

**Corollary 1.** *When the margin of the leading candidate over the next candidate is more than $(6k+4)B$, a coalition of $B < \sqrt{N}$ dishonest nodes cannot influence the outcome of the poll.*

*Proof.* By Theorem 3, colluding dishonest users may decrease the score of the leading candidate by at most $(3k+2)B$ and boost the score of some other candidates by at most $(3k+2)B$. This decreases the lead of the top candidate over the runner up by at most $(6k+4)B$, which is not sufficient to change the outcome of the poll. □

Note that one can infer a relation between the margin by which the leading candidate wins and the maximum number of dishonest nodes the system can tolerate (i.e., to output the correct winner). Consider the binary case for instance where a proportion $\alpha > 0.5$ of nodes promote the first candidate. Then the margin is $N(2\alpha-1)$ and the maximum number of dishonest nodes the system can tolerate is $N(2\alpha-1)/(6k+4)$.

**Corollary 2.** *If the proportions of nodes voting for each candidates are nonzero, DPol is asymptotically accurate for $B = o(N)$ dishonest nodes.*

*Proof.* Let $(\alpha_1,\ldots,\alpha_d)$ be the proportions of nodes voting for the respective candidates. It can be inferred from Theorem 3 that the relative error on the score of the $i$-th candidate is bounded by $(3k+2)B/(\alpha_i N)$. Using the fact that $B$ is a sublinear function of $N$ proves that the relative error on the vector of scores computed by DPol tends to 0 when $N$ tends to infinity, which concludes the proof. □

**Lemma 4** (Voting)**.** *When voting, a dishonest node can affect each candidate score by at most $k+1$.*

*Proof.* Due to the overlay structure, a node can only send ballots to the proxies it is assigned (otherwise the ballots are discarded), which is a maximum of $2k+1$ ballots. Therefore a dishonest node may affect each component of the global tally by either *(i)* sending less ballots than it is supposed to or *(ii)* by sending less than $k$ or more than $k+1$ positive ballots for a given candidate (i.e., the bit corresponding to that candidate is set in the considered ballot). In the worst case, the dishonest node sends either $2k+1$ or 0 positive ballots for a candidate. Since the node should send either $k$ or $k+1$ positive ballots for that candidate, the maximum impact is $|(2k+1)-k| = |0-(k+1)| = k+1$. Note that if the node sends $2k+1$ positive ballots for the candidate it is voting for, or 0 positive ballots for a candidate that it is not voting for, the impact is $k$. □

**Lemma 5** (Computing individual tallies)**.** *Assuming $B < \sqrt{N}$, there exists a public verification scheme such that if a dishonest node modifies the individual tally for a candidate by more than $2k+1$, then the attack is detected with certainty and the node is exposed.*

*Proof.* The overlay structure we consider ensures that any node has exactly $2k+1$ clients and thus receives $2k+1$ ballots during the voting phase. A dishonest node can modify the candidate score by pretending that it received some other number of positive ballots for that candidate, thus affecting the candidate score in its individual tally. If the dishonest node tries to forge too many ballots, specifically by reporting a candidate score outside the range $[0, 2k+1]$, then the attack is identified by its honest officemates (the assumption $B < \sqrt{N}$ ensures that at least one such node exists in each voting office). Therefore, in order to not to be publicly detected with certainty, a node that corrupts or forges ballots must output an individual tally in that range. Consequently, the worst case occurs when a dishonest node receives $2k+1$ positive ballots for a candidate and discards them all while summing them, leading to a maximum impact of $2k+1$ on that candidate's score. □

We stress that the verification schemes described in the proof succeed to detect misbehavior even in case of collusion. Consider the following situation: a client sends an erroneous ballot (e.g., with value 2 for the first candidate) to a colluding proxy who will aggregate it with its individual tally without reporting the misbehavior. If the dishonest proxy has received only positive ballots for the first candidate (or already turned all the negative ballots into positive ones), the first component in its individual tally will become $2k+2$ which is larger than $2k+1$. This range violation will be detected with certainty. If the dishonest proxy received at least one ballot negative for the first candidate, then covering up for its co-conspirator comes down to turning this negative ballot into a positive one, and the impact will therefore be bounded by $2k+1$. Finally, the assumption that at most $\sqrt{N}-1$ nodes are dishonest ensures that there is at least one honest node in each group to report individual tallies outside the range $[0, 2k+1]$.

**Lemma 6** (Broadcasting individual tallies)**.** *There exists a public verification scheme so that a dishonest node that broadcasts inconsistent copies of its individual tally to honest nodes, i.e., sending different values to its honest officemates, is detected with certainty and the node is exposed.*

*Proof.* Before deciding on a local tally, every node broadcasts the set of individual tallies it received to its officemates. This way, an honest officemate will trivially detect 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, suppose that $k = 2$ and that some proxy has $2k+1 = 5$ clients of which only two are dishonest. In this 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 because it is the most represented.

