Public Verification of Private Effort

Giulia ALBERINI

School of Computer Science

McGill University, Montreal

December 2015

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Ph.D in Computer Science

©Giulia Alberini, 2015

"The isolated man does not develop any intellectual power. It is necessary for him to be immersed in an environment of other men, whose techniques he absorbs during the first twenty years of his life. He may then perhaps do a little research of his own and make a very few discoveries which are passed on to other men. From this point of view the search for new techniques must be regarded as carried out by the human community as a whole, rather than by individuals."

A. TURING

To my parents,
who shaped my mind.

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 previous 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 associated to the responders' participating to the poll, and whose edges cross-certify that adjacent nodes correspond to honest participants. Cross-verification 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 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.

Résumé

Nous introduisons un nouveau cadre formel afin de scruter les réponses d'une grande population de votants. Notre cadre permet la collecte des votes tout en garantissant l'anonymat des participants ainsi que la possibilité de vérifier publiquement le résultat du scrutin. Contrairement aux approches précédentes à ce sujet, nous n'exigeons pas que le scrutateur soit digne de confiance lors de l'annonce des résultats, ni ne nous reposons sur une vérification "forte" de l'identité des participants.

Nous proposons un protocole de scrutin "basé sur l'effort" dont les résultats du vote peuvent être vérifié publiquement via la construction d'un "graphe de certification des votants" dont les sommets sont étiquetés par diverses informations liées à chaque votant et dont les arêtes servent à la certification de l'honnêteté mutuelle de certaines paires de votants. La certification de l'honnêteté mutuelle est obtenu par l'elaboration du concept de "preuves d'effort privées" (PEP) vérifiables en privé. Dans les faits, de notre protocole se dégage un méthode générale permettants de convertir des preuves vérifiables en privé en preuves vérifiables en public. La justesse de la transformation repose sur les propriétés d'expansion du graphe de certification.

Nos résultats sont applicables dans divers scenarios dans lesquels la collecte collaborative d'information de masse est nécessaire. Ceux-ci incluent les crypto-monnaies, les sondages préélectoraux, les élections, les systèmes de recommendation, les votes télévisuels et les sondages d'opinions.

Acknowledgements

When we achieve a goal in life we never do it alone. I would like to start by thanking my supervisor, Claude Crépeau, for accepting me as one of his students when I had no background on cryptography. Thank you for your constant support and all the needed advice you gave me during the writing of this thesis. A warm thank you goes to Gilles and Louis for keeping me afloat throughout the years. It was very much appreciated. A very special thank you goes to Alon. I'm not sure how many students get a research assistantship offer without asking and from someone who doesn't even really know you. I feel incredibly lucky for the opportunity you gave me. Through my experience at IDC I had the chance to be part of a challenging, vibrant, and exciting research group. Thank you for your endless support and encouragement. You believe in me more than I do, and I owe you my success. To Tal, thank you for involving me in the project that eventually became this thesis. Thank you for all the insightful discussions and for being available to answer my questions at any time during the night. A sincere thank you to Diti for dealing with all my questions and my aversion to deadlines and bureaucracy.

I would like to thank my sister Alice and my cousin Lorenza for being here in Montreal, and for making me feel at home every day. Thank you to Sanni for putting up with all my ranting, for always providing invaluable advice, and for making any night better with some good beers and great company. A huge thank you goes to Linda. Without her I would not know what the state of my mental health would have been. Thank you for the immense support you have been able to provide me even from across the ocean.

Finally, *grazie a mamma e papà* for being open minded and unconventional parents. Thank you for allowing me to get through thirteen years of university without any debts. And a final special thank you goes to my significant other Billy, for his endless patience in dealing with me (especially when I'm hungry). Thank you for always finding a way to make me smile and for helping me find the serenity I really needed.

Contributions of Authors

Most of the work contained in this thesis appears in a paper [2] accepted to TCC (Theory of Cryptography Conference) 2015. This is a joint work with Tal Moran and Alon Rosen to which we all equally contributed. The project began in 2012 when I was working as a research assistant at IDC Herzliya under the supervision of Alon Rosen.

Contents

Chapter 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 [45]).

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 popularized "cryptocurrency", is in achieving a distributed, decentralized consensus about the currency's public transaction ledger (the record of all Bitcoin transactions) [45].

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 section 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 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" 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: 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 [35]). 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 [57] 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*: every CAPTCHA is generated together with its solution.)

Beyond being easier to achieve, 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

[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 by that candidate. However, this still fits in our generalized polling framework.

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.

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 [12] and define effort in terms of oracle calls; see chapter 3 for details).

The peer-to-peer nature of PPEs makes them potentially easier to achieve than their publicly-verifiable counterparts (which require costly distributed coordination). In section 3.3, 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.

1.2 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. 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, every party controlled by the adversary is considered "cheating", even if it follows the honest protocol exactly).

If each node of the graph is published together with its 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 the indices of two nodes to determine whether 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 in the certification graph: this is the number of PPE executions in which each responder is expected to participate.

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 without expending any effort.

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 $\mathcal{A}$ 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 $\mathcal{A}$ controls more than $\Omega\left(\frac{a}{d}\right)$ of the responses in the poll outcome, then verification will fail with overwhelming probability (in κ).*

See Section 4.3 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 4.4 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

Verifiable voting systems use cryptographic techniques to assure election integrity from voter intent to final tally. These systems are characterized by the following two properties: voter auditing and universal verifiability. The former enables any participant to verify whether his or her vote was correctly included in the electronic ballot box. The latter allows anyone to verify if all the votes have been actually counted. In 2004, David Chaum proposed a first verifiable voting system in [14]. Since then, many solutions have been presented [15], [1], [51].

