Public Verification of Private Effort

Giulia Alberini * Tal Moran † Alon Rosen ‡

December 6, 2014

Abstract

We introduce a new framework for polling responses from a large population. Our framework allows gathering information without violating the responders' anonymity and at the same time enables public verification of the poll's result. In contrast to prior approaches to the problem, we do not require trusting the pollster for faithfully announcing the poll's results, nor do we rely on strong identity verification.

We propose an "effort based" polling protocol whose results can be publicly verified by constructing a "responder certification graph" whose nodes are labeled by responders' replies to the poll, and whose edges cross-certify that adjacent nodes correspond to honest participants. Cross-certification is achieved using a newly introduced (privately verifiable) "Private Proof of Effort" (PPE). In effect, our protocol gives a general method for converting privately-verifiable proofs into a publicly-verifiable protocol. The soundness of the transformation relies on expansion properties of the certification graph.

Our results are applicable to a variety of settings in which crowd-sourced information gathering is required. This includes crypto-currencies, political polling, elections, recommendation systems, viewer voting in TV shows, and prediction markets.

Keywords: polling, anonymity, protocols, random graphs, public verifiability, proof of work, CAPTCHA

*School of Computer Science, McGill University. Email: giulia.alberini@mail.mcgill.ca

†Efi Arazi School of Computer Science, IDC Herzliya. Email: talm@idc.ac.il

‡Efi Arazi School of Computer Science, IDC Herzliya. Email: alon.rosen@idc.ac.il

1 Introduction

The Internet enables reciprocal communication on a massive scale. Thus, it has the potential to allow new forms of information gathering and "crowd-sourced" decision making. Some examples (already in widespread use) are political polling, elections (which are a mechanism for achieving consensus among voters about which candidate to put in office), recommendation systems (e.g., based on users' opinions about products and services), prediction markets (leveraging the "wisdom of the crowds" to predict future events) and "crypto-currencies" (such as Bitcoin [15]).

We can think of all these cases as a generalized "opinion poll":the outcome is the result of aggregating the opinions of a large population of Internet users. The "protocols" that implement the poll (and the methods of computing the results) are different in each case, but in all of them we can categorize the participants into three types (some parties may belong to multiple categories):

1. *pollsters* are responsible for collecting the information and publishing the result.
2. *responders* are the parties who provide inputs to the poll.
3. *verifiers* are interested in (should agree on) the result, but may not be active participants.

Although at first glance the examples mentioned above may not necessarily appear to be a *distributed* protocol problem (e.g., in elections there is a central election authority that can broadcast results to everyone), it is natural to consider the case when the central authorities are *untrusted*, and can potentially act maliciously. Viewed this way, verifiable polling is a generalization of the fundamental problem of achieving consensus between mutually-distrustful parties. While in the general polling setting, inputs of various parties could differ and are aggregated into the poll's "tally", the basic consensus problem focuses on the special case in which parties only have to agree on a specific output if all of their inputs match. Correctness of the consensus is guaranteed by the verifiability property of the polling protocol.

In their general form, verifiable opinion polls are also useful as building blocks in more complex protocols. For example, the main technical innovation of Bitcoin, a recently popular "crypto-currency", is in achieving a distributed, decentralized consensus about the currency's public transaction ledger (the record of all Bitcoin transactions) [15].

In the "traditional" setting for the verifiable polling problem (and its variants), the number and identities of the parties are known in advance. Using standard cryptographic techniques, solutions are known to many of them. Techniques for verifiable voting, for example, provide solutions that hide the individual responses of the participants, even after revealing the tally (see subsection 1.5 for references).

Unfortunately, adapting the traditional solutions to work in a decentralized internet environment is non-trivial. One of the major problems encountered in this setting is the lack of identity verification. Strong identity verification on a large scale is expensive, and in many cases completely impractical (e.g., when participants are spread across national boundaries, there might not be a single entity trusted by all of them to certify identities). The mechanisms for identity verification become even more complex when anonymity (or pseudonymity) of the participants is required. In the absence of identity verification, it is impossible to distinguish a fake identity from a real one; this opens the door to "Sybil attacks" (attacks based on creating multiple fake identities).

There are various methods used to mitigate Sybil attacks without requiring identity verification. A recurring idea is to force participants to prove they expended some valuable resource: for example, spending money or performing a computational task. This serves to limit the number of fake identities an adversary can create. In this paradigm, we have no choice but to relax our requirements from the poll: rather than requiring "one vote per participant", we now allow "one vote per effort" (where an "effort unit" corresponds to expending some resource). We call this *effort-based polling*.

The Bitcoin protocol is an excellent example of this type: in effect, consensus is achieved by having parties constantly "vote" on which version of the transaction ledger they accept, where for each "vote" the party must also generate a "proof-of-work" to prove that the required amount of computational effort was expended.[1]

Proofs of work are one of the very few examples of proofs of effort that are *publicly* verifiable. However, they suffer from significant drawbacks. First of all, they are inherently wasteful in that the computation "does nothing" except prove work (indeed, this is one of the strongest arguments against the Bitcoin currency [11]). Secondly, and perhaps more importantly, a party with access to more computing power than most honest responders may gain a hugely disproportionate influence on the results (not to mention the wide disparities between the responders themselves).

1.1 Privately Verifiable Proofs of Effort

An alternative to publicly-verifiable proofs of work, and one that may be potentially easier to achieve, is that of *privately*-verifiable proofs of resource expenditure. One well known example is that of enforcing human involvement in each response. In voting for the "American Idol" TV show, for example, online viewers must solve a CAPTCHA [18] for each vote, but the total number of votes is effectively unlimited. (What makes the CAPTCHA solution privately verifiable is the fact that all currently known CAPTCHAs are *private coin*: in effect, every CAPTCHA is generated together with its solution.)

Beyond being easier to achieve (and not being "wasteful") The "human effort" requirement may be useful when there is a "resource gap" between honest and malicious parties. For example, show producers have significantly more money and access to more computing power than most honest viewers (and there are wide disparities between the viewers themselves)—using proof-of-work in this context could give them a hugely disproportionate influence on the results.

In effect, what CAPTCHAs enable us to achieve is what we call a *privately-verifiable proof of effort* (PPE). Informally, this is an interactive protocol between two parties: if both parties are honest the test returns "true" to both, otherwise the test returns "false" to the honest participant.

Definition 1.1 (PPE, informal)**.** *A two-party protocol is a PPE if it satisfies:*

1. Effort *If both parties honestly follow the protocol, they expend one "effort unit".*
2. σ-Completeness. *If both parties honestly followed the protocol, they will both output "true" at the end of the protocol with probability at least* $1 - \sigma$*.*
3. ε-Soundness. *If one party is malicious (invests less than the required effort) and the other honestly follows the protocol, the honest party will output "false" with probability at least* $1 - \varepsilon$*.*

We note that this definition is necessarily informal, since the term "effort unit" is itself not well defined. In our analysis, we sidestep the problem by reversing the definition: instead of defining a PPE as a proof of effort, we define a "proof of effort" as successful completion of PPE with at least one honest participant (formally, we follow Canetti et al.'s framework for defining CAPTCHAs [3] and define effort in terms of oracle calls; see section 2 for details).

The peer-to-peer nature of PPEs makes them potentially easier to realize than their publicly-verifiable counterparts (which require costly distributed coordination). In Appendix B, we list several potential mechanisms for PPEs, most of which do not require human involvement (making them fully automatizable, and hence scalable). These include proofs of storage, human interaction (including symmetric CAPTCHAs) and leveraging social networks.

[1] The outcome of a Bitcoin "poll" is not a majority-vote, but a randomized selection in which the probability for selecting a "candidate" is proportional to the total effort expended for that candidate. However, this still fits in our generalized polling framework.

1.2 Our Results

While PPEs seem easier to realize, it is not at all clear how to utilize them in order to deal with the problem of a cheating pollster. For instance, in the American Idol example, a malicious CAPTCHA generator can use the solutions to the CAPTCHAs without expending any human effort. Thus, existing CAPTCHAs cannot be publicly verified (hence cannot be used to achieve a consensus about the result of the poll when the generator is untrusted).

Our main result is a new protocol for *publicly*-verifiable effort-based polling, based on any *privately*-verifiable proof-of-effort (PPE). The protocol uses PPEs to generate a "responder certification graph": each responder is a node in the graph while an edge between two responders corresponds to a PPE execution. Loosely speaking, we guarantee that, as long as enough honest users participate in the protocol, a large number of cheating nodes will be publicly detected (note that, unlike most standard definitions of an "honest party", in our case every party controlled by the adversary is considered "cheating", even if it follows the honest protocol exactly).

If each node in the graph is published together with their response to the poll, the poll results cannot be skewed significantly by the pollster without being detected.

In its simplest variant, our protocol assumes that the responder certification graph is sampled at random. This sampling can be performed in a publicly-verifiable way, say by applying a "random-looking" function (e.g., SHA-1) to two nodes' indices to determine if there is an edge between them in the graph. Since our protocol's analysis relies only on expansion properties of such randomly chosen graphs, the construction can potentially be derandomized—using an explicit graph with the appropriate expansion properties, we could remove our assumption about SHA-1 and improve the protocol parameters, at the cost of making the protocol more complex.

We note that while the structure of certification graph is fixed (it depends only on the number of nodes), we allow the adversary to specify the number of nodes (within bounds) and to *arbitrarily* control the assignment of honest nodes to vertices in the graph. We prove that security holds *for every assignment*.

1.3 Main Theorems

The total number of nodes in the certification graph is denoted m and corresponds to the total number of responders (some of whom may be controlled by the adversary). The number of honest responders is denoted by n. We denote by d the average degree of the responder certification graph: this is the number of PPE executions each responder is expected to participate in.

We model our assumption that the pollsters have bounded resources by specifying that a cheating pollster cannot participate in too many successful PPEs with honest responders. In terms of the certification graph, this assumption implies a bound a on the number of "attack edges"—PPE executions in which the cheating pollster participates as one party and convinces an honest responder to accept. The ratio a/d gives a lower bound on the number of "cheating" nodes; an attacker can always create this many cheating nodes without detection by following the protocol honestly. Thus, our security guarantees make sense only when $a/d \ll n$ (we can think of a/dn as a small constant).