**Lemma 7** (Forwarding local tallies)**.** *There exists a public verification scheme to detect with certainty if a group forwards inconsistent copies of a local tally, i.e., nodes sending different values to their proxies, and to expose the dishonest nodes.*

*Proof.* Inconsistency in 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 $(2k+1)$, $j$ colluding 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). This is illustrated in Figure 4. To conclude, in order 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). The proof holds for every hop of the forwarding, including the initial hop which is conducted by the nodes who actually computed the local tally being forwarded. □

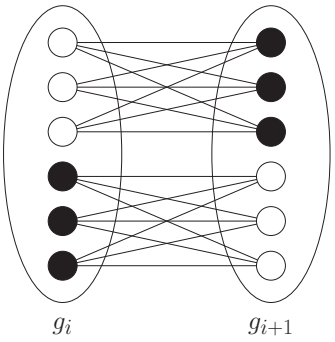



Figure 4: Corrupted local tally that remains undetected.

*4.2.2. Leveraging private verifications*

So far we have only considered public verifications in which the contents of the ballots are never disclosed. Now assume that the nodes accept, with nonzero probability, to relax privacy for the sake of verifications and reveal *a subset* of the ballots they sent or received or both. This partial information can then be leveraged to detect the dishonest behaviors described in Lemmas 4 and 5.

Consider as a first step, for the sake of simplicity, the case of binary polling. During the voting phase, a dishonest node

that sent $k+1+j$ ballots $(1,0)$ and only $k-j$ ballots $(0,1)$ ($1 \leq j \leq k$) is unable to provide the identifier of $k-j+1$ proxies to which it sent a $(0,1)$ ballot. Therefore, a simple verification is to ask the suspected node to provide a list of proxies who can testify that the node sent at least $j'$ ballots of each kind, for a random value $j'$ ranging from 1 to $k$. Note that an inspected node can disclose up to $j' = k$ ballots without revealing its vote. For $d > 2$, the components of an inspected node's vote may be verified independently. Effectively, to check that the vote of a node for the $i$-th candidate is in $[j', 2k+1-j']$ ($j'$ ranging from 1 to $k$), the verifier asks the inspected node to provide the address of $j'$ proxies to which it sent a ballot with a 0 at the $i$-th position and $k$ proxies to which it sent a ballot with a bit set in the $i$-th position.

During a ballot corruption attack (Lemma 5), partial information about the ballots received by the inspected node can be leveraged to refine the bound on its individual tally: suppose the inspected node received $n_b$ ballots (i.e., under the perfect client-proxy matching assumption $n_b = 2k+1$), if we further know that it received at least $n_b^+$ ballots $(1,0)$ and $n_b^-$ ballots $(0,1)$, then the bound on the score of the first candidate can be refined from $[0, n_b]$ to $[n_b^+, n_b - n_b^-]$. Similarly, the score of the second candidate in the individual tally must be in the range $[n_b^-, n_b - n_b^+]$. This verification scheme extends naturally to the cases where $d > 2$.

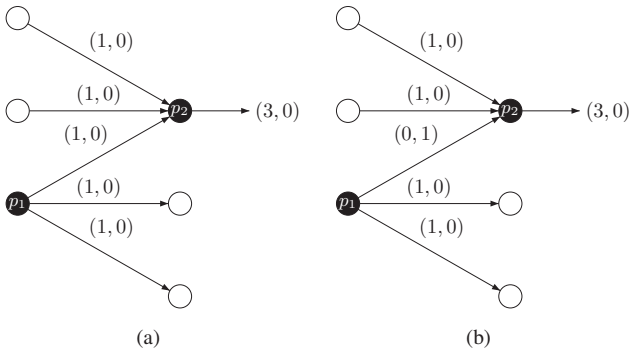



Figure 5: Dishonest nodes ($p_1$ and $p_2$) do not benefit from covering up for each other (illustrated in the binary case).

In both of the aforementioned verification schemes, dishonest nodes have no interest in covering up for one another. Consider the examples depicted in Figure 5 where a dishonest node $p_1$ is the client of a dishonest node $p_2$. In Figure 5(a), if $p_1$'s vote is verified and $p_2$ covers $p_1$ up, i.e., it testifies that $p_1$ sent a ballot $(0,1)$, then it exposes itself to a private verification on its individual tally. Note that a node's statement has to be consistent across different verifications, 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 5(b): if $p_2$'s individual tally is verified and $p_2$ covers up for $p_1$, i.e., it testifies it sent a ballot $(0,1)$ to $p_2$, it puts itself at risk should it be subject to a private verification on its vote. Since we assume that dishonest nodes are selfish, they never cover each other up when privately verified. In conclusion, relaxing privacy ensures that every dishonest node has nonzero probability to be exposed. A lower bound on this probability is given in Section 5.5.