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 (modelled 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. Unlike most voting protocols, not only interaction with the pollster is needed, but also interaction with a subset of other responders is included.

   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* [13])—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 [23]. 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 [20]. The paper by Ellison et al. [23] deals with the problem of establishing identities. One of the first discussions of trust metrics based on social graphs appears in [37]. The term "Sybil Attack" (attributed to Brian Zill from MSR) was introduced

in [20], 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 [34], and the possibility of using "Turing tests" to limit Sybil nodes is mentioned in [6]. In [16] 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.

Moderately hard problems (cryptographic problems which are not computationally infeasible to solve, but also not easy) have been used to limit an adversary's abuse of resources such an spam [21] and denial of services [7], [32]. Aspenes et al. [4] adapt techniques which use moderately hard problems to implement algorithms to solve the distributed consensus problem. In their paper they present two algorithms, *Democracy* and *Monarchy*, that are meant to be run before the actual protocol in order to price each identity based on their computational power. Every participant is required to interact with all the others, and they assume the list of participant is agreed-upon by all the nodes before the beginning of the validation protocol. After such protocol, the consensus algorithm can be safely run based on the assumption that the adversary controls a limited fraction of the total computational power.

The technique of random walks on a social networks to bound the effect of Sybil attack is introduced in [62] (see [63] for an expanded version with full proofs). A 2006 survey of Sybil attack literature can be found in [38]. An improved version of [62] (slightly different protocol, same goal but better parameters) appears in [61], and a newer protocol to identify Sybil nodes in a social graph is presented in [18]. The protocol makes very similar assumptions about the social graph, and Bayesian methods to compute the probabilities that nodes are Sybils. Finally, [56] uses the social network graph to aggregate votes for online content.

Most of these techniques implicitly or explicitly use assumptions about expansion properties of social-network graphs. Hence if "adding edges is hard" (i.e. if successfully performing an "action" that will result in adding an edge in the corresponding graph is hard) in an expander

graph, the adversary is limited in the effect bad nodes can have. However, 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 labelling 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 [58]. The first suggestion for cryptographic verifiability in voting, which also gives a mechanism for establishing anonymous channels (mix-nets) was made in [13]. More recently, the works of [14, 50] 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 [44].

1.6 Structure of this thesis

In Chapter 2 we introduce the mathematical and cryptographical notions that will be used throughout the thesis. In Chapter 3 we describe the model we will be using, and we introduce the notion of Privately verifiable Proof of Effort (PPE). In Chapter 4 we define our polling protocol and present the main proofs. Finally, the conclusion, together with a brief summary and a discussion on future work directions are given in Chapter 5.

Chapter 2

Preliminaries

This section presents the definitions that we need in the thesis.

2.1 Mathematical Background

2.1.1 Basic Probability

Given a random variable X, we denote by $\mathrm{E}[X]$ the expected value of X. Given an event a, X is said to be the indicator random variable for a if

$$X = \begin{cases} 1, & \text{if } a \text{ occurs} \\ 0, & \text{otherwise.} \end{cases}$$

Note that if $\Pr[a] = p$, then we have that $\mathrm{E}[X] = p$.

Union Bound. If we have a finite or countable set of events $\{a_i\}$, then we might be interested in the probability that one of those events occurs, that is the probability of the event $\bigcup_i a_i$. The *union bound* tells us that the probability for at least one of the events to happen is no greater than the sum of the probabilities of each individual event. That is, for any countable set of events $\{a_i\}$ the following holds

$$\Pr\left[\bigcup_i a_i\right] \leq \sum_i \Pr\left[a_i\right]$$

Chernoff Bound. Let X be a sum of n *independent* random variables $\{X_i\}$, with $\mathrm{E}[X_i] = p_i$. Let μ denote the expected value of X, that is

$$\mu = \mathrm{E}\left[\sum_i X_i\right] = \sum_i \mathrm{E}[X_i] = \sum_i p_i$$

The Chernoff bound intuitively states that the probability that the random variable X lies far from its mean is bounded by an exponential function of its variance.

Theorem 2.1. (Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent random variables with $0 \leq X_i \leq 1$, $X = \sum_i X_i$, and $\mu = \mathrm{E}[X]$. Then for any $\delta > 0$ the following inequalities hold:*

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2}{2+\delta}\,\mu\right)$$
$$\Pr[X \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2}{2}\,\mu\right)$$
$$\Pr[|X-\mu| \geq \delta\mu] \leq 2\exp\left(-\frac{\delta^2}{2+\delta}\,\mu\right)$$

2.1.2 Random Graphs and Expansion

Random graphs are one of the most important concepts in Combinatorics and Theoretical Computer Science. Throughout this thesis they serve as essential tools in proving the main combinatorial statements. In this section, we review all the necessary definitions and results on random graphs. For more details, we refer the reader to two books [10], [31] devoted entirely to this subject.

The theory of random graphs began in the late 1950's with a series of seminal papers by Paul Erdős and Alfréd Rényi [24], [25], [26]. However, only much later, research in this field took off, starting with the works of Bender and Canfield [8], Bollobás [9], and Wormals [60]. Finally, the works by Watts and Strogatz [59] (introducing the small-world model) and by Barabási and

Albert [3] (on the preferential attachment model) underlined how random graphs differ from real-world networks and led to a rapid increase of research in the field.