We denote by κ the security parameter. Our main theorems guarantee the soundness (a malicious pollster can't cheat undetectably) and completeness (an honest execution will be accepted) of our protocol. For simplicity, we will consider PPEs for which ε (the soundness error of a PPE) is negligible in the security parameter and omit it. For our completeness proof, we require an additional independence property: that for a given node, the probability of failure in each PPE execution is independent (the probability can depend on the node, however).

Theorem 1.2 (Soundness—Informal)**.** *Let A be an adversarial pollster that cannot succeed in more than a PPEs with honest responders. If there are at least $n \geq \alpha m$, $\alpha \in (0,1)$, honest responders to the poll and A controls more than $\Omega(\frac{a}{d})$ of the responses in the poll outcome, then verification will fail with overwhelming probability (in κ).*

See section 4 for the full theorem and proof. Note that our proof holds in the random oracle model, but under a very reasonable assumption about the cryptographic hash function (that the generated graph has good expansion parameters) it holds in the standard model as well.

Theorem 1.3 (Completeness—Informal)**.** *If the pollster is honest, and malicious responders are bounded by $O(m)$ successful PPEs, the probability that verification fails is negligible in κ.*

See section 5 for the full theorem and proof. The bound on successful PPEs by malicious responders is required to guarantee robustness of the protocol—when the pollster is honest the verification should succeed even if some of the responders are malicious.

1.4 Comparison to Verifiable Voting

At a high level, our polling protocol has the same form as most universally verifiable voting protocols (involving an "election authority", "voters", "receipts" and "verification procedure"):

1. The pollster sets up the poll and publishes public parameters on a bulletin-board (modeled as a broadcast channel). This corresponds to the role of the "election authority".

2. Honest responders (corresponding to the "voters") send their responses to the pollster and engage in an interactive proof protocol to ensure that they are expending the correct amount of "effort" for each response. This protocol includes interaction with the pollster and also, unlike most voting protocols, interaction with a subset of other responders.

   The pollster signs the transcript of each communication with a responder and sends this signature to the responder (think of this as the "receipt" in the voting protocol).

3. The pollster publishes the empirical distribution of responses, together with a proof of correctness.

4. The verification procedure consists of both a *local* verification step performed by the responders (which in a voting protocol corresponds to verifying that the voter's receipt appears on the bulletin board) and a *global* verification step performed by the verifiers (which corresponds to the "universal verification" step in voting protocols). Note that responders can also act as verifiers if they wish.

A significant difference between effort-based polling and verifiable voting is the issue of voter identity. In our polling protocol, parties are identified only by self-chosen pseudonyms (for our purposes, a pseudonym is a verification key of a public-key signature scheme). We do not limit the number of pseudonyms a party may generate, or require parties to link their pseudonyms to their real identities.

In contrast, most voting protocols assume each party in the protocol has been identified by a trusted authority, in order to ensure that each voter gets only a single vote. By relaxing this requirement to "one vote per effort expended", we can dispense with the complexity, expense and privacy implications of securely identifying responders.

In particular, our protocol is compatible with completely anonymous polling (if responders communicate with the pollster over *anonymous channels*)—in addition to hiding the link between their real identities and their responses, use of anonymous channels can hide the fact of participation in the poll, with the degree of anonymization depending only on the anonymous channel (in contrast, cryptographic voting protocols that support hiding the voters' participation require a separate non-anonymous registration step, and anonymity depends on the election trustees in addition to the anonymous channel).

1.5 Related Work

Sybil Defense In a "Sybil attack", an adversary creates multiple "fake" identities in order to manipulate a protocol. The problem of establishing trustworthy virtual identities has plagued the Internet from its inception [9]. It is particularly acute in distributed systems with no central authority—without additional assumptions, vulnerability to some forms of Sybil attacks is unavoidable in this case [8]. The paper by [9] deals with the problem of establishing identities. One of the first discussions of trust metrics based on social graphs appears in [12]. The term "Sybil Attack" (attributed to Brian Zill from MSR) was introduced in [8], where it is shown that in the absence of a central certifying authority, some attacks are always possible.

A reputation system for p2p with similar ideas to pagerank (doesn't handle sybils) is developed in [10], and the possibility of using "Turing tests" to limit Sybil nodes is mentioned in [2]. In [6] it is shown that there exists no symmetric sybil-proof reputation mechanism. Since the existing sybil-defense protocols all care about reputations (e.g., determining which nodes are "real" and which are sybils), they all strongly rely on breaking symmetry: having at least one trusted node. Our protocol is symmetric, however we can sidestep the impossibility proof because we don't care about individual nodes' reputations—only about the aggregate opinion of all the nodes.

The technique of random walks on a social networks to bound the effect of Sybil attack is introduced in [22] (see [21] for an expanded version with full proofs). A 2006 survey of sybil attack literature can be found in [13]. An improved version of [22] (slightly different protocol, same goal but better parameters) appears in [20], and a newer protocol to identify sybil nodes in a social graph is presented in [7]. The protocol makes very similar assumptions about the social graph, and Bayesian methods to compute the probabilities that nodes are sybils. Finally, [17] uses the social network graph to aggregate votes for online content. The "vote collector" is assumed to be honest, and votes are collected using max-flow in the social graph.

Most of these techniques implicitly or explicitly use assumptions about expansion properties of social-network graphs. We also make use of the idea that if "adding edges is hard" in an expander graph the adversary is limited in the effect bad nodes can have, but in our case the graph is artificially generated, so we can prove (in the random-oracle model, at least) that our graph has the required properties. On the other hand, the labeling of the graph is adversarial; despite this, we get results that are—in some sense—stronger than the results on social networks: we can bound the total number of "bad" nodes (rather than just their influence).

Verifiable and Private Polling. A widely used technique for privacy-preserving polling is called "randomized response" and was introduced in [19]. The first suggestion for cryptographic verifiability in voting, which also gives a mechanism for establishing anonymous channels (mix-nets) was made in [4]. More recently, the works of [5, 16] propose taking into account *human* voters in End-to-End verifiability, and introduce the notion of separate verification steps for the voter and external observers. Another incarnation of this idea is verifiable (for the pollster) privacy-preserving polling using scratch-off cards [14].

2 Model and Definitions

We now introduce a formal model for capturing the notion of verifiable effort-based polling. The definition will have to address both the syntax of a polling protocol and the issue of the "effort" involved in the protocol execution. To model the effort expended by each one of the protocol participants, we give parties access to an *effort oracle*. The effort spent by each party is measured as the number of calls that party makes to the oracle. To justify this measure, we propose to use "peer-to-peer" protocols that presumably require the expenditure of one call to an effort oracle per successful execution. One well known example for such a protocol is

a CAPTCHA, automatically generated challenges that should be solvable only if given a call to an effort oracle (and moreover accommodate automatic verification of the solution). Other options, (some of which may be more practical) are described in Appendix B.

Before delving into the details, we note that for the convenience of the reader, Appendix C contains a table of the parameters and notation used in the paper.

2.1 Verifiable Effort-Based Polling

An m-responder polling scheme is a multi-party protocol between a pollster, denoted P and m responders, denoted $R_1, \ldots, R_m$. The i^{th} responder holds an input $x_i \in D \cup \{\perp\}$, where D is the domain from which the responses are taken and $\perp$ denotes lack of participation in the poll. In practice m will be an upper bound on the number of responders; We denote by $n < m$ the actual number of (honest) participants. The number of honest responders is known only to the adversary. Thus, the adversary can create "fake" responders by replacing some of the $\perp$ inputs with adversarially chosen values. As the adversary knows all the inputs and controls all the outputs in our protocols, we do not need to consider corrupted responders—the adversary can just replace an honest responder's input with a different one to simulate a corrupted responder.

We give parties access to an oracle denoted E, and let R_i^E (resp. P^E) denote the execution of R_i (resp. P) with access to the oracle E. Let e_i denote the total number of oracle calls made by R_i to E. Let $\langle P^E, R_1^E(x_1), \ldots, R_m^E(x_m)\rangle$ be a random variable describing the output of a protocol execution, where the probabilities are taken over the parties' coin tosses. The output of the protocol takes the form $(\overline{Y}, \overline{z})$, where $\overline{Y} = (\overline{y}, \overline{w})$ denotes the output of the pollster ($\overline{y} = (y_1, \ldots, y_m)$ indicates the outputs of the responders as announced by the pollster, and $\overline{w}$ contains a proof of correctness of the result) and $\overline{z} = (z_1, \ldots, z_m)$ denotes the local outputs of the responders, where z_i corresponds to the local output of R_i following the protocol execution. The role of the local outputs z_i is to enable local verification by the parties.

To make the polling scheme publicly-verifiable we will additionally require the existence of a verifier V that takes $\overline{Y}$ and $\overline{z}$ as inputs (the verification procedure can use the output of the local verification; e.g., global verification could fail if too many responders complain).

Definition 2.1 (Verifiable Effort-Based Polling)**.** *Let $\kappa, m, a \in \mathbb{N}$ and let $\alpha, \theta \in [0,1]$ and $B : \mathbb{N}\times\mathbb{N} \mapsto \mathbb{N}$. An m-responder effort-based polling scheme is said to be* (α, B)-sound *and* θ-robust *if there exists a probabilistic polynomial-time algorithm V such that for any $x_1, \ldots, x_m \in D\cup\{\perp\}$ with $n = \#\{i \in [m] | x_i \neq \perp\}$, the following properties are satisfied:*

Soundness: *For every PPT (Probabilistic Polynomial Time) P^*, if $n \geq \alpha m$ and $\Delta(\overline{x}, \overline{y}) \geq B(a, m)$ then*

$$\Pr\left[V(\overline{Y}, \overline{z}) = \texttt{accept}\right] < 2^{-\kappa} ,$$

where the probability is taken over $(\overline{Y}, \overline{z}) \leftarrow (P^{*E}, R_1^E(x_1), \ldots, R_m^E(x_m))$, a is the total number of oracle calls made by P^* to E,and $\Delta(\overline{x}, \overline{y})$ is the minimum Hamming distance between $\overline{x}$ and some permutation of $\overline{y}$ (i.e., this corresponds to the number of responses changed/added by the adversary).