## 5. Polling in practice

So far we evaluated the impact of a coalition of dishonest nodes in a worst case scenario. In this section, we revisit the results and assumptions of the previous section for the average case. More specifically, we assume a random uniform distribution of nodes across the $r$ groups and that nodes in the next groups are distributed uniformly at random as proxies in the preceding groups. We justify this assumption by sketching an overlay construction protocol, inspired by various techniques found in the literature, which guarantees a uniform pseudo-random distribution of nodes in groups on which nodes will have no or very little control (§5.1). We then study the average impact of dishonest nodes on privacy (§5.2) and accuracy (§5.3). Then we refine the security condition that prevents dishonest nodes from biasing local tallies in an unbounded way during forwarding (i.e., controlling two consecutive groups as explained in Lemma 7) (§5.4). Finally, we give a lower bound on the probability of detecting a dishonest node who cheats within the bounds (i.e., $3k+2$) by means of private verifications as function of the willingness of honest nodes to compromise their privacy (i.e., disclose their vote) (§5.5). The results presented in this section are consistent with simulations and experiments on a 400-node PlanetLab testbed.

### 5.1. Overlay construction

The overlay driving DPol organizes the nodes in $\sqrt{N}$ groups arranged on a ring. Each node is connected to all the nodes in its group (i.e., its officemates), as well as exactly $2k+1$ *proxy* nodes in the next group on the ring, and to $2k+1$ *client* nodes in the previous group on the ring. A node is assigned to the group $\text{hash}(\text{IP}) \mod \sqrt{N}$. Note that this assignment may be verified locally by any node. Nodes can discover the nodes in their own group (and thus connect to their officemates) and in the next one using a (robust) peer sampling service. By making each node sort the list of the nodes in its own group and in the next one by increasing hashes and making the $j$-th node in group $i$ choose node number $j+1, \ldots, j+k$ in group $i+1$ as proxies, we obtain the overlay required by DPol, with the desired uniformity property, in a scalable and decentralized fashion. The overlay construction protocol, while being deterministic in its initial design, can be randomized by concatenating a random value shared by all nodes to the nodes' IP addresses when computing the hashes.

### 5.2. Privacy

We now assess the privacy guarantees provided by DPol when the dishonest nodes are placed (i.e., the groups they belong to and their set of clients and proxies) uniformly at random.

**Theorem 8** (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.* We proved in Theorem 2 that the vote of a node is recovered with certainty by the dishonest nodes if and only if the $k+1$ proxies who received the $k+1$ ballots containing a vote for the chosen candidate collude. This event occurs with probability $\binom{B}{k+1}/\binom{N}{k+1}$ when nodes are randomly distributed in the overlay. For all $k, B$ and $N$, this probability is bounded by $(B/N)^{k+1}$. □

In order to link the average level of privacy guaranteed by the protocol to the parameter $k$, the system administrator needs to estimate the proportion of dishonest nodes. However, it is important to note that DPol is *oblivious* of this proportion: be the proportion of dishonest nodes higher than the estimation, the level of privacy offered would be lower than expected but DPol would still function properly.

### 5.3. Accuracy

We now report on the evaluation of DPol in the binary case over the PlanetLab testbed. The goal of the evaluation is to compare DPol against the presented theoretical bounds. Our experiments focus on binary polling. These experiments show that, in a practical setting, DPol suffers an average impact of dishonest nodes of around $(4k+2)B$ on the outcome of the poll (i.e., the score of the first candidate minus the score of the second candidate). With a proportion $\alpha$ of nodes voting for the first candidate, the outcome is $\alpha N-(1-\alpha)N = N(2\alpha-1)$.

We consider the worst case: dishonest nodes perform every possible attack that does not compromise their reputation with probability 1 to promote the second candidate, i.e., each dishonest node *(i)* sends $2k+1$ ballots $(0,1)$, and *(ii)* inverts every ballot $(1,0)$ it receives into a ballot $(0,1)$. Figure 6 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 $N=400$ 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 6). Since a dishonest node can impact the score of each candidate by up to $3k+2$, its maximum impact on the outcome is $6k+4$. However, the average impact of the coalition is less than $6k+4$ (considering the worst case where the dishonest proxy receives only ballots $(1,0)$ 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 the proportion $\alpha$ of nodes voting for the first candidate.