In this thesis we focus on the random graph model introduced by Erdős and Rényi. We denote by $G(n,p)$ the probability space of all labelled undirected graphs of n vertices $\{1,\dots,n\}$ where for each pair of vertices $i,j \in \{1,\dots,n\}$, $i \neq j$, the corresponding edge (i,j) exists with probability p, independently of any other edge. If $G=(V,E)$ is generated using $G(n,p)$, we can compute the expected degree of one of its nodes i by using basic probability properties. Let X_i be the random variable denoting the degree of $i \in V$, and let $X_{i,j}$ be the indicator random variable for the event "$(i,j) \in E$". Since $G \in G(n,p)$, the $X_{i,j}$'s are independent and $\Pr[X_{i,j}=1]=p$. Then,

$$\begin{aligned} X_i &= \sum_j X_{i,j} \\ \mathrm{E}[X_i] &= \sum_j \mathrm{E}[X_{i,j}] \\ &= \sum_j p = (n-1)p. \end{aligned}$$

We can think of p as $\frac{d}{n-1}$ for some d, then the expected degree of $i \in V$ is d.

For our purposes, the most important parameter of a random graph is its edge expansion. Let $S \subseteq V$ be a set of nodes, $|S|$ denote its size, $\overline{S}$ its complement, and $cut(S)$ denote the number of edges with endpoints in each of S and $\overline{S}$. The edge expansion of a graph is defined as follows,

Definition 2.2. (Edge Expansion) Let $G=(V,E)$ be a graph of n vertices. We define the *edge expansion* coefficient e_G as

$$e_G = \min_{|S| \leq n/2} \frac{cut(S)}{|S|}.$$

For any $\alpha \in (0, 1/2]$, we define $e_G(\alpha)$ as

$$e_G(\alpha) = \min_{\alpha n \leq |S| \leq (1-\alpha)n} \frac{cut(S)}{|S|}.$$

The notion of Edge Expansion give rise to a specific notion of expander graphs. Informally, a good expander graph is a graph with low degree, but high expansion factor. Expander graphs and their constructions have been thoroughly studied ([40], [41], [42], [43], [52]). It is not straightforward if and which construction could be applicable to our protocol. In fact, in standard expander graphs, their expansion property is defined between any subset of vertices and their complement. In Chapter 4, we define an expansion property that looks at the boundary between a "big enough" set and any large subset of its complement. This property, which we call *Large-Set Expanding* (LSE) property, is enough to ensure the security of our protocol.

2.2 Cryptographic background

2.2.1 Negligible Success Probability

In cryptography, we consider a protocol to be secure if the probability that it can be broken is asymptotically smaller than $1/p(n)$ for all polynomials p. Throughout this thesis when referring to negligible functions, denoted by negl, we will be referring to a function f such than for every polynomial $p(\cdot)$ there exists an N such that for all $n > N$ it holds that $f(n) < 1/p(n)$.

2.2.2 Message Authentication Codes

Message Authentication Codes (MAC) are used in order to guarantee message integrity: each party should be able to verify whether the message it has received was indeed sent by a specific party. Thus, the aim of a MAC is to prevent an adversary from modifying a message without the parties detecting that a modification has been made. Two parties wishing to communicate using a MAC begin by generating and sharing a secret key k. When a party wants to send a message M, he first computes a *tag* τ using k and M and he sends the pair (M, τ). The second party, after receiving (M, τ) then verifies, using the key k, whether τ is a valid tag for the message M. Formally, we can define a MAC as follows:

Definition 2.3. A *Message Authentication Code* (or *MAC*) consists of three polynomial-time algorithms $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ defined as follows:

- The randomized `key-generation algorithm` Gen on input a security parameter 1^κ returns a secret key k.
- The probabilistic `tag-generation algorithm` Mac takes as inputs a message M, a key k, and returns a tag τ. We denote this by $\tau \leftarrow \mathrm{Mac}_k(M)$
- The deterministic `verification algorithm` Vrfy takes as inputs the pair (M, τ), a key k, and returns a bit b. If $b = 1$ the tag τ is considered to be valid, otherwise is said to be invalid. We write $b := \mathrm{Vrfy}_k(M, \tau)$.

It is required that for any key k output by Gen, and every message M, it holds that

$$\mathrm{Vrfy}_k(M, \mathrm{Mac}_k(M)) = 1$$

Security of message authentication codes. An adversary $\mathcal{A}$ whose goal is to break a MAC should be able to generate a valid tag for any message that was not previously authenticated by a sender S. We want for a security definition to take into account the possibility that the adversary $\mathcal{A}$ has access to many messages (even of its choice) and their corresponding MAC tags. We model this by giving the adversary access to a *MAC oracle* $\mathrm{Mac}_k(\cdot)$. After requesting as many tags as it desires, $\mathcal{A}$ has to output a pair (M, τ). We say that $\mathcal{A}$ broke the code if τ is a valid tag for M, and $\mathcal{A}$ did not query the oracle with M. Formally,

Definition 2.4. Let $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a message authentication code, and $\mathcal{A}$ an adversary. Then the *authentication experiment* $\text{Mac-Exp}_{\Pi,\mathcal{A}}(\kappa)$ is as follows:

1. Gen is run on input 1^κ to obtain a key k.
2. The adversary $\mathcal{A}$ is given oracle access to $\mathrm{Mac}_k(\cdot)$. Let Q denote the set of queries of $\mathcal{A}$ to such oracle. The adversary then outputs (M, τ).
3. The experiment outputs 1 if and only if $\mathrm{Vrfy}_k(M, \tau) = 1$ AND $M \notin Q$.