Completeness: *For every subset $\{i_1, \ldots, i_t\} \subseteq [m]$ of responders (corresponding to malicious responders), if $e_{i_1} + \ldots + e_{i_t} < \theta m$ then*

$$\Pr\left[V(\overline{Y}, \overline{z}) = \texttt{accept}\right] > 1 - 2^{-\kappa},$$

where the probability is taken over $(\overline{Y}, \overline{z}) \leftarrow (P^E, R_1^E(x_1), \ldots, R_m^E(x_m))$.

Informally, we can interpret (α, B)-soundness as a guarantee that if at least an α-fraction of responders are honest, then the adversary cannot change too many responses without getting caught. The influence of the adversary is captured by the function B. Generally, we would

expect $B(a, m)$ to be proportional to the number of responses an honest user could add using a calls to the effort oracle. Thus, an intuitive measure of the protocol's soundness is a bound on the multiplicative advantage of the adversary:

$$C(a) = B(a, m)\frac{d}{a}$$

If the multiplicative advantage is bounded by C, then any adversary who can change $C \cdot \ell$ responses using an optimal cheating strategy could have altered ℓ responses (in expectation) by honestly following the protocol and expending the same amount of effort.

The θ-robustness of the protocol guarantees that if the total effort available to malicious responders is less than θm, then they cannot cause the verification procedure to fail except with negligible probability.

2.2 Formally Defining Proofs of Effort

In the "effort-oracle" model we can fully formalize definition 1.1. Note that while we define PPE to be a two-party protocol, we require soundness to hold even in a concurrent setting, in which a malicious party A^* participates concurrently in multiple executions of the protocol with other parties. To achieve this, we assume each protocol execution has a unique identifier *id* (e.g., in practice this could be a concatenation of the identities of the participating parties and the current time).

Definition 2.2 (PPE)**.** *A protocol $\Pi^E(A, B)$ between two parties A and B is a PPE if it satisfies the following properties:*

1. **σ-Completeness** *If A and B execute an instance of $\Pi^E(A, B)$ and both honestly follow the protocol, then with probability at least $1 - \sigma$ both parties will output "true" at the end of the protocol.*

2. **ε-Soundness** *For every pair of PPT (Probabilistic Polynomial time) A^* and B that execute an instance of $\Pi^E(A^*, B)$ with identifier* id*, if B honestly follows the protocol but A^* does not make at least one oracle call to E with input* id*, then the probability that B outputs "true" is at most ε.*

3 The Protocol

The main technical innovation in this paper is the construction of the Pollster's proof for the correctness of the published results. To do this, we borrow ideas from the literature on defense against Sybil attacks using pre-existing trust relations.

To account for the possibility that an honest responder can fail a PPE execution independently of his honesty, we denote by η_E the fraction of failing PPEs that the protocol tolerates before discarding someone's vote. On the other hand, we indicate by η_V the upper bound on the fraction of responders whose vote can be discarded by the pollster (if the number of discarded votes is greater than η_V, the overall verification will fail). Moreover, in order to avoid denial-of-service attacks caused by malicious responders that intentionally fail all their PPEs, our protocol will require to register for the poll by solving a single-sided PPE (i.e., a PPE that requires effort only from the voters side in order to be successful). With high probability this kind of attack will then be unsuccessful whenever the cheating responders are limited in the amount of effort they can expend. Following, is a high-level description of our protocol. For each of the steps below, the full formal protocol description appears in subsection A.3.

1. **Parameter Announcement.** (Protocol 1) This phase consists of a single broadcast by the pollster, consisting of the public parameters for the poll. The pollster generates a

unique, random identifier id for the poll and public key parameters for a digital signature scheme. We denote by SK, VK the secret and public key respectively.

The public parameters are the tuple $(id, questions, p, VK)$, where $questions$ is the set of poll questions. p is a probability that determines the expected degree of the certification graph (see Appendix C for a full explanation of how it's chosen).

2. **Registration.** (Protocol 2) Each responder R_i samples a private key SK_i and a public key VK_i for their signature scheme, and sends $(addr_i, VK_i)$ to the pollster (where $addr_i$ is the responder's network address). Each responder then solves a single-sided PPE (verified by the pollster). If verification was successful, the pollster adds $(addr_i, VK_i)$ to its list of registered responders.

   When the registration phase is over, the pollster broadcasts the list of registered public keys. Note that the network addresses are not required to appear in the broadcast list. The order of public keys in the list maps each registered responder to a unique index (i.e., the i^{th} key in the list is mapped to index i).

   For each index i, we define N_i to be "the neighborhood of i" in the certification graph. N_i is computed from i and m (the total number of parties) using a cryptographic hash H: $j \in N_i$ iff $H(i, j) \leq p$, where the output of H is treated as a binary fraction in $[0, 1]$ (e.g., H could be SHA-1). Since all of the parameters are public, every party can compute the list of its neighbors in the graph.

   However, while R_i may know the verification key of every neighbor, it does not necessarily know their network addresses. The parties can communicate via the pollster, or alternatively, the pollster can send each party i the network addresses of all its neighbors in the graph.

3. **Responder Certification (PPE execution).** (Protocol 3) As just described, every pair of responders is paired in a PPE instance with probability p. Now, for each $R_j \in N_i$, responder R_i engages in a PPE with R_j. The actual execution is peer-to-peer, however the communication may be facilitated by the pollster (e.g., the pollster's website can be used as a conduit for a VOIP chat). If the PPE execution succeeded (both parties received "true"), the parties sign each other's public keys (concatenated with a unique "poll identifier", to prevent the signatures being reused in other polls) and send the signed values to each other.

4. **Poll Response.** (Protocol 4) Every responder R_i sends to the pollster the results of the certification phase (a signature on VK_i from each neighbor with which it successfully completed a PPE) and x_i, the actual response to the poll questions.

5. **Results and Proof.** (Protocol 5) We can think of the responders as nodes of a graph G_c in which they are connected by edges if and only if they were supposed to interact through a PPE. Let $V = \{1, \ldots, m\}$ denote the set of responders and $E := \{(i, j) | i, j \in V, H(i, j) < p\}$ the set of edges. We call $G_c = (V, E)$ the "certification graph". Note that anyone can compute G_c given the serial numbers associated to the responders and p. Then, as a "proof of correctness" the pollster publishes the graph consisting of the following[2]:

   Node labels: For each responder R_i he publishes $(x_i, sig_{SK_i}(x_i), VK_i, id_i)$.

   Edge signature: For each successful PPE he publishes $(sig_j(VK_i), sig_i(VK_j))$, where VK_i, VK_j are the public keys of the responders involved in it.

[2]the information as described is redundant (e.g., the list of deleted nodes can be computed from the list of edge signatures and node labels), but we describe it in this way to make the description of the verification process simpler.

List of deleted nodes: The list of all nodes whose response will not count in the result because they failed more than a η_E fraction of the PPEs.

The empirical distribution of the responses can be computed by counting the votes associated to the non-deleted nodes. Note that the graph published by the pollster, call it G_p, is composed of the same nodes as G_c, but it's missing all the edges associated to unsuccessful PPEs. So, $G_p = (V, E')$ is a subgraph of $G_c = (V, E)$ where (i, j) is in E' if and only if R_i and R_j *successfully* interacted through a PPE.

6. **Verification.** (Protocol 6) The procedure is divided in two steps:

   Local verification (performed by each responder) consists of verifying that the corresponding node was published correctly, as were the edge signatures in which he was involved (no adjacent edge is missing, and all the adjacent edges in the graph were verified with a successful PPE). If any of these verifications fail, the responders sends a "complaint".

   Global verification (can be performed by anyone) consists of checking that all the nodes, and no others, that failed more then $\eta_E d$ edges are indeed marked as deleted. To verify if a node i is marked correctly, the verifier needs to find its neighbors in the graph (by computing the hash function $H(i, j)$ for every $j \neq i$) and checking how many of the signed edges appear in the published graph. Then, the verifiers need to check that no more than a η_V fraction of the nodes were deleted and that not "too many" valid complaints were sent.

An adversarial pollster can attempt to manipulate results either by changing the responses associated with honest nodes or by "controlling" many nodes (they will be nodes that do not correspond to any honest participant, but appear in G_p and their "behaviour" is dictated by the pollster), such that the overall empirical distribution differs from the empirical distribution over the honest nodes. In the former case, the local verification will detect the adversary and many valid complaints will be sent. In the latter case, we use an expansion property of the graph to prove that any large enough set of "bad" nodes (nodes that are controlled by the adversary) must have many edges to its complement in the graph. Thus, an adversary who wants to control a big enough set of nodes must succeed in many PPE executions with honest nodes; since the adversary is bounded in the number of successful PPE executions, it will be caught with high probability.

The protocol also provides a measure of robustness against malicious responders. Cheating responders cannot undetectably *modify* the results for the same reason that a cheating pollster cannot. However, they can attempt to launch a denial-of-service attack by causing verification to fail. As explained above, the single-sided PPE in step 4 will prevent this form of attack, as long as the cheating responders are limited enough in the amount of effort they can expend.

4 Soundness

To prove the soundness of our protocol we need to show that the number of votes that the adversary can control is at most proportional to the amount of effort that he is willing to invest. That is, whenever the adversary is able to control a "meaningful" amount of votes that is significantly greater that the number of votes that she could have controlled by honestly following the protocol (with the same effort investment), our verification procedure will fail with overwhelming probability. The proof of such a result will rely both on the security of the signature schemes and on an expansion property of the graph G_p published by the pollster as proof of correctness.

The use of the signatures is entirely straightforward: they prevent the adversary from changing honest users' votes and from claiming a failed PPE with an honest user was successful (to

do this, the adversary would have to forge the honest node's signature). Similarly, the pollster's signature on the honest user's registration information and the signatures of its neighboring nodes allows the honest user to verifiably complain about being omitted from the count despite successfully completing the requisite number of PPEs. The "meat" of the security proof is in the analysis of the certification graph, and that will be the focus of this section.