Effectively, a voting node sends $k+1$ ballots $(1,0)$ out of $2k+1$ if it votes for the first candidate and $k$ otherwise. Therefore, the number of ballots $(1,0)$ received by a proxy is $(2k+1)\left[\alpha\frac{k+1}{2k+1}+(1-\alpha)\frac{k}{2k+1}\right]=k+\alpha$ on average. The impact of a dishonest user who turns ballots $(1,0)$ into ballots $(0,1)$ is $2(k+\alpha)$ on average. In addition, dishonest nodes impact the outcome by another $2k$ by sending $2k+1$ ballots $(0,1)$ during the voting phase. Their total impact is therefore around $4k+1$.

Considering a system with $B$ such dishonest users, the biased outcome can be expressed as $N(2\alpha-1)-B(4k+2\alpha)=2(N-B)(\alpha-\frac{1}{2})-B(4k+1)$. For $k=2$, fitting our 55-data point cloud with a least-squares regression line (plain line in Figure 6) $a(\alpha-\frac{1}{2})+b$ 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 6(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 derive analytically that with $N=10{,}000$ and $B=99$, the proportion $\alpha$ for which all nodes decide correctly is $\alpha>0.52$.

### 5.4. Security

Assuming that dishonest nodes are able to choose their proxies and clients, we have shown in Lemma 7 that they can both perform and cover dishonest actions as soon as the number of dishonest nodes in two consecutive groups is greater than $\sqrt{N}$. In other words, the protocol is secure against this attack when the condition $\forall i, |g_i\cap\mathscr{B}|+|g_{i+1}\cap\mathscr{B}|<\sqrt{N}$ holds. If $B<\sqrt{N}$, this condition holds with certainty, regardless of the distribution of dishonest nodes among groups. Otherwise, there is a nonzero probability that the condition is violated.

**Theorem 9** (Tolerance to dishonest nodes)**.** *The probability that B dishonest nodes compromise the system (i.e., control two consecutive groups as defined in Lemma 7) is* 1 *when* $B\geq\frac{N}{2}$*, and converges to* 0 *exponentially fast in* $\sqrt{N}$ *when* $B<\frac{N}{2}$*.*

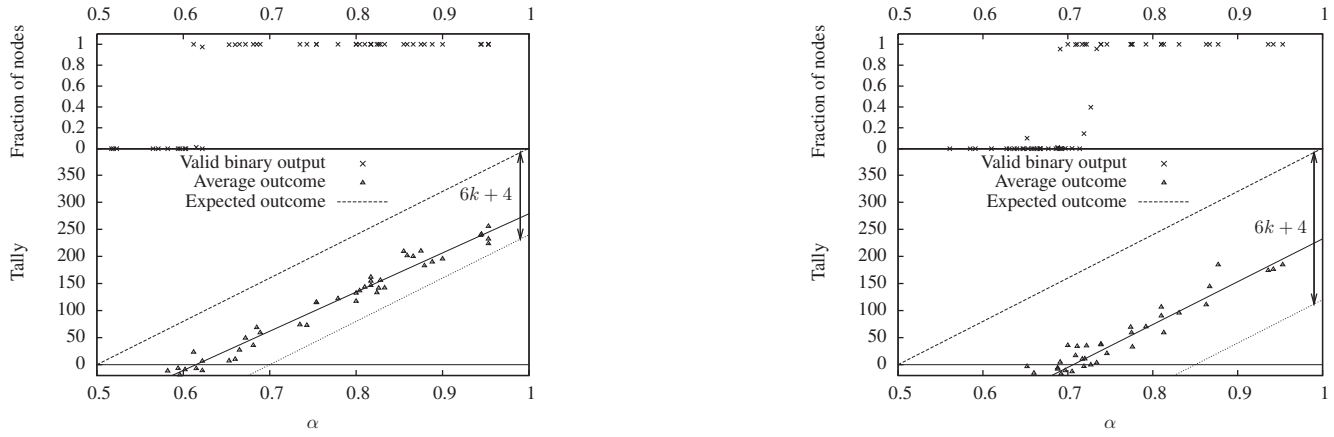
*Proof.* As defined in Lemma 7, groups $i$ and $i+1$ have been *compromised* if $|g_i\cap\mathscr{B}|+|g_{i+1}\cap\mathscr{B}|\geq\sqrt{N}$. Since $g_i$ and $g_{i+1}$ are disjoint, this event is equivalent to $|G_i\cap\mathscr{B}|\geq\sqrt{N}$ where $G_i=g_i\cup g_{i+1}$ for all $i$.