The *advantage* of $\mathcal{A}$ is defined by,

$$\mathrm{Adv}_{\Pi,\mathcal{A}}(\kappa) = \Pr[\text{Mac-Exp}_{\Pi,\mathcal{A}}(\kappa) = 1]$$

We can now formally define a secure MAC if no efficient adversary can succeed the above experiment, that is

Definition 2.5. A Message Authentication Code $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ is said to be *existentially unforgeable under an adaptive chosen-message attack* if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible. That is, there exists a negligible function negl such that

$$\Pr[\text{Mac-Exp}_{\Pi,\mathcal{A}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$$

Note that this definition, and message authentication codes in general, do not protect against *replay attacks*: it is always possible for $\mathcal{A}$ to just replay a message that was previously sent with its corresponding tag. Such protection is left to higher-level application and it usually involves using either sequence-numbers or time-stamps. That is, when sending a message M the sender will have to assign to M either a sequence-number or the time in which the message is sent and compute the tag over the concatenation of the message M and its unique identifier.

2.2.3 Digital Signature Schemes

Consider a sender S who has generated a pair (sk, vk) of secret and a public key, respectively. A digital signature scheme allows S to "sign" its messages in such a way that anyone who also knows vk can verify that the message sent by S has not been modified in any way. Digital signatures are the public-key counterpart of message authentication. An essential difference lies in the fact that signatures are *publicly verifiable*. This implies that a party receiving a signed message can be sure that if the verification succeed on his side, then such a signature will be considered valid by any other party.

Definition 2.6. A *signature scheme* consists of three polynomial-time algorithms $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ defined as follows:

- The randomized `key-generation algorithm` Gen on input the security parameter 1^κ returns a pair (sk, vk) of keys. These are called the *secret key* and the matching *verification key*, respectively.

- The probabilistic `signing algorithm` Sign takes the secret key sk and a message M as inputs and returns a signature σ. We denote this by $\sigma \leftarrow \mathrm{Sign}_{sk}(M)$.
- The deterministic `verification algorithm` Vrfy takes the verification key sk, a message M, and a candidate signature σ as inputs and returns a bit b. If $b = 1$, the signature σ is considered valid, otherwise it is said to be invalid. We write $b := \mathrm{Vrfy}_{vk}(M, \sigma)$.

It is required that for any key-pair (sk, vk) output by Gen, and every message M, it holds that

$$\mathrm{Vrfy}_{vk}(M, \mathrm{Sign}_{sk}(M)) = 1$$

A sender S can then use the signature scheme by running Gen in order to obtain a pair (sk, vk) of keys. S publicizes vk and whenever it wants to transmit a message M, it can compute the signature $\sigma \leftarrow \mathrm{Sign}_{sk}(M)$ and send the pair (M, σ). Any receiver who has access to vk can verify the authenticity of M by running the verification algorithm and check whether $\mathrm{Vrfy}_{vk}(M, \sigma) = 1$.

Security of signature schemes. The goal of an adversary $\mathcal{A}$ is *forgery*: given a public key vk generated by S, we say that $\mathcal{A}$ successfully forged S's signature if it can produce a pair (M, σ) such that $\mathrm{Vrfy}_{vk}(M, \sigma) = 1$ and M was not previously signed by S. We want a notion of security that ensures us that an adversary $\mathcal{A}$ is not be able to output a forgery even if it can obtain signatures on many other messages of its choice. To formalize the measure of security we define an experiment in which the adversary's actions are viewed as divided into two phases. The first is a "learning" phase in which the adversary is given oracle access to the signing algorithm. Once this phase is over, the adversary enters the "forgery" phase in which it outputs a pair (M, σ). The adversary is successful if $\mathrm{Vrfy}_{vk}(M, \sigma) = 1$ and $\mathcal{A}$ did not query the oracle with M. We call $\mathcal{A}$'s probability of success its advantage.

Definition 2.7. Let $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ be a signature scheme and $\mathcal{A}$ be an adversary. Then the *signature experiment* $\text{Sign-Exp}_{\Pi,\mathcal{A}}(\kappa)$ is as follows:

1. Gen is ran on input 1^{κ} in order to obtain (sk, vk).

2. The adversary $\mathcal{A}$ is given oracle access to $\mathrm{Sign}_{sk}(\cdot)$. Let Q denote the set of queries of $\mathcal{A}$ to such oracle. The adversary then outputs (M, σ).

3. The experiment outputs 1 if and only if $\mathrm{Vrfy}_{vk}(M, \sigma) = 1$ AND $M \notin Q$.

The *advantage* of $\mathcal{A}$ is defined by,

$$\mathrm{Adv}_{\Pi,\mathcal{A}}(\kappa) = \Pr\left[\text{Sign-Exp}_{\Pi,\mathcal{A}}(\kappa) = 1\right]$$

We can now give a formal definition of security for a signature scheme.

Definition 2.8. A signature scheme $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ is said to be *existentially unforgeable under an adaptive chosen-message attack* if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible. That is, there exists a negligible function negl such that

$$\Pr\left[\text{Sign-Exp}_{\Pi,\mathcal{A}}(\kappa) = 1\right] \leq \mathrm{negl}(\kappa)$$

2.2.4 Bit Commitment Schemes

A bit commitment scheme is a two-stage (interactive) protocol between a sender S and a receiver R. Informally, after the *commit stage*, the sender is committed to (at most) one value, usually a single bit b, in such a way that the receiver has no idea what b is. After the *unveil stage*, the receiver learns the value that was concealed in the previous stage. One could think about it as if the sender puts b in a safe of which he is the only owner of the key, and gives the safe to the receiver (commit stage). Later on, it would be enough to give the key of the safe to the receiver so that he can learn what was the value inside the safe (unveil stage). The two security properties that we can draw out are referred to as *binding* (after the commit stage, the sender is bound to at most one value) and *concealing* (the receiver cannot learn the value in the safe before the unveil stage).