As described in the previous sections, in a m-responders polling scheme, each responder holds an input $x_i \in D \cup \{\perp\}$, where D is the domain from which the responses are taken and $\perp$ denotes lack of participation in the poll. We call a node *honest* if its corresponding party participated in the poll (its input was not $\perp$). We call a node *bad* if it is not honest but its response in the output is not $\perp$. Finally, we say a node is *deleted* if it failed more than $\eta_E d$ of the PPEs it was assigned (note that both honest and bad nodes may be deleted), where d is the number of PPE executions each responder is expected to participate in. Note that for soundness to hold, we need that at least a certain portion, say α, of the responders are actually honest. That is, we need at least αm responders to participate to the poll by sending an input. The adversary could in theory be the one controlling the remaining $(1-\alpha)m$ votes by replacing $\perp$ as an actual vote in the output and by spending the effort he has available. We prove that if the number of controlled nodes is significantly greater than the number of votes he would have controlled by acting honestly, then he will be detected with high probability.

In order to prove soundness, we bound separately the number of bad nodes (corresponding to "fake" parties generated by the adversary) and the number of changes the adversary can make to the input of honest nodes (that is, responder R_i voted x_i and the pollster output $y_i \in D \setminus \{x_i\}$ or $y_i = \perp$ instead). To prove the first bound, we rely on an expansion property of the graph G_p output by the pollster. In the following subsection we give a general definition of such a property and we prove some lemmas that will be useful for our proof.

4.1 Large-Set Expanding Property

The LSE property is similar to the "jumbled" graphs of Thomason, but is weaker since we don't care if small sets do not expand. This lets us get better LSE parameters for random graphs than are possible for the standard jumbled graphs.

Definition 4.1 (Large-Set Expanding (LSE))**.** A graph $G = (V, E)$, with $m = |V|$, is said to be (K, ρ, q)-*LSE* if for every pair of disjoint sets $A, B \subset V$ such that $K \leq |A| \leq m/2$, $|B| \geq m - |A| - \rho$ it holds that the set of edges between A and B, denoted by $e(A, B)$, has cardinality greater than $|A||B|q$.

In our analysis K will denote a bound on both the maximum number of bad nodes that we will allow and on the minimum number of good nodes that we require, ρ will be the maximum number of deleted nodes and q a function of the probability that two voters have to run a PPE.

Lemma 4.2. *Let $G(m, p) = (V, E)$ be a random graph with $p = d/m$, For every $\rho \geq 1$, $\rho \in \mathbb{N}$ and every $b > 1$, if*

$$d > \frac{4b^2 m}{m - 2\rho}(\ln m + 1)$$

then G is $(K, \rho, \frac{b-1}{b}p)$-LSE with probability at least $1 - 2^{-\kappa}$ for $K = \kappa + (\rho + 2)\ln m + \rho$ (where the probability is over the choice of graph).

Proof. Consider an arbitrary pair of sets $A, B \subset V$ such that $K \leq |A| \leq m/2$, $|B| = m - |A| - r$ with $1 \leq r \leq \rho$. Define the random variable $X_{i,j}$ to be the indicator variable for the event $(i, j) \in E$.

Since G is a random (m, p)-graph, the $X_{i,j}$'s are independent and $Pr[X_{i,j} = 1] = p$. Then

$$|e(A,B)| = \sum_{i\in A}\sum_{j\in B} X_{i,j}$$
$$E[|e(A,B)|] = |A||B|p = \mu$$

For $A, B \subset V$ such that $K \le |A| \le \frac{m}{2}$ and $m - |A| - \rho \le |B| \le m - |A|$, let $Bad(A,B)$ be the event that

$$|e(A,B)| < \frac{b-1}{b}\mu$$

(For A, B not satisfying the size restrictions, we define $Bad(A,B)$ to be the null event.)

To prove the lemma, we must bound the probability that $Pr\left[\exists A, B \subset V : Bad(A,B)\right]$ First, since the $X_{i,j}$'s are independent, by the Chernoff bound we have for any disjoint sets A and B:

$$\begin{aligned}
&\Pr[|e(A,B)| < \frac{b-1}{b}\mu] \\
&\quad \le \exp\{-\frac{\mu}{2b^2}\} = \exp\{-\frac{|A||B|p}{2b^2}\} \\
&\quad = \exp\{-\frac{|A|(m-|A|-r)p}{2b^2}\} \le \exp\{-\frac{|A|(m/2-r)p}{2b^2}\}
\end{aligned}$$

Next, we bound the probability that there exist two sets A and B of fixed sizes $|A| = x$, $|B| = m - x - r$ such that $Bad(A,B)$ occurs. Denote

$$\varepsilon = \Pr\left[\bigcup_{\substack{A,B\subset V, A\cap B=\emptyset \\ |A|=x, |B|=m-x-r}} Bad(A,B)\right]$$

By the union bound, this probability is bounded by

$$\begin{aligned}
\varepsilon &\le \sum_{\substack{A,B\subset V, A\cap B=\emptyset \\ |A|=x, |B|=m-x-r}} \Pr\left[|e(A,B)| < \frac{b-1}{b}\mu\right] \\
&\le \sum_{\substack{A,B\subset V, A\cap B=\emptyset \\ |A|=x, |B|=m-x-r}} \exp\{-\frac{|A||B|p}{2b^2}\} \\
&= \sum_{\substack{A,B\subset V, A\cap B=\emptyset \\ |A|=x, |B|=m-x-r}} \exp\{-\frac{x(m-x-r)p}{2b^2}\} \\
&= \binom{m}{x}\binom{m-x}{r}\exp\{-\frac{x(m-x-r)p}{2b^2}\} \\
&\le \binom{m}{x}\binom{m}{r}\exp\{-\frac{x(m-x-r)p}{2b^2}\} \\
&\le \left(\frac{me}{x}\right)^x\left(\frac{me}{r}\right)^r\exp\{-\frac{x(m-x-r)p}{2b^2}\}
\end{aligned}$$

Since $|A| = x \le \frac{m}{2}$,

$$\exp\{-\frac{x(m-x-r)p}{2b^2}\} \le \exp\{-\frac{x(m/2-r)p}{2b^2}\}$$

Hence

$$\begin{aligned}
\varepsilon &\leq \exp\left\{x(\ln m + 1 - \ln x) + r(\ln m + 1 - \ln r) - x\left(\frac{m}{2} - r\right)\frac{p}{2b^2}\right\} \\
&\leq \exp\left\{-x\left(\frac{d}{4b^2} - \frac{dr}{2b^2 m} - \ln m - 1 + \ln x\right) + r(\ln m + 1 - \ln r)\right\} \\
&\leq \exp\left\{-x\left(\frac{d}{4b^2} - \frac{dr}{2b^2 m} - \ln m\right) + r(\ln m + 1)\right\} \\
&\leq \exp\left\{-x + r(\ln m + 1)\right\}
\end{aligned}$$

Where the last two inequalities hold as long as $\ln x > 1$ (which is always true assuming $K > 3$), $\ln r \geq 0$ (which is always true for $r \geq 1$) and $d > \frac{4b^2 m}{m-2r}(\ln m + 1)$.

Applying the union bound again, we get

$$\begin{aligned}
&\Pr\left[\exists A, B \subset V : Bad(A, B)\right] \\
&\leq \sum_{x=K}^{m/2}\sum_{r=1}^{\rho} Pr\left[\bigcup_{\substack{A\subset V \\ |A|=x}}\bigcup_{\substack{B\subset V \\ |B|=m-x-r}} \{|e(A,B)| < \frac{b-1}{b}\mu\}\right] \\
&\leq \frac{m}{2}\rho e^{-K+\rho(\ln m+1)} \\
&\leq 2^{-\kappa}
\end{aligned}$$

since $K > \kappa \ln 2 + (\rho + 1)\ln m + \ln \rho + \rho$. □

In our analysis, we will use this lemma to prove that the certification graph G_c is indeed expanding with specific parameters K, ρ, and q. We will then need to use the following lemma, in order to prove that our protocol is sound:

Lemma 4.3. *Consider a graph $G = (V, E)$ with $m = |V|$ nodes. Let $G' = (V, E')$ be the graph obtained from G by deleting at most s edges per node. If G is (K, ρ, q)-LSE, then G' is $(K, \rho, q - \frac{2s}{m-2\rho})$-LSE.*

Proof. For simplicity let $q' = q - \frac{2s}{m-2\rho}$. Consider $A, B \subset V$ such that $K \leq |A| \leq m/2$ and $m - |A| - \rho \leq |B| \leq m - |A|$. We want to prove that $|e_{G'}(A, B)| > |A||B|q'$, where $e_G(\cdot, \cdot)$ indicates the set of edges between A and B in the graph G.