We first show that if $B\geq\frac{N}{2}$ then dishonest nodes will compromise the system regardless of their allocation to groups. We prove the contrapositive statement, so suppose no groups are compromised. Then $|g_i\cap\mathscr{B}|<\sqrt{N}$ for all groups $i$. Summing the inequalities up for all the $\sqrt{N}$ groups, we get $\sum_i|g_i\cap\mathscr{B}|+|g_{i+1}\cap\mathscr{B}|<N$, that is, $2\sum_i|g_i\cap\mathscr{B}|<N$, which implies that $2B<N$ and finally $B<N/2$. Thus if $B\geq\frac{N}{2}$, then *some* pair of consecutive groups is compromised.

We now consider the case where $B<\frac{N}{2}$. Let $\beta=\frac{B}{N}<\frac{1}{2}$ be the proportion of dishonest nodes in the system. By the standard Hoeffding bounds for sampling from a finite population without replacement, the probability that groups $i$ and $i+1$ are compromised is:

$$\begin{aligned} p_i &= \mathbb{P}\left[|G_i\cap\mathscr{B}|\geq\sqrt{N}\right] \\ &= \mathbb{P}\left[|G_i\cap\mathscr{B}|-2\sqrt{N}\beta\geq 2\sqrt{N}\left(\frac{1}{2}-\beta\right)\right] \\ &\leq \exp\left(-4\sqrt{N}\left(\frac{1}{2}-\beta\right)^2\right). \end{aligned}$$

The right-hand function converges to zero exponentially fast in $\sqrt{N}$ since $\beta<\frac{1}{2}$. Using the union bound, the probability that




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

Figure 6: 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 $k = 2$. [400 nodes PlanetLab tested]

*some* pair of consecutive groups have been compromised is:

$$\begin{aligned} p &= \mathbb{P}\left[\bigcup_i \{|G_i \cap \mathscr{B}| \geq \sqrt{N}\}\right] \\ &\leq \sum_{i=1}^{\sqrt{N}} \mathbb{P}\left[|G_i \cap \mathscr{B}| \geq \sqrt{N}\right] \\ &\leq \sqrt{N}\exp\left(-\sqrt{N}(1-2\beta)^2\right), \end{aligned}$$

which also converges to zero as $N$ grows to infinity if $\beta < \frac{1}{2}$.

We can conclude that the probability of a compromise incurs a phase transition when half of the nodes in the system are dishonest. The asymptotic number of dishonest nodes that DPol can tolerate is therefore $N/2$. □

We now evaluate the probability of the system being compromised by simulation. We first assume that dishonest nodes are randomly distributed among the groups. For example, groups could be built consecutively, i.e., the first group is built by picking $\sqrt{N}$ nodes uniformly at random, the second is built by picking uniformly at random $\sqrt{N}$ nodes from the remaining nodes, and so forth. Under this assumption, the probability of violating the condition of consecutive groups being compromised can easily be computed. In Figure 7(a), we plot this probability as a function of $B$ in a $10,000$-node network. It can be seen that for $B < 2,900$ the probability is less than 1%. Therefore, if the deterministic bound is relaxed by using a probabilistic bound instead, the number of dishonest nodes that the system can tolerate is consistently higher. In Figure 7(b), we plot the maximum number of dishonest nodes that the system can tolerate, that is the maximum number of dishonest nodes tolerated to keep the probability of violating the above condition below 1%).

### 5.5. Detecting dishonest nodes with private verifications

We now evaluate the probability of detecting a dishonest node cheating within the bounds derived in Section 4.2 when honest nodes agree to disclose private information with a nonzero probability.

**Theorem 10.** *There is nonzero probability of detecting a dishonest node which misbehaves, even if its impact on each candidate score is less than* $3k+2$.

*Proof.* Consider as a first step the binary case (as shown in Section 4.2.2, the results obtained in the binary case can be easily extended to for $d > 2$). Assume that each node is willing to disclose each of the ballots it sent or received with probability $p_d > 0$. Now suppose a dishonest node sends $k' > k+1$ ballots of the same kind, say $k'$ ballots $(0,1)$. This node is detected by a private verification if at least $k+2$ of its proxies who received a ballot $(0,1)$ disclose it. Therefore, the probability for the dishonest node to be detected is:

$$\sum_{j=k+2}^{k'} \binom{k'}{j} p_d^j (1-p_d)^{k'-j} \ .$$

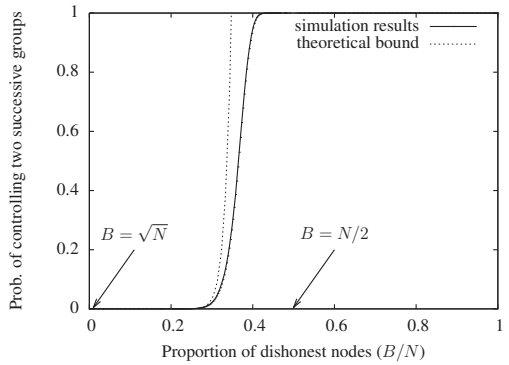
Similarly, consider a dishonest node that receives $n_b^+$ ballots $(1,0)$ and turns $n$ of them into ballots $(0,1)$. The dishonest node is detected by a private verification if at least $n_b^+ - n + 1$ of its clients disclose the ballots they sent to it. Therefore, the probability for the dishonest node to be detected is:

$$\sum_{j=n_b^+-n+1}^{n_b^+} \binom{n_b^+}{j} p_d^j (1-p_d)^{n_b^+-j} \ .$$

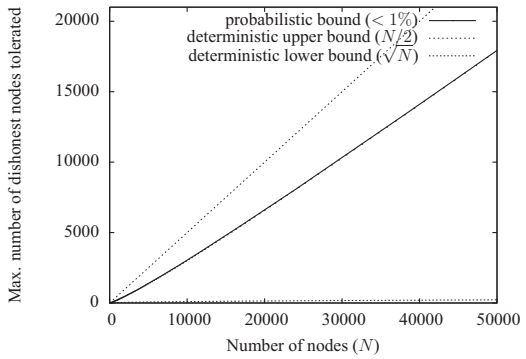
□

## 6. Related Work

We now discuss related distributed voting protocols with particular attention on those that do not depend on intractability of mathematical computations. Like most non-cryptographic voting protocols, DPol ensures privacy via secret sharing techniques. DPol distinguishes itself from related work in the sense




(a) Probability of ring being compromised as the fraction of dishonest nodes grows.




(b) Number of dishonest nodes tolerated to keep probability of compromise below 1%.

Figure 7: Average tolerance to dishonest nodes.

that no participant has a special role, following the peer-to-peer paradigm, which results in increased scalability and robustness.

A large amount of work on secret sharing schemes (introduced by Rivest *et al.* in [31]) has been published in the late 80's. Benaloh [2] proposed a scheme for privately sharing secrets based on polynomials. Since this scheme is homomorphic with respect to addition, it may be used for polling. However, a dishonest participant can easily corrupt the initial shares in the protocol, thus potentially impacting the final outcome to a significant degree.

Assuming a majority of honest participants, Rabin and Ben-Or extended Benaloh's secret sharing and proposed *verifiable secret sharing scheme* (VSS) [24]. Based on VSS, they proposed a secure multi-party computation (MPC) protocol to privately compute 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 input itself (in contrast to DPol where the ballots are in $\{0, 1\}$ for each candidate and therefore the vote is at most $\pm(2k + 1)$), a dishonest participant may still share an arbitrarily high value and thus affect the outcome in a potentially unbounded way. Series of follow-up work on MPC have improved various aspects of the scheme, but only recently begun thinking about making it scalable and usable in practice [6, 5]. The appeal of this class of protocols lies in strong privacy guarantees to participants, including the dishonest ones, but also makes such schemes less suitable for polling applications. Note that the same issues also apply to complex secret sharing scheme and private multiparty computation such as AMPC [21].

In [20], Malkhi *et al.* proposed an e-voting protocol based on AMPC and enhanced check vectors. While powerful, participants of this protocol have distinct and predefined roles (dealers, talliers, and receivers). This may result in decreased scalability as the load of distributing initial ballots to voters falls on a small set of nodes that are not part of the system (i.e., dealers) and robustness if specific nodes fail. Nonetheless, these design choices are fully justified by the requirements inherent in e-voting applications, such as democracy, verifiability, and unconditional accuracy. Instead, polling applications can relax such constraints for the sake of simplicity. Another related distributed voting protocol is the one proposed by Baudron *et al.* [1] but it uses asymmetric cryptography.

At a high level, DPol also relates to distributed ranking schemes. The principle of ranking is similar to polling, in that a participant evaluates the quality of one of her peers by *(i)* locally grading its behavior (input value), and *(ii)* collecting the local grades assigned by the rest of the system. However, to the best of our knowledge, most published work [15, 28] has focused on designing accurate grading mechanism rather than providing efficient polling schemes. Dutta *et al.* [10] consider grading free riders and take into account potential collusion. Nevertheless, none of the proposed ranking schemes provide a global polling mechanism, as grading generally relies on polling only a subset of nodes (peers usually collaborate with a small part of the network). In addition, privacy is generally not addressed in these schemes.

## 7. Conclusion

We considered the distributed polling problem in a social network where participants are concerned about their reputation. To address it, we presented DPol, a simple fully decentralized polling protocol and proved that it can ensure privacy and accuracy, despite the presence of dishonest participants, by means of verification procedures. Our contribution is therefore twofold. First, we define a new model of faulty nodes in distributed systems which incorporates the human and social nature of participants through privacy and reputation concerns. Second, we provide a combination of secret sharing and verification techniques to ensure privacy and accuracy under this model. We find our model of adversaries to be compelling for various non-critical (i.e., not sensitive to small deviations on their outcome) private and secure distributed computation problems in social settings and that DPol thus paves the way for a new area of research in distributed computing. In this spirit, a natural extension of our protocol is to support arbitrary aggregation functions and revisit

traditional problems of distributed computing. For instance, can distributed consensus be reached under this model?