As with most cryptographic primitives, these properties come in two main flavours: *computational* (the property holds only against a computationally bounded, for instance polynomial-time, party) and *statistical* (the property holds even against an all-powerful party).

Commitments are basic components of many cryptographic protocols: they are used as important building blocks in zero-knowledge proofs, coin-flipping, identification schemes, and multi-party computation. Statistically binding bit commitments have been well understood since a long time and they can be obtained from the minimal assumption that one-way functions exist using the results in [46] combined with the fact that we can construct pseudorandom generators from any one-way function [30]. On the other hand, *perfectly*[1] concealing commitment schemes were first shown to exist based on specific algebraic assumptions by Brassard, Chaum, and Crépeau [11] and later Naor, Ostrovsky, Venkatesan, and Yung [48] showed how to construct perfectly concealing commitment schemes based on the assumption that one-way permutations exist. In [27], Haitner, Horvitz, Katz, Koo, Morselli, and Shaltiel reduced the assumptions further, and finally Haitner, Nguyen, Ong, Reingold, and Vadhan in [28] and, later Haitner, Reingold, Vadhan, and Wee with a simpler proof, in [29] gave a construction of statistically concealing commitment schemes under the minimal complexity assumption that one-way functions exist.

Definition 2.9. A (bit) commitment scheme $(\mathcal{S}, \mathcal{R})$ is an efficient two-party protocol consisting of two stages. Throughout, both parties receive the security parameter 1^κ as input.

COMMIT. The sender $\mathcal{S}$ has a private input $b \in \{0,1\}$, which he wishes to commit to the receiver $\mathcal{R}$, and a sequence of coin tosses s. In this stage, $\mathcal{S}$ and $\mathcal{R}$ are allowed to interact and at the end of it, both parties receive as common output a commitment z.

UNVEIL. Both parties receive as input a commitment z. $\mathcal{S}$ also receives the private input b and coin tosses s used in the commit stage. This stage is non-interactive: $\mathcal{S}$ sends a single message to $\mathcal{R}$, and $\mathcal{R}$ either outputs a bit (and accepts) or rejects.

Definition 2.10. A commitment scheme $(\mathcal{S}, \mathcal{R})$ is computationally binding and statistically concealing if

COMPLETENESS. When both parties are honest, then for any bit $b \in \{0,1\}$ that $\mathcal{S}$ gets as private input, $\mathcal{R}$ accepts and outputs b, at the end of the unveil stage, with probability 1.

[1] Not even a negligible probability of breaking the scheme is allowed.

STATISTICALLY CONCEALING. For any $\mathcal{R}^*$, the distributions $\text{view}_{\mathcal{R}^*}(\mathcal{S}(0), \mathcal{R}^*)$ and $\text{view}_{\mathcal{R}^*}(\mathcal{S}(1), \mathcal{R}^*)$ are statistically indistinguishable, i.e. the probability that any machine can distinguish between the two views is negligible in κ.

COMPUTATIONALLY BINDING. For every PPT (Probabilistic Polynomial-Time) $\mathcal{S}^*$, $\mathcal{S}^*$ succeeds in the following "game" with negligible probability in κ:

- $\mathcal{S}^*$ interacts with an honest $\mathcal{R}$ in the commit stage, which yields a commitment z.
- $\mathcal{S}^*$ can output one of the two messages x_0, x_1 at will, such that for both $b = 0$ and $b = 1$, $\mathcal{R}$ on input (z, x_b) accepts and outputs b.

Definition 2.11. A commitment scheme $(\mathcal{S}, \mathcal{R})$ is statistically binding and computationally concealing if

COMPLETENESS. When both parties are honest, then for any bit $b \in \{0, 1\}$ that $\mathcal{S}$ gets as private input, $\mathcal{R}$ accepts and outputs b, at the end of the unveil stage, with probability 1.

COMPUTATIONALLY CONCEALING. For every PPT $\mathcal{R}^*$, $\mathcal{R}^*$ succeeds in distinguishing between $\text{view}_{\mathcal{R}^*}(\mathcal{S}(0), \mathcal{R}^*)$ and $\text{view}_{\mathcal{R}^*}(\mathcal{S}(1), \mathcal{R}^*)$ with negligible probability in κ.

STATISTICALLY BINDING. For any $\mathcal{S}^*$, $\mathcal{S}^*$ succeeds in the following "game" with negligible probability in κ:

- $\mathcal{S}^*$ interacts with an honest $\mathcal{R}$ in the commit stage, which yields a commitment z.
- $\mathcal{S}^*$ can output one of the two messages x_0, x_1 at will, such that for both $b = 0$ and $b = 1$, $\mathcal{R}$ on input (z, x_b) accepts and outputs b.

Note that $\text{view}_{\mathcal{R}^*}(\mathcal{S}(i), \mathcal{R}^*)$ we denotes "everything $\mathcal{R}^*$ sees" in the interaction with $\mathcal{S}$ during the commit phase, when $\mathcal{S}$ has as private input i. In general, let ρ be the string contained in the random tape of $\mathcal{R}^*$, and say that the interaction between $\mathcal{S}$ and $\mathcal{R}^*$ consist of ℓ turns where s_i and r_i are the ith messages of $\mathcal{S}$ and $\mathcal{R}^*$ respectively. Then, we say that $(\rho, r_1, s_1, \ldots r_\ell, s_\ell)$ is the *view* of $\mathcal{R}^*$. Note that in the definition of the concealing property, we do not include the unveiling phase when referring to the *view* of $\mathcal{R}^*$.