First, by assumption the maximum number of edges that can be missing in G' from v are exactly s. Therefore, the maximum number of edges that can be missing in G' from the set of all edges with at least one node in A is $|A|s$. In the worst case, for us, all the missing edges were part of $e(A, B)$ in G. Thus,

$$|e_{G'}(A, B)| \geq |e_G(A, B)| - |A|s$$

Now we can use the fact that G is (K, ρ, q)-LSE to obtain the following:

$$|e_{G'}(A, B)| \geq |e_G(A, B)| - s|A| > |A||B|q - s|A| \ .$$

It remains to show that $|A||B|q' \leq |A||B|q - s|A|$. From $q' = q - \frac{2s}{m-2\rho}$ and $|A| \leq m/2$ we get

$$\begin{aligned}
|A||B|q' &= |A||B|\left(q - \frac{2s}{m-2\rho}\right) \\
&\leq |A||B|\left(q - \frac{s}{m - |A| - \rho}\right) \\
&= |A||B|q - |A|s\left(\frac{|B|}{m - |A| - \rho}\right) \\
&\leq |A||B|q - |A|s \ ,
\end{aligned}$$

from which we can conclude $|e_{G'}(A, B)| > |A||B|q'$ as required. □

4.2 Main Theorem and Proofs

We can now apply the results obtained in the previous subsection specifically to our protocol. Let a denote the maximum number of effort oracle calls that the adversary is willing to make and let $K = \kappa + (\eta_V m + 2)\ln m + \eta_V m$.[3] Formally, we prove

Theorem 4.4 (Soundness)**.** *Let*

$$b = \sqrt{\frac{d(\frac{1}{2} - \eta_V)}{2(\ln m - 1)}} \ .$$

If $b > (\frac{1}{2} - \eta_V)/(\frac{1}{2} - \eta_V - \eta_E) > 1$*, then the protocol of section 3 is an* (α, B)*-sound verifiable polling protocol for*

$$\alpha = K/m + \eta_V$$

and

$$B(a, m) = \max\left\{K, \left(\frac{b}{(b-1)(\frac{1}{2} - \eta_V) - b\eta_E}\right)\frac{a}{d}\right\} + \theta m \ .$$

When a is sufficiently large (so we can ignore the K "free" responses), this implies the multiplicative advantage of the adversary is bounded by

$$C(a) = \left(\frac{b}{(b-1)(\frac{1}{2} - \eta_V) - b\eta_E}\right) + \frac{\theta m d}{a} \ .$$

One way to interpret this is that an adversary gets resources equivalent to θm honest users "for free", but any more powerful adversary has multiplicative advantage bounded by

$$C^* = \left(\frac{b}{(b-1)(\frac{1}{2} - \eta_V) - b\eta_E}\right) + 1$$

(recall that an honest user must solve, in expectation, d PPEs during the protocol execution, so an adversary more powerful than that must have $a > \theta m d$).

Proof. As we discussed at the beginning of the section, there are two ways for the pollster to affect the vote count:

1. By possibly controlling some of the nodes.
2. By replacing or deleting the votes of honest participants.

For the latter, the bound relies on the security of the signature scheme and on the local verification of honest parties. In fact, the signature scheme ensures that the adversary cannot modify responses (with a $y_i \neq \perp$) (since that would require forging a signature compatible with the node's verification key). Thus, the local verification of honest nodes will catch the adversary deleting or completely replacing nodes; Global verification fails whenever more then θm nodes complain—thus, the number of deleted/replaced nodes in a successful protocol execution can be at most θm.

It is left to show that if the number of controlled nodes is higher than B, then global verification will fail. The proof proceeds as follow:

- Using Lemma 4.2 and Lemma 4.3 we prove that G_p is LSE with high probability.
- We will then have an lower bound on the number crossing edges between a possible set of bad nodes and the set of honest nodes.

[3] Recall that η_V is a parameter denoting the max fraction of nodes that can be deleted before verification fails.

- We conclude by noticing that the pollster, in order to control a set of nodes larger than B, would have had to succeeded in more than a PPEs involving honest participants.

Let F denote the nodes in G_p corresponding to voters that have failed more than $\eta_E d$ PPEs. It must be that $|F| \leq \eta_V m$, otherwise the verification procedure would fail. Let B and H denote the set of bad and honest nodes, respectively, that have not been labeled as "deleted". Thus B, H and F are disjoint sets whose union is V. That is, since we have a total of m nodes, if $|B| = x$ then $|H| = m - x - |F|$. Recall that a successful PPE corresponds to an edge in G_p. Thus, a lower bound on the number of edges in G_p between the sets B and H translates to a lower bound on the number of PPEs in which the adversary must have succeeded, and hence on the number of oracle calls made by the adversary.

Note that from Lemma 4.2, we know that G_c is $(K, \eta_V m, \frac{b-1}{b}p)$-LSE with probability at least $1 - 2^{\kappa}$. Thus, from Lemma 4.3, we can conclude that, with probability at least $1 - 2^{\kappa}$, G_p is $(K, \eta_V m, \frac{b-1}{b}p - \frac{2\eta_E d}{m - 2\eta_V m})$-LSE. Wlog assume

$$|B| < m/2 \qquad \text{and} \qquad |B| \geq \max\left\{K, \left(\frac{b}{(b-1)(\frac{1}{2} - \eta_V) - b\eta_E}\right)\frac{a}{d}\right\}$$

(the case $|H| < m/2$ is analogous, using $|H| \geq (\alpha - \eta_V)m \geq K$). Then, we get:

$$\begin{aligned}
|e(B,H)| &> |B||H|\left(\frac{b-1}{b}p - \frac{2\eta_E d}{m - 2\eta_V m}\right) \\
&\geq |B|\left(m - |B| - \eta_V m\right)\left(\frac{(b-1)d(1-2\eta_V) - 2bd\eta_E}{mb(1-2\eta_V)}\right) \\
&\geq \left(\frac{2b}{(b-1)(1-2\eta_V) - 2b\eta_E}\right)\frac{a}{d}\left(\frac{m}{2} - \eta_V m\right)\left(\frac{(b-1)d(1-2\eta_V) - 2bd\eta_E}{mb(1-2\eta_V)}\right) \\
&= \left(\frac{2b}{(b-1)(1-2\eta_V) - 2b\eta_E}\right)\frac{a}{d}\left(\frac{m(1-2\eta_V)}{2}\right)\left(\frac{d[(b-1)(1-2\eta_V) - 2b\eta_E]}{mb(1-2\eta_V)}\right) \\
&= a
\end{aligned}$$

Thus, with probability at least $1 - 2^{\kappa}$, $|e(B,H)| > a$ which contradicts the assumption of the adversary being limited to a successful PPEs.

Therefore, the number of votes controlled by a pollster that invests a effort oracle calls must be lower than $\max\{K, \left(\frac{b}{(b-1)(\frac{1}{2} - \eta_V) - b\eta_E}\right)\frac{a}{d}\} + \theta m$, as wanted.

□

5 Completeness

It is now left to show that in the case of an honest pollster, the verification procedure will succeed with overwhelming probability. Even when dealing with an honest pollster, we still need to take into account the possibility that malicious voters might try to force the verification to fail. This can be done by registering for the poll but aborting in all the PPE executions. Such a strategy will force the verification procedure to label the node as *deleted* and all its edges as *failing*. It will thus increase the number of *deleted* nodes which, for the verification to output `accept`, needs to be smaller than $\eta_V m$.

To make sure that such an attack would require the adversary to expend actual effort, we require each responder to solve a single-sided PPE (where the effort is required only from the responders) in order to be allowed to participate to the poll. We think of the number of maliciously controlled nodes as bounded by θm, where θ will depend on the "effort" invested by the malicious voters. Theorem 5.2 gives a bound on η_V as a function of θ and κ that will enable

the verification procedure, in case of an honest pollster, to output `accept` with probability at least $(1 - 2^{\kappa})$.

To prove the main theorem of this section, we will require the following lemma (whose proof is below):

Lemma 5.1. *Assuming static corruption, the probability that malicious responders with a θm bound on effort (in total) can control* $\max\{3\theta md, 3\kappa\}$ *edges in the certification graph is bounded by $e^{-\kappa}$.*

Theorem 5.2 (Completeness)**.** *Let θm denote the maximum number of effort oracle calls that can be made by malicious responders and*

$$\eta_V^{min} = \theta + \frac{3 \cdot \max\left\{\frac{\kappa}{md}, \theta\right\}}{\eta_E} + \frac{2\sigma}{\eta_E}\left(1 + \max\left\{2, \frac{2\kappa}{md\sigma}\right\}\right)$$

If the pollster follows the protocol honestly, $\eta_E > 0$ and $\eta_V \geq \eta_V^{min}$ then the probability that the verification procedure outputs `accept` *is at least $1 - 2^{-\kappa}$.*

We note that for non-trivial soundness, the values of η_E and η_V are further constrained. See Appendix C for a discussion on choosing the parameters.

Proof. Let $G(m, p) = (V, E)$, with $p = d/m$, be the random graph generated by the pollster. Recall that an edge (i, j) is labeled as *failing* whenever the PPE between i and j fails. We denote by σ the probability that such an event occurs between honest voters. Moreover, η_E is the highest fraction of PPEs that can fail before a node/voter gets labeled as *deleted*, and η_V is the maximal fraction of deleted nodes accepted by the verification procedure.

Let $X_{i,j}$ denote the indicator random variable for the event "$(i, j) \in E$ is a failing edge". Note that, if i and j are both honest, the $X_{i,j}$'s are independent and $Pr[X_{i,j} = 1] = \sigma p$. Let $X = \sum_{i \in V} \sum_{j \in V} X_{i,j}$. Then, $E[X] = m^2 \sigma p = md\sigma$ and X denotes the number of failing edges in the graph. Since each edge affect 2 nodes, $2X$ is actually the cardinality of the set containing (with repetitions) all the nodes affected by failing X edges. Note that for a node to be labeled as deleted, such a node needs to be connected to at least $d\eta_E$ failing edges, which means that such a node has been counted at least $d\eta_E$ times in $2X$. Thus, the expected number of deleted nodes in case of honest responders is bounded by $2X/d\eta_E$. Now, in our analysis, we need to take into account that, in the worst case scenario, there will be θm nodes maliciously controlled who will intentionally fail all their PPE's. Therefore, we will have to account for the following:

1. The malicious nodes (which are θm) will be failing nodes;
2. Enough bad edges will cause an honest node to be marked deleted. However, by Lemma 5.1, with high probability the malicious responders cannot affect more than $3 \cdot \max\{\kappa, \theta md\}$ honest edges. Which means that at most another $3 \cdot \max\{\kappa/d, \theta m\}/\eta_E$ nodes can be "forced" to be labeled as deleted.