Turning our model and DPol into a solution that can be adopted in practice will require some effort. First, our model of adversaries in social networks asserts that honest participants will always report misbehaviors, and that dishonest participants do not blame honest participants because this may eventually be detected, thus tarnishing the reputation of the dishonest participants. To make these assumptions more realistic in practice, the challenge is to design an automated tool to help users of a social network evaluate and quantify the reputation of a participant by cross-checking information such as tags and social connectivity. Given the selectivity and specificity of such a tool, it would be interesting, within the framework of game theory, to study equilibria and optimal strategies for non-cooperating participants who attach different values to their privacy, their reputation, the outcome of the poll and the accuracy of the tally. Second, DPol relies on a number of assumptions to provide privacy and accuracy guarantees, including the uniform assignment of nodes to groups, synchronization between different phases of the protocol, a limited rate of churn, and that the number of nodes is a perfect square. We intend to address each of these assumptions to make DPol a practical peer-to-peer protocol.

## References


[1] O. Baudron, P.A. Fouque, D. Pointcheval, J. Stern, G. Poupard, Practical Multi-candidate Election System, in: PODC'01: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, ACM, 2001, pp. 274–283.

[2] J. Benaloh, Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret, in: CRYPTO'86: Proceedings of the 6th Annual International Conference on Advances in Cryptology, Lecture Notes in Computer Science, Springer, 1986, pp. 251–260.

[3] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, A. Shraer, Brahms: Byzantine Resilient Random Membership Sampling, Computer Networks 53 (2009) 2340–2359.

[4] M. Castro, B. Liskov, Practical Byzantine Fault Tolerance and Proactive Recovery, ACM Transactions on Computer Systems 20 (2002) 398–461.

[5] I. Damgård, Y. Ishai, M. Krøigaard, J.B. Nielsen, A. Smith, Scalable Multiparty Computation with Nearly Optimal Work and Resilience, in: CRYPTO'08: Proceedings of the 28th Annual International Conference on Advances in Cryptology, Lecture Notes in Computer Science, Springer, 2008, pp. 241–261.

[6] I. Damgård, J.B. Nielsen, Scalable and Unconditionally Secure Multiparty Computation, in: CRYPTO'07: Proceedings of the 27th Annual International Conference on Advances in Cryptology, Lecture Notes in Computer Science, Springer, 2007, pp. 572–590.

[7] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, E. Ruppert, Secretive Birds: Privacy in Population Protocols, in: OPODIS'07: Proceedings of the 11th International Conference on Principles of Distributed Systems, Lecture Notes in Computer Science, Springer, 2007, pp. 329–342.

[8] Digg, 2010. http://digg.com/.

[9] Doodle AG, Facebook Doodle, 2010. http://www.facebook.com/apps/application.php?id=6314677796.

[10] D. Dutta, A. Goel, R. Govindan, H. Zhang, The Design of a Distributed Rating Scheme for Peer-to-Peer Systems, in: P2P Econ'03: Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems, pp. 1–6.

[11] Z. Galil, M. Yung, Partitioned Encryption and Achieving Simultaneity by Partitioning, Information Processing Letters 26 (1987) 81–88.

[12] J. Goodman, C. Verbrugge, A Peer Auditing Scheme for Cheat Elimination in MMOGs, in: NetGames'08: Proceedings of the 7th Annual Workshop on Network and Systems Support for Games, ACM, 2008, pp. 9–14.

[13] R. Guerraoui, K. Huguenin, A.M. Kermarrec, M. Monod, Decentralized Polling with Respectable Participants, in: OPODIS'09: Proceedings of the 12th International Conference on Principles of Distributed Systems, Lecture Notes in Computer Science, Springer, 2009, pp. 144–158.

[14] I. Gupta, K. Birman, P. Linga, A. Demers, R. van Renesse, Kelips: Building an Efficient and Stable P2P DHT through Increased Memory and Background Overhead, in: IPTPS'03: Proceedings of the Second International Workshop on Peer-to-Peer Systems, Lecture Notes in Computer Science, Springer, 2003, pp. 160–169.

[15] M. Gupta, P. Judge, M. Ammar, A Reputation System for Peer-to-Peer Networks, in: NOSSDAV'03: Proceedings of the 13rd International Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, 2003, pp. 144–152.