2.3 CAPTCHAS

CAPTCHA is an acronym for *Completely Automated Public Turing test to tell Computers and Humans Apart*. These are tests that are easy for humans and hard to solve for computer programs. This kind of "proofs of being a human" were first introduced in [39], independently discovered by [47], and formalized by [57]. Many type of CAPTCHAs have been proposed, but the most common ones in use are distorted images which are hard to read for a computer, but still recognizable by humans (see Figure 2.1). Applications include preventing automated programs from participating to online polls, using free email services, or carrying dictionary attacks.

FIGURE 2.1: A CAPTCHA from Google

The definition of CAPTCHA is based on some limitation of the power of computers. The security of such tests relies on the fact that no automated machine can solve the test efficiently. From a cryptographic point of view, CAPTCHAs bring up two interesting questions: which "hard problems" can we use to build CAPTCHAs?, and can we use CAPTCHAs as building blocks for achieving general cryptographic tasks? The first question is analyzed in [57] were the authors provide constructions of CAPTCHAs based on the assumption that certain Artificial Intelligence problems are hard to solve. The second problem has been investigated in many different papers in the last few years: Canetti, Halevi, and Steiner construct a scheme to mitigate dictionary attacks in [12], Diaz-Santiago and Chakraborty built an encryption protocol using CAPTCHA in [19], Dziembowski introduces a protocol for *Human Key Agreement* based only on CAPTCHAs in [22], and Kumarasubramanian, Ostrovsky, Pandey, and Wadia construct a CAPTCHA-based commitment scheme in [36].

CAPTCHAs have been defined in different ways in the literature. Here we want to present a definition that follows the Canetti et al.'s framework [12] that models the presence of a human

entity as a *human oracle* H capable of solving CAPTCHAs. Informally, a party generates a CAPTCHA by running a "generation algorithm" which outputs a pair (t, s) consisting of a test t and its solution s. Then, to solve a CAPTCHA, each party can query the oracle H with the test t and obtain the solution s as an answer. There will then be standard PPT machines for which solving the test is a hard problem, and machines with oracle access to H which will be able to solve CAPTCHAs efficiently. We denote by M^H a machine M with oracle access to H. Let $\mathrm{Gen}(1^\kappa)$ denote the randomized generation algorithm which on input a security parameter κ outputs a pair (t, s), with $t \in \mathcal{T}_\kappa$, and $s \in \mathcal{S}_\kappa$. We call $\mathcal{T}_\kappa$ the *test space*, and $\mathcal{S}_\kappa$ the *solution space*. We denote by $H_\kappa : \mathcal{T}_\kappa \to \mathcal{S}_\kappa$ the *solution function* which maps tests to their corresponding solutions. Then, to model the idea of "human effort", machines will be given oracle access to the following family of functions $H = \{H_\kappa\}_{\kappa\in\mathbb{N}}$. Formally,

Definition 2.12. (**CAPTCHA**) Let $\mathcal{T} = \{\mathcal{T}_\kappa\}_{\kappa\in\mathbb{N}}$, and $\mathcal{S} = \{\mathcal{S}_\kappa\}_{\kappa\in\mathbb{N}}$ be a collection of test and solution spaces respectively. A CAPTCHA $\mathcal{C} = (\mathrm{Gen}, H)$ over $(\mathcal{T}, \mathcal{S})$ consists of the following pair:

- A `randomized generation algorithm` Gen that on input 1^κ outputs a pair $(t, s) \in \mathcal{T}_\kappa \times \mathcal{S}_\kappa$.
- A `collection of solution functions` $H = \{H_\kappa\}_{\kappa\in\mathbb{N}}$ with $H_\kappa : \mathcal{T}_\kappa \to \mathcal{S}_\kappa$.

It is required that for any pair (t, s) output by Gen on input 1^κ, it holds that $H_\kappa(t) = s$. Furthermore, the following condition is required:

- For every probabilistic polynomial-time $\mathcal{A}$, and every sufficiently large κ there exists a negligible function negl such that

$$\Pr[(t, s) \leftarrow \mathrm{Gen}(1^\kappa), \mathcal{A}(1^\kappa, t) = s] \leq \mathrm{negl}(\kappa)$$

2.4 Proofs of Retrievability

Consider the case in which a client decides to use outsourced storage services to store its data. One of the challenges raised by this scenario is to verify whether the server storing the data

keeps it available and ready for retrieval. A PoR (Proof of Retrievability) is an interactive protocol that enables a verifier $\mathcal{V}$ (the client or whoever else if the PoR is publicly verifiable), using a small amount of communication, to determine if it is possible to retrieve a specific file M from the prover $\mathcal{P}$ (the server). PoR's can be constructed to be either privately-verifiable where only the client who originally stored the data can verify whether is it still available, or publicly-verifiable where anyone can undertake the role of a verifier in the PoR. The first to define a formal model for proofs of storage (retrievability) were Naor and Rothblum [49], and Juels and Kaliski [33]. The first to propose a publicly-verifiable proofs of storage were Ateniese et al. [5]. We will use the formal model for PoR introduced by Shacham and Waters [53].

Definition 2.13. A *proof of retrievability* protocol consists of four algorithms $(\mathrm{Gen}, \mathrm{Store}, \mathcal{P}, \mathcal{V})$ defined as follows:

- The randomized `key-generation algorithm` Gen on input a security parameter 1^κ returns a public-key/private-key pair (pk, sk).
- The randomized `storing algorithm` Store takes the secret key sk and a file $M \in \{0,1\}^*$ as input and returns a pair (M', τ) consisting of the file M' that will be stored by the server and a corresponding tag τ.
- The randomized `proving` and `verifying algorithms` $\mathcal{P}$ and $\mathcal{V}$ which both take as input the public key pk and the tag τ. The prover also takes M' as input, while the verifier takes the secret key sk. The two algorithms interact and at the end of the protocol, $\mathcal{V}$ outputs 1 if the file M is stored on the server and 0 otherwise.

Security of PoRs. We say that a proof of retrievability is *correct* if for all (pk, sk) generated by Gen, for all $M \in \{0,1\}^*$, and for all (M', τ) output by Store on input (sk, M), the verification algorithm accepts when interacting with a valid prover. On the other hand, we say that a PoR is *sound* if any cheating prover $\mathcal{P}'$ that convinces the verification algorithm that it is storing the a file M is actually storing that file. To formally define such a property we use the notion of *extractor* ([53]) and we say that if a cheating prover $\mathcal{P}'$ can convince $\mathcal{V}$ that it is storing a file M, then when an extractor has access to $\mathcal{P}'$ it will be able to output M.

Formally, an extractor Extr is a probabilistic polynomial time algorithm which takes as inputs the pair (pk, sk), a tag τ, the description of the machine $\mathcal{P}'$, and outputs a file M. Note that Extr has a non-black-box access to $\mathcal{P}'$ and can thus rewind it. To formalize the notion of soundness, we consider an experiment, which we call the *retrievability experiment*, in which an adversary $\mathcal{A}$ has access to a PoR scheme and plays the role of a cheating prover:

Definition 2.14. Let $\Pi = (\mathrm{Gen}, \mathrm{Store}, \mathcal{V})$ be a proof of retrievability scheme (without the prover), and $\mathcal{A}$ an adversary. Then the *retrievability experiment* $\mathrm{PoR\text{-}Exp}_{\Pi,\mathcal{A}}(\kappa)$ is as follows:

1. Gen is ran on input the security parameter 1^κ to obtain (pk, sk) and pk is provided to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ is given oracle access to $\mathrm{Store}(sk, \cdot)$. Let Q denote the set of queries of $\mathcal{A}$ to such an oracle.
3. For any $M \in Q$, the adversary $\mathcal{A}$ can execute the proof of retrievability protocol playing the part of the prover and providing the tag τ corresponding to M to the verifier. At the end of each execution the adversary is provided with the output of $\mathcal{V}$.
4. The adversary outputs a challenge tag τ' (corresponding to some $M \in Q$) and a description of a prover $\mathcal{P}'$.

We say that the cheating prover $\mathcal{P}'$ is ϵ-*admissible* if it convincingly answers the verification algorithm on input (pk, sk, τ') with probability greater or equal to ϵ.

We now have all the necessary tools to formally define a sound proof of retrievability:

Definition 2.15. Let τ' be a valid tag for M. We say that a proof of retrievability scheme Π is ϵ-*sound* if there exists an probabilistic polynomial time Extr such that for all adversary $\mathcal{A}$ and $(\mathcal{P}', \tau') \leftarrow \mathrm{PoR\text{-}Exp}_{\Pi,\mathcal{A}}(\kappa)$, if $\mathcal{P}'$ is ϵ-admissible, then $\mathrm{Extr}(pk, sk, \tau', \mathcal{P}') = M$ with overwhelming probability in κ.

Chapter 3

Model and Definition

We now introduce a formal model for capturing the notion of verifiable effort-based polling. The definition addresses 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 section 3.3.

3.1 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 carry forward an attack against the poll either by creating

"fake" responders (e.g. by replacing some of the $\perp$ inputs with adversarially chosen values), or by actually modifying the inputs submitted by honest responders. 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 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). In the protocol we present, verification will take place both locally and globally. For the latter, anyone can take the role of a verifier.

Definition 3.1 (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 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.

3.2 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 $\mathcal{A}^*$ 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 3.2 (one-sided PPE)**.** A protocol $\Pi^E(P, V)$ between a prover P and a verifier V is a one-sided PPE if it satisfies the following properties:

1. **Efficiency** An honest execution of $\Pi^E(P, V)$ requires P and V to make at most one oracle call to E (each).

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

3. ε**-Soundness** For every PPT P^* that executes an instance of $\Pi^E(P^*, V)$ using identifier *id*, if V honestly follows the protocol but P^* does not make at least one oracle call to E with input *id*, then the probability that V outputs "true" is at most ε.

Definition 3.3 (two-sided PPE)**.** A protocol $\Pi^E(A, B)$ between two parties A and B is a two-sided (symmetric) PPE if it is both a one-sided PPE $\Pi^E(A, B)$ and a one-sided PPE $\Pi^E(B, A)$.

3.3 Implementing PPEs

Below we give several examples of how to implement some simple PPEs.

Bitcoin and Proofs of Retrievability. The original motivation for proofs of retrievability (PoR) 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. Roughly, a PoR 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-retrievability 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).

Existing privately verifiable PoR's, such as the one introduced by Shacham and Water [53], can be trivially used to construct a PPE: an honest user sends good (incompressible) files to its peers (e.g., by encrypting the file), and it can verify using the PoR that the files were stored as required. The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ in the PPE also act as the prover and the verifier, respectively, in the PoR. Completeness for the PPE directly follows from the correctness of the PoR: if both parties are honest, the verifier will output 1. We can thus hope for almost perfect completeness. On the other hand, from the security definition of a PoR we know that if a verifier $\mathcal{V}$ outputs 1 when interacting with a cheating prover $\mathcal{P}'$ with probability greater than ϵ, then it is possible to *extract* m from $\mathcal{P}'$ with overwhelming probability. That is, the cheating prover $\mathcal{P}'$ would have actually had to spend the "effort" of storing such a file. Thus, using PoR's we can get a PPE with 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 [35]; 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.

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 as 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.

Symmetric CAPTCHAs.

In this version of the PPE, each party generates a standard 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).

As seen in Chapter 2, a CAPTCHA $\mathcal{C}$ consists of the pair (Gen, H), where H is a collection of solution functions $\{H_i\}_i$, and Gen is a randomized algorithm that on input a security parameter κ outputs a pair (t, s) such that t is a test, s a solution, and $H_\kappa(t) = s$. Anyone with access to the random coins r used by Gen would also be able to generate the exact pair (t, s). To make the fact that Gen uses random coins explicit, just for this example we will provide such coins r as input to Gen. We can then build a PPE based on $\mathcal{C}$ in the following way: let pk_P, pk_V, be the Prover's and Verifier's public keys, respectively, then:

Protocol 0

Step	$\mathcal{P}$		$\mathcal{V}$
1.		$\xleftarrow{t}$	Generates a secret key k and computes $r \leftarrow \mathrm{Mac}_k(pk_P \| pk_V)$. Let $\mathrm{Gen}(r) = (t, s)$.
2.	Solves the test t and creates a commitment $\boxed{\mathrm{s}}$ to the solution s.	$\xrightarrow{\boxed{\mathrm{s}}}$	
3.		$\xleftarrow{k}$	
4.	Verifies that the test t received was correctly generated, i.e. computes $r' = \mathrm{Mac}_k(pk_P \| pk_V)$ and verifies $(t, s) \stackrel{?}{=} \mathrm{Gen}(r')$. If test fails, abort protocol otherwise:	$\xrightarrow{\text{Open } \boxed{\mathrm{s}}.}$	

5.	If commitment and solution are correct, then outputs $b = 1$, otherwise outputs $b = 0$.

This protocol ensures that malicious parties cannot use an honest party to solve a CAPTCHA that was not meant for it, since a prover would refuse to open its commitment if it sees that the CAPTCHA was not correctly generated using the correct public keys. Note that in terms of the effort required, this is not harder than just solving a CAPTCHA, in fact the rest of the protocol can be completely automated.

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 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 publishes a commitment to r_i on his 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.

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, limiting a

poll to 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 verify physical proximity, we can guarantee the vast majority of participants must be local (assuming a large enough fraction is).

Another example is polling groups whose membership is secret (e.g., a poll of the "Anonymous" organization). If 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.

Limiting a poll to specific communities in an existing social network can be done similarly. 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.

Chapter 4

A Publicly Verifiable Polling Scheme

4.1 High-level description

The main technical innovation in this thesis 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 fraction 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.

Unless otherwise specified, from now on when referring to a double-sided PPE we will simply write PPE. Following is a high-level description of our protocol:

1. **Parameter Announcement.** 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 signing key and the verification key respectively (note that these are required only for completeness—responders will have their own signature and verification keys).

   The public parameters are the tuple $(id, questions, p, \mathrm{vk})$, where $questions$ is the set of poll questions, p is a probability that determines the expected degree $p \cdot m$ of the certification graph.

2. **Registration.** Each responder R_i samples a private key sk_i and a public key vk_i for their signature scheme, and sends $(addr_i, \mathrm{vk}_i)$ 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 $(addr_i, \mathrm{vk}_i)$ 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 neighbourhood of i" in the certification graph. The set N_i is computed from i and m (the total number of parties) using a cryptographic random function H: $j \in N_i$ iff $i < j$ and $H(i||j) \leq p$ (or, alternatively, $j < i$ and $H(j||i) \leq p$), 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 neighbours in the graph.

   However, while R_i may know the verification key of every neighbour, 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 neighbours in the graph.

3. **Responder Certification (PPE execution).** As just described, every pair of responders is paired in a PPE instance with probability p. Now, for each $j \in N_i$, 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.** Every responder R_i sends to the pollster the results of the certification phase (a signature on vk_i from each neighbour with which it successfully completed a PPE) and x_i, his actual response to the poll questions.

5. **Results and Proof.** 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 indices of the 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[1]:

   Node labels: For each i the pollster publishes

   $(x_i, sig_{\mathrm{sk}_i}(x_i), \mathrm{vk}_i)$.

   Edge signature: For each successful PPE the pollster publishes

   $(sig_{\mathrm{sk}_j}(\mathrm{vk}_i), sig_{\mathrm{sk}_i}(\mathrm{vk}_j))$, where $\mathrm{vk}_i, \mathrm{vk}_j$ are the public keys of the responders involved in it.

   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.

[1] 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.

6. **Verification.** 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 send a "complaint".

   Global verification (can be performed by anyone) consists of checking that all the responders, 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 neighbours 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 (nodes that do not correspond to any honest participant, but appear in G_p and whose "behavior" 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 detects the adversary and many valid complaints are 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 prevents this form of attack, as long as the cheating responders are limited enough in the amount of effort they can expend.

4.2 Formal Protocol Description

4.2.1 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 the 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 neither the pollster nor the honest parties know the identities of all the parties we assume that all communication are over *anonymous* channels. For more information on the properties and constructions of anonymous channels we refer the reader to the survey by Danezis and Diaz [17] devoted entirely to this subject. To simplify the analysis, we 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 Interactive Turing Machines) 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 any 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 the 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. Recall that in our scenario, the adversary