To conclude, we want to prove that the probability that $\frac{2X}{d\eta_E} + \theta m + \frac{3\theta m}{\eta_E}$ is greater than $\eta_V m$ is negligible. Let $\eta_V = \theta + \frac{3\theta}{\eta_E} + \frac{2\sigma}{\eta_E}(1 + \delta)$. (we will set δ below.) Then,

$$\begin{aligned}\Pr\left[\frac{2X}{d\eta_E} + \theta m + \frac{3\theta m}{\eta_E} > \eta_V m\right] &= \Pr\left[\frac{2X}{d\eta_E} > \frac{2\sigma}{\eta_E}(1+\delta)m\right] \\ &= \Pr\left[X > (1+\delta)md\sigma\right]\end{aligned}$$

By the Chernoff Bound,

$$\Pr\left[X > (1+\delta)md\sigma\right] \leq \exp\left\{-\frac{\delta^2}{2+\delta}md\sigma\right\}$$

Setting $\delta = \max\left\{2, \frac{2\kappa}{md\sigma}\right\}$, we ensure that $\Pr\left[X > (1+\delta)md\sigma\right] \leq e^{-\kappa}$. □

We conclude by proving Lemma 5.1, as a corollary of the following claim:

Claim 5.3. *Let $S \subset V$ be an arbitrary set of vertices and denote $\delta = \max\{2, 2\kappa/(|S|d)\}$. Then*

$$\Pr\left[|\{(i,j) \in E | i \in S\}| > (1+\delta)d|S|\right] < e^{-\kappa}$$

(i.e., the probability S has more than $(1+\delta)d|S|$ edges is bounded by $e^{-\kappa}$).

Proof. For every pair of vertices $i, j \in V$, let $X_{i,j}$ be the indicator variable for the event $(i,j) \in E$. By definition, $E[X_{i,j}] = p$. Denote $X = \sum_{\substack{i \in S \\ j \in V}} X_{i,j}$ the number of edges adjacent to S. Then $E[X] = mp|S| = d|S|$. By Chernoff,

$$\Pr\left[X_i > (1+\delta)d|S|\right] \leq \exp\left\{-\frac{\delta}{2/\delta+1}d|S|\right\} \leq \exp\left\{-\frac{\delta}{2}d|S|\right\} = \exp\{-\kappa\}$$

□

Proof of Lemma 5.1. Since the pollster randomly shuffles the nodes in the certification path during the registration phase, any set of responders is assigned a random set of nodes in the certification graph. By symmetry, we can consider the probability for any specific set of size θm. The result follows by setting $|S| = \theta m$ in Claim 5.3. □

6 Discussion and Open Questions

General Verifiable Computation Among Anonymous Participants While we state our main results in terms of polling, the security guarantee we give is that the final published graph does not contain too many "bad" nodes. It may be possible to leverage this technique for doing more general computations, where the edges in the graph correspond to a private computation between two parties, and the final goal is a joint, publicly-verifiable computation (in this case, the "responses" might be some intermediate public values of the computation).

Parallel and Distributed Verification The verification procedure in our protocol is highly parallelizable: each responder must verify three properties, each of which can be done by reading only a small part of the graph:

- that her own node was correctly published on the bulletin-board (requires $O(m)$ evaluations of the hash function, but only $O(d)$ communication),
- that the total number of deleted nodes was small (requires reading a small list of nodes),
- and that no edges were missed (this is a local property of each potential edge that can be computed from the node labels and the size of the graph).

The only non-local part in the verification is the aggregation of the results from all the nodes. However, by publishing a small amount of additional information, this computation can be distributed as well. Given an aggregation tree, where each node aggregates the results from its children, a verifier can check a single local neighborhood and a path from that neighborhood to the root in the tree. Thus, if we can assume that enough honest responders will participate in verification, the total amount communication for each responder can be made logarithmic in the size of the graph.

Practicality of the Protocol The parameters achieved by our protocol are not quite good enough to be practical for interaction-based PPEs (the degree of the graph would be about 180 for reasonable parameters). However, this may already be good enough for PPEs that can be automated (for example, the social-network based PPE). Moreover, we believe further research can significantly improve the efficiency.

Improving Efficiency by Using Hypergraphs. Our bound on the degree of the graph may be slightly high for some uses of the protocol. However, we can extend the PPE definition to a multi-party setting, in which several parties certify each other simultaneously (e.g., using a multi-person chat, such as "Google Hangout" or "Skype"). This has the potential of significantly lowering the degree. Extending our protocol in this way may be an interesting direction for future work.

Improving Efficiency by Using Explicit Graphs. Our bound on the degree of the graph is for a randomly chosen graph. In particular, our soundness analysis includes the event that the chosen graph is not a good expander as a failure mode. Thus, we require the properties to hold for random graphs *with overwhelming probability*. However, it is fairly easy to prove that graphs with better parameters (e.g., lower degree for the same expansion rate) *exist*: if we have an explicit representation of such a graph, soundness will hold unconditionally.

References

[1] G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In *ASIACRYPT*, pages 319–333, 2009.

[2] B. Awerbuch and C. Scheideler. Group spreading: A protocol for provably secure distributed name service. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 183–195. Springer, 2004.

[3] R. Canetti, S. Halevi, and M. Steiner. Mitigating dictionary attacks on password-protected local storage. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 160–179. Springer, 2006.

[4] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. 24(2):84–88, 1981.

[5] D. Chaum. E-voting: Secret-ballot receipts: True voter-verifiable elections. 2(1):38–47, Jan./Feb. 2004.

[6] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, P2PECON '05, pages 128–132, New York, NY, USA, 2005. ACM.

[7] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*. The Internet Society, 2009.

[8] J. R. Douceur. The sybil attack. In P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[9] C. M. Ellison. Establishing identity without certification authorities. In *USENIX SSYM '96*, SSYM'96, page 7, Berkeley, CA, USA, 1996. USENIX Association.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, pages 640–651, 2003.

[11] P. Krugman. Bits and barbarism. *New York Times*, Dec. 2013. `http://www.nytimes.com/2013/12/23/opinion/krugman-bits-and-barbarism.html`.

[12] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *USENIX SSYM '98*, SSYM'98, pages 18–18, Berkeley, CA, USA, 1998. USENIX Association.

[13] B. N. Levine, C. Shields, and N. B. Margolin. A survey of solutions to the sybil attack. Tech Report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[14] T. Moran and M. Naor. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In S. Vaudenay, editor, *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 88–108. Springer-Verlag, May 2006.

[15] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, May 2009.

[16] C. A. Neff. Practical high certainty intent verification for encrypted votes, October 2004. `http://www.votehere.net/vhti/documentation/vsv-2.0.3638.pdf`.

[17] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In J. Rexford and E. G. Sirer, editors, *NSDI*, pages 15–28. USENIX Association, 2009.

[18] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Advances in CryptologyEUROCRYPT 2003*, pages 294–311. Springer, 2003.

[19] S. Warner. Randomized response: a survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, pages 63–69, 1965.

[20] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. *IEEE/ACM Trans. Netw.*, 18(3):885–898, 2010.

[21] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman. Sybilguard: defending against sybil attacks via social networks. Expanded Technical Report IRP-TR-06-01, Intel Research, Pittsburgh, Pittsburgh, PA, Jun 2006. `http://www.pittsburgh.intel-research.net/people/gibbons/papers/sybilguard-tr.pdf`.

[22] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman. Sybilguard: defending against sybil attacks via social networks. *IEEE/ACM Trans. Netw.*, 16(3):576–589, 2008.

A Formal Protocol Description

A.1 Communication Model and Party Identities

Our protocol involves several different classes of participants. There is a single *pollster*, and multiple *responders* and *verifiers* (the same physical party may participate in the protocol in more than one role).

Since the main problem we are trying to solve with this protocol is lack of identity verification, we cannot assume that the identities of all parties are known in advance. To simplify the analysis, we will only assume that the pollster's identity is publicly known.

Anonymous Channels and Network Addresses. To model the fact that the neither the pollster nor the honest parties know the identities of all the parties we will assume that all communication are over *anonymous* channels. To simplify analysis, we will only assume such channels between the pollster and every honest party (i.e., the communication graph is a star, with the pollster as the central node).

The pollster and all responders (modeled as ITMs) have one standard outgoing communication tape and one standard incoming communication tape. Every honest party has a unique *network address* in $\{0,1\}^*$ (the network address is given to the party as input). A message written to the outgoing communication tape by every party except for the pollster is copied to the pollster's incoming communication tape. An honest party will always write messages of the form $(addr, msg)$ to its outgoing communication tape, where $addr$ is its network address. Every message written by the pollster to its outgoing communication tape is also parsed as an $(addr, msg)$ tuple, and the message is copied to incoming communication tape of the honest party with network address $addr$ (if one exists). Regardless of whether $addr$ is valid or not, the message is also copied to the adversary's incoming communication tape.

This model allows two-way anonymous communication; Corrupt parties may write arbitrary source addresses (including using addresses allocated to the honest parties). Thus, the pollster cannot identify which party wrote the message (except by what is revealed from the contents of the message). Since a copy of every message is also sent to the adversary, sending a "fake" source address doesn't prevent a malicious responder from two-way communication with the pollster.

When we describe communication between two responder parties (e.g., when the protocol instructs responder A to communicate with responder B), this is shorthand for their communicating messages by passing them via the pollster. Although we omit this from the formal protocol description (to reduce clutter), this can be easily accomplished by adding a special header in the message that the honest pollster interprets as "forward to the specified address"; these messages are otherwise ignored by the honest pollster.

Broadcast Channel. In addition to point-to-point channels, we will assume a broadcast channel from the pollster to all the other parties. This is a special outgoing communication tape with corresponding incoming broadcast tapes for each other party; any message written by the pollster to the broadcast tape is copied to the incoming broadcast tapes of all parties. Verifiers have an incoming broadcast tape, but no standard incoming or outgoing tapes.

Complaint Channel. All parties have a special broadcast channel for "complaints". This is used only in cases where the pollster misbehaves in a way that cannot be publicly detected by other honest parties (for example, if the pollster refuses to interact with an honest party, or omits its inputs from the final proof).

Implementation with Peer-to-Peer Communication. Although our analysis assumes a star-shaped communication graph, this is not a very efficient strategy for networks that do

allow peer-to-peer communication (such as the Internet). Our protocol does not *require* any specific topology, as long as parties who need to can communicate—for example, the pollster can "introduce" every two parties who need to communicate (by giving them each other's network addresses), and let them communicate directly. This can only improve soundness, since a malicious pollster will lose access to the communication between honest parties, so our soundness guarantees will continue to hold (and it has no effect on completeness, since the pollster doesn't use its privileged position in the communication graph in any way except to pass messages).

A.2 Adversary and Corruption Model

We consider two types of adversaries: a malicious pollster (who attempts to violate the soundness guarantee) and malicious responders (who attempt to violate the completeness guarantee).

Soundness. In the malicious pollster case, our adversarial model does not include corruption of responders: this is without loss of generality, since the pollster can create an arbitrary number of "fake" responders and control them completely (honest responders do not have any secret inputs, so there is no information to be gained by corrupting them). Thus, modeling "sybils" is natural in this model (the pollster just broadcasts additional identities). When we refer to "corrupt" parties, this is shorthand for identities that were created by the pollster in this way.

We note that, since all communication is under the control of the pollster, the pollster *can* perform message replay attacks as well as copy message contents but forge the originator (as long as the fake source of the message is a corrupt party—it can't forge a signature for an honest party).

Completeness. In the malicious responder case, we can assume the pollster is honest (since a malicious pollster can always perform a denial-of-service attack by simply aborting). In this case, we can assume a single malicious responder, who creates an arbitrary number of fake identities (this is w.l.o.g. in the static corruption setting, since there are no secret inputs).

A.3 Full Protocol

We divide the protocol into six phases (as described in section 3). The formal protocol description for each phase appears below. The phases are executed in order. In addition to the specified inputs, each party receives as input its view from the previous phases.

A.3.1 Common Elements

We make use of several elements common to all of the subprotocols.

Signature Scheme. In all of the subprotocols, we make use of a public-key signature scheme, $(Gen, Sign, Ver)$. denoting the key generation, signing algorithm and verification algorithm respectively. $Gen(1^\kappa)$ outputs a key pair (sk, vk). $Sign(msg, sk)$ outputs a signature $\sigma \in \{0,1\}^*$, and $Ver(msg, \sigma, vk)$ outputs either 1 (if σ is a valid signature on msg with verification key vk) or 0 if not (if Ver outputs one we say the signature is *accepted*).

Pseudonyms and Network Addresses. The first step for every party in the protocol is generating a signature scheme key pair. The verification key is used as the party's *pseudonym*. Since parties communicate directly only through the pollster, we can identify the parties' network addresses with their pseudonyms (the pollster can keep a table mapping one to the other, or provide the address when needed). Note we do not assume a pre-existing PKI—a malicious

pollster can play man-in-the-middle between any pair of honest parties (however, the protocol will ensure that such an attack is not advantageous to the pollster).

Session identifiers, serial numbers and message validity. To prevent replay attacks, the pollster generates a unique session identifier sid as the first step of the protocol. A message between parties vk_{src} and vk_{dst} is considered valid only if it begins with the tuple $(sid, vk_{src}, vk_{dst}, serial, \sigma)$, where $serial$ is the number of messages sent so far in session sid between vk_{src} and vk_{dst}, and σ is a valid signature under vk_{src} of the entire message contents, including the initial header tuple $(sid, vk_{src}, vk_{dst}, serial)$.

Dealing with errors. At any point in the protocol, malicious parties may deviate from the honest execution. Any detected deviation is treated as if the deviating party aborted. For example, if party A receives an invalid message from party B, it treats this as if B aborted (and ignores any further messages from B).

Unless we explicitly say otherwise, the same holds for PPE verification (if parties A and B engage in a PPE execution, and party A did not successfully verify party B, it treats this as if B has aborted).

Protocol 1 Parameter Announcement

This protocol is executed by the pollster with inputs $\kappa, questions, p$:

1: $(SK, VK) \leftarrow Gen(1^\kappa)$ // Generate pollster's key pair
2: Broadcast $(sid, \kappa, p, VK, questions)$ to all parties.

Protocol 2 Registration

This protocol is between the pollster and every responder R_i. Every R_i receives as input $addr_i$ (its network address).

For every $i \in \{1, \ldots, m\}$, R_i and the pollster execute:

1: **(Responder R_i):** $(SK_i, VK_i) \leftarrow Gen(1^\kappa)$ // Generate responder's key pair
2: **(Responder R_i):** send $(addr_i, VK_i)$ to pollster.
3: **(Pollster and R_i):** Engage in a one-sided PPE with identifier (sid, VK_i) (in which R_i is the prover).
4: **(Pollster):** If verification was successful, add $(addr_i, VK_i)$ to the responder list.

When the registration phase is completed:

1: **(Pollster):** Choose a random permutation $\pi : [m] \mapsto [m]$ and broadcast the shuffled list $(VK_{\pi(1)}, \ldots, VK_{\pi m})$. // The shuffle is used only as defense against malicious responders; to reduce clutter we ignore it except in the completeness proof; when we write R_i below we actually mean $R_{\pi(i)}$.
2: **(Responder R_i):** Verify that VK_i appears in the list. If not, broadcast a complaint.

B Implementing PPEs, Extensions and Selective Polling

The peer-to-peer nature of PPEs seems to facilitate implementation with relatively simple mechanisms. Below we give several examples.

B.1 Bitcoin and Proofs of Storage

The original motivation for proofs of storage (PoS) is to allow clients to outsource data storage "to the cloud". In this setting a storage provider stores a large file on behalf of a client.

Protocol 3 Responder Certification

This protocol is executed between every responder R_i and its neighbors in the certification graph: $N_i = \{R_j | j \leq m, j \neq i, H(i,j) < p\}$. Note that N_i can be computed independently by R_i given the public parameters H, p and the list $(VK_1, \ldots, VK_m)$. For responder R_i:

1: **for all** $R_j \in N_i$ (concurrently) **do**
2: Engage in a PPE $\Pi^E(R_i, R_j)$ with identifier $(sid, VK_{\min\{i,j\}}, VK_{\max\{i,j\}})$.
3: If $\Pi^E(R_i, R_j) = 1$, R_i computes $\sigma \leftarrow Sign(sid||VK_j, SK_i)$ and sends σ to R_j.
4: Let σ_j be corresponding the signature received from R_j
5: **end for**
6: Let $Good_i = \{\sigma_j | Ver(sid||VK_i, \sigma_j, VK_j) = 1\}$ be the set good signatures from neighbors.

Protocol 4 Poll Response

This protocol is between is the pollster and every responder R_i. R_i's input in this phase is x_i, the answers to the poll questions.

For every $i \in \{1, \ldots, m\}$:

1: R_i sends $resp_i = (x_i, Good_i, Sign(sid||x_i||Good_i, SK_i))$ to the pollster.

Roughly, a PoS protocol allows the provider to prove to the client that it is still storing the file (can reconstruct the entire file), using a small amount of communication.

Since storage is a valuable resource, it is tempting to use proofs-of-storage as the "effort unit" in an effort-based polling scheme (e.g., one unit of effort could be storing 1GB of data for 1 day).

Moreover, publicly-verifiable proofs of storage have been constructed [1]—given a "public-key" generated for a specific file, anyone can verify that the PoS that the storage provider publishes for that file.

The problem here is that "backup" is a peer-to-peer concept. In particular, any solution must prevent malicious parties from sending each other "fake" data to store: e.g., they store a short seed instead of a large pseudorandom file generated by that seed.

However, the existing PoS protocols can be trivially used to construct a PPE: an honest user will send good (incompressible) files to its peers (e.g., by encrypting the file), and it can verify using the PoS that the files were stored as required. The soundness and completeness properties of this PPE are inherited directly from the PoS protocol, hence we can hope for almost perfect completeness and negligible soundness-error.

This implementation of PPEs may be most interesting in the context of Bitcoin. One of the strong arguments against the currency is the inherent waste of the Bitcoin protocol [11]; this is a direct consequence of using proof-of-work as the basis of its effort-based polling scheme. If we could replace proof-of-work with, for example, "proof-of-backup", instead of generating heat as a side-effect, the Bitcoin network would function as a distributed backup system in addition to a currency.

Protocol 5 Results and Proof

This protocol is executed by the pollster.

1: **for all** $i \in \{1, \ldots, m\}$ **do**
2: Broadcast $resp_i$
3: **end for**

Protocol 6 Verification

This protocol is executed by the responders and verifiers

Local Verification. Executed by every responder. For every $i \in \{1, \ldots, m\}$:

verify that $resp_i$ was published.
if $resp_i$ was not published, broadcast a complaint.

Global Verification. Executed by every verifier. The verifiers receive θ as a parameter.

Set $badNodes \leftarrow 0$.
Set $complaints \leftarrow 0$.
Initialize an array $count[]$ indexed by the poll answers.
for all $i \in \{1, \ldots, m\}$ **do**
    Verify that $resp_i$ contains a valid signature from under VK_i
    Verify that $Good_i$ is valid (all signatures in it are valid).
    if $|N_i| - |Good_i| > \eta_E d$ **then**
        Increment $count[x_i]$. // We don't count nodes if they failed more than $\eta_E d$ PPEs
    else
        Increment $badNodes$
    end if
    if R_i broadcast a complaint **then**
        Increment $complaints$
    end if
end for
if $complaints > \theta m$ **then**
    Output $\perp$ // Too many complaints
end if
if $badNodes > \eta_V m$ **then**
    Output $\perp$ // Too many deleted nodes
end if
Output the array $count$.

B.2 Human Interaction

The simplest type of PPE consists of human interaction: participants certify each other's effort by simply talking with each other (e.g., using VOIP, video, or even textual chat). This is at least as hard to pass than a "real" Turing test (which consists solely of textual interaction), so its soundness properties seem to be very robust.

To prevent a proxying attack (in which Eve convinces Alice that she has expended effort by relaying Alice's challenges to Bob and vice versa), the protocol can include Bob reading aloud his partner's identity. Thus, to act as a person-in-the-middle, Eve would have to translate Bob saying Eve's public key to Bob saying Alice's public key, which seems like it would require some actual effort.

B.3 Symmetric CAPTCHAs

In this version of the PPE, each party generates a "real" CAPTCHA to be solved by the other party while simultaneously solving the CAPTCHA she received.

To prevent a proxying attack, we bind the CAPTCHA to the parties' identities using a combination of cryptographic commitments and a Message Authentication Code (MAC). Define the CAPTCHA as a problem-generator $G(r)$ that given a random input r generates a CAPTCHA C along with its solution V.

1. When Bob generates a CAPTCHA for Alice, he chooses a secret MAC key and sets as the random input to G the MAC of the pair of public keys (Alice,Bob). He sends $G(r)$ to Alice.
2. Alice solves C, and sends a commitment to her solution to Bob.
3. Bob then sends his secret MAC key to Alice.
4. Alice verifies that the challenge she received is correctly generated (i.e., bound to Alice and Bob's public keys). If not, she aborts
5. Otherwise, Alice opens her commitment
6. Bob verifies that Alice correctly solved the challenge.

This protocol ensures that Eve can't use Alice to solve Bob's challenge to her, since Alice would refuse to open her commitment if she sees the CAPTCHA wasn't meant for her. Note that in terms of the effort required, this is not harder than just solving a CAPTCHA—the rest of the protocol can be completely automated.

B.4 Leveraging Existing Social Networks

Instead of an online effort, a possible PPE implementation can use an existing social network (basing the "effort" on the assumption that becoming "well connected" in a social network is difficult). For example, two parties can verify that they have several short, vertex-disjoint paths between them in the social network (or use some other measure of distance for which the effort assumption seems reasonable).

In this version of the protocol, parties are not guaranteed anonymity (since they must reveal their identities in order to verify their distance in the social network), but the public transcript of the protocol does not reveal anything about their identities or their social-network neighborhood.

The main problem here is preventing an adversary from using the same social-network identity in multiple different PPE invocations. The fact that the PPE is a private-coin primitive

makes this problem easy to solve, assuming the social network allows users to publish information linked to their real identity (e.g., a "home page"). Party i chooses a random nonce r_i and publish a commitment to r_i on their homepage. When executing the PPE with party j, i will publish r_i and privately open the commitment to r_i towards party j; Party j can verify by looking at i's homepage that the nonce is the correct one. Assuming the homepage provides a consistent view to all honest users, i cannot use a different nonce in different PPE invocations. However, the public transcript cannot be linked to i's social-network identity due to the hiding property of the commitment.

B.5 Other PPE Extensions

Our basic definition of PPE only guarantees that "effort" is expended by the parties. This can be easily extended to capture more complex conditions that are hard to verify publicly but may be easy to verify in a peer-to-peer manner. For example, we may want to poll participants only in a small geographic area. While certifying location in a publicly-verifiable way is difficult, verifying that someone else is physically nearby can be much easier (e.g., using speed of response or shared environmental cues, such as noise or micro-local weather conditions). By extending the PPE to also verify physical proximity, our protocol guarantees that if enough locals participate, the vast majority of participants must be local.

A similar situation occurs when we want to poll groups whose membership is secret (e.g., a poll of the "Anonymous" organization). If we can assume that members of the group can recognize each other (e.g., they have a "secret handshake"), we can use the same technique to guarantee that our poll is targeting the group.

We can also use an existing social network to poll specific communities (here the intent is to use the social network *in addition* to an effort test. Thus we can conduct verifiable polls on a social-network graph while keeping the graph itself secret—this can be important, since the structure of the social network often reveals a large amount of information about the identity of its nodes.

C Choosing Parameters

Below is a table containing a list of the most common parameters used throughout the paper. We partition the parameters into *fixed* parameters (in Table 1)—those that depend on assumptions about adversarial behavior and the effectiveness of the PPEs, and *tunable* parameters (in Table 2)—those that can be set by the poll designer (subject to certain constraints) and *computed* parameters (in Table 3)—these are functions of the previous parameters.

Table 1: Fixed Parameters

Symbol	Description
m	Total number of responders to the poll / Number of nodes in the graph.
n	Number of honest responders.
a	Upper bound on the number of oracle calls that the adversary can successfully perform / Upper bound on the number of attack edges.
	Continued on next page

Table 1 – continued from previous page

Symbol	Description
θ	Upper bound on the fraction of malicious responders: the total number of oracle calls made by malicious responders is at most θm.
σ	Probability of a PPE failing when both parties honestly follow the protocol.
ϵ	Probability of a PPE succeeding when one party does not make at least one oracle call.

Table 2: Tunable Parameters

Symbol	Description
κ	Security parameter.
d	Expected degree of the graph (expected number of PPE executions per responder). This can be tuned by changing p ($p = d/m$).
α	Minimum fraction of honest responders required to guarantee soundness.

Table 3: Computed Parameters

Symbol	Description
p	Edge probability. Every pair of responders will be required to engage in a PPE with probability p.
η_E	Upper bound on the fraction of PPE's that a responder can fail without getting deleted.
η_V	Upper bound on the fraction of nodes that can be deleted without causing the verification procedure to fail.
K	Number of nodes in the graph that the adversary can control "for free".

Continued on next page

Table 3 – continued from previous page

Symbol	Description
C^*	Upper bound on the multiplicative advantage of the adversary (an adversary has no more influence than an honest user that can invest C^* times the effort).

C.1 Constraints on Parameters

First, from Theorem 4.4 we have:

$$\sqrt{\frac{d(\frac{1}{2}-\eta_V)}{2(\ln m-1)}} > (\frac{1}{2}-\eta_V)/(\frac{1}{2}-\eta_V-\eta_E)$$

which implies that

$$d > \frac{\frac{1}{2}-\eta_V}{(\frac{1}{2}-\eta_V-\eta_E)^2}(2\ln m-2)\ . \tag{1}$$

By the definitions of α and K in Theorem 4.4, we get

$$\alpha \geq K/m+\eta_V = \frac{\kappa+2\ln m}{m}+\eta_V(lnm+2) \geq \frac{\kappa+2\ln m}{m} \tag{2}$$

Isolating η_V instead of α, we have:

$$\eta_V \leq \eta_V^{\max} = \frac{\alpha-\frac{\kappa+2\ln m}{m}}{2+\ln m} \tag{3}$$

Combining this with the bound on η_V from Theorem 5.2, we get

$$\eta_V^{\min} = \theta+\frac{3\cdot\max\left\{\frac{\kappa}{md},\theta\right\}}{\eta_E}+\frac{2\sigma}{\eta_E}\left(1+\max\left\{2,\frac{2\kappa}{md\sigma}\right\}\right) \leq \eta_V \leq \eta_V^{\max}$$

Which implies the following bound on η_E:

$$\eta_E \geq \eta_E^{\min} = \eta_E\cdot\frac{\eta_V^{\min}-\theta}{\eta_V^{\max}-\theta} = \frac{3\cdot\max\left\{\frac{\kappa}{md},\theta\right\}+2\sigma\left(1+\max\left\{2,\frac{2\kappa}{md\sigma}\right\}\right)}{\eta_V^{\max}-\theta} \tag{4}$$

Finally, note that we must have $\theta < \eta_V \leq \eta_V^{\max}$, but this is not sufficient. Since we need $\frac{1}{2}-\eta_V-\eta_E > 0$:

$$\begin{aligned}\frac{1}{2} &< \eta_V+\eta_E \geq \eta_V^{\min}+\eta_E^{\min}\\ &\geq \theta+\frac{3\theta+2\sigma\left(1+\max\left\{2,\frac{2\kappa}{md\sigma}\right\}\right)}{\eta_V^{\max}-\theta}\end{aligned}$$

Assuming $\theta < \frac{1}{2}\eta_V^{\max}$, this implies

$$\begin{aligned}\frac{1}{2} &> \theta+\frac{6\theta+4\sigma\left(1+\max\left\{2,\frac{2\kappa}{md\sigma}\right\}\right)}{\eta_V^{\max}}\\ &= \theta\left(1+\frac{6}{\eta_V^{\max}}\right)+\frac{4\sigma}{\eta_V^{\max}}\left(1+\max\left\{2,\frac{2\kappa}{md\sigma}\right\}\right)\end{aligned}$$

This gives us the following bound on θ:

$$\theta < \frac{\frac{1}{2} - \frac{4\sigma}{\eta_V^{\max}} \left(1 + \max\left\{2, \frac{2\kappa}{md\sigma}\right\}\right)}{1 + \frac{6}{\eta_V^{\max}}} \tag{5}$$

Since the θ must be non-negative, we also have a bound on σ:

$$\frac{12\sigma}{\eta_V^{\max}} \leq \frac{4\sigma}{\eta_V^{\max}} \left(1 + \max\left\{2, \frac{2\kappa}{md\sigma}\right\}\right) \leq \frac{1}{2}$$

hence

$$\sigma \leq \frac{\eta_V^{\max}}{24} \tag{6}$$

C.2 Examples of Parameter Settings

For simplicity we will consider PPEs for which the soundness error ϵ is negligible and we will omit it. Moreover, depending on the context in which we would like to use our protocol and the level of security we would like to achieve, different type of PPEs might be more suitable. As presented in Appendix B, there are multiple ways we could think of implementing PPEs and, naturally, each implementation comes with its own advantages/disadvantages. For instance, opting for a proof-of-storage based implementation can provide us with PPEs with almost perfect completeness ($\sigma = 0$), but requires a lot of communication. On the other hand, other implementations which would give us a worse completeness error (e.g., based on CAPTCHAs), might have higher error but require fewer (or different) resources.

In Table 4 the reader can find example parameter settings for two parameter regimes: in Scenarios 1 and 2, there are 5000 responders and PPEs are error-free, while Scenarios 3 and 4 have 100000 responders with PPEs that have a non-negligible (albeit small) error rate. The first scenario in each pair has degree close to the minimum possible for those parameters, while the second demonstrates the soundness advantage of increasing the degree (we note that the values are based on our worst-case bounds—in practice it may be possible to achieve better parameters).

Table 4: Possible Parameters

Symbol	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
κ	40	40	40	40
m	$5,000$	$5,000$	$100,000$	$100,000$
θ	1/1000	1/1000	1/10000	1/10000
σ	0	0	1/1000	1/1000
η_E	1/8	1/8	0.23	0.23
η_V	0.025	0.025	0.028	0.028
α	0.28	0.28	0.38	0.38

Continued on next page

Table 4 – continued from previous page

Symbol	Scenario 1	Scenario 2	Scenario 3	Scenario 4
K	1246	1246	$35,192$	$35,192$
d	60	120	165	240
C^*	200	10	670	23

As to be expected, higher the degree of the graph (that is the number of PPEs each responder is required to carry out) lower is the advantage the adversary gets.