[16] H.D. Johansen, A. Allavena, R. van Renesse, Fireflies: Scalable Support for Intrusion-Tolerant Network Overlays, in: EUROSYS'06: Proceedings of the 2006 EuroSys Conference, ACM, 2006, pp. 3–13.

[17] P. Kabus, W.W. Terpstra, M. Cilia, A. Buchmann, Addressing Cheating in Distributed MMOGs, in: NetGames'05: Proceedings of the 4th Annual Workshop on Network and Systems Support for Games, ACM, 2005, pp. 1–6.

[18] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The EigenTrust Algorithm for Reputation Management in P2P Networks, in: WWW'03: Proceedings of the 12th International World Wide Web Conference, ACM, 2003, pp. 640–651.

[19] Kremsa Design, Facebook Poll, 2010. http://www.facebook.com/apps/application.php?id=20678178440.

[20] D. Malkhi, O. Margo, E. Pavlov, E-Voting without 'Cryptography', in: FC'02: Proceedings of the Sixth International Financial Cryptography Conference, Lecture Notes in Computer Science, Springer, 2002, pp. 1–15.

[21] D. Malkhi, E. Pavlov, Anonymity without 'Cryptography', in: FC'01: Proceedings of the Fifth International Financial Cryptography Conference, Lecture Notes in Computer Science, Springer, 2001, pp. 108–126.

[22] A. Mislove, A. Post, K.P. Gummadi, P. Druschel, Ostra: Leverging Trust to Thwart Unwanted Communication, in: NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2008, pp. 15–30.

[23] R. Morales, I. Gupta, AVMON: Optimal and Scalable Discovery of Consistent Availability Monitoring Overlays for Distributed Systems, IEEE Transactions on Parallel and Distributed Systems 20 (2009) 446–459.

[24] T. Rabin, M. Ben-Or, Verifiable Secret Sharing and Multi-Party Protocols with Honest Majority, in: STOC'89: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, ACM, 1989, pp. 73–85.

[25] R. Richmond, Facebook Tests the Power of Democracy, 2009.

[26] R. Rivest, Chaffing and Winnowing: Confidentiality without Encryption., RSA Laboratories CryptoBytes 4 (1998).

[27] R. Rivest, W. Smith, Three Voting Protocols: ThreeBallot, VAV, and Twin, in: EVT'07: Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Workshop, USENIX Association, 2007, p. 16.

[28] M. Rodriguez-Perez, O. Esparza, J.L. Muñoz, Analysis of Peer-to-Peer Distributed Reputation Schemes, in: COLCOM'05: Proceedings of the 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing, IEEE, 2005, pp. 1–6.

[29] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, Session Traversal Utilities for NATs (STUN), Technical Report RFC 5389, IETF, 2008.

[30] F. Schneider, A. Feldmann, B. Krishnamurthy, W. Willinger, Understanding Online Social Network Usage from a Network Perspective, in: IMC'09: Proceedings of the 9th ACM SIGCOMM conference on Internet Measurement Conference, ACM, 2009, pp. 35–48.

[31] A. Shamir, How to Share a Secret, Communications of the ACM 22 (1979) 612–613.

[32] M. Sirivianos, K. Kim, X. Yang, SocialFilter: Introducing Social Trust to Collaborative Spam Mitigation, in: INFOCOM'11: Proceedings of the 30th IEEE International Conference on Computer Communications, IEEE, 2011, pp. 2300–2308.

[33] B. Stelter, Facebook's Users Ask Who Owns Information, 2009.

[34] D. Stutzbach, R. Rejaie, Understanding Churn in Peer-to-Peer Networks, in: IMC'06: Proceedings of the 6th ACM SIGCOMM conference on Internet Measurement Conference, ACM, 2006, pp. 189–202.

[35] N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient Online Content

Voting, in: NSDI'09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2009, pp. 15–28.

[36] G. Urdaneta, G. Pierre, M. van Steen, A Survey of DHT Security Techniques, ACM Computing Surveys 43 (2011) 8:1–8:49.

[37] H. Xie, Y.R. Yang, A Measurement-Based Study of the Skype Peer-to-Peer VoIP Performance, in: IPTPS'07: Proceedings of the Sixth International Workshop on Peer-to-Peer Systems, Lecture Notes in Computer Science, Springer, 2007, pp. 1–6.

[38] H. Yu, P. Gibbons, M. Kaminsky, F. Xiao, SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks, in: SP'08: Proceedings of the 2008 IEEE Symposium on Security and Privacy, IEEE, 2008, pp. 3–17.

[39] H. Yu, M. Kaminsky, P. Gibbons, A. Flaxman, SybilGuard: Defending Against Sybil Attacks via Social Networks, IEEE/ACM Transactions on Networking 16 (2008) 576–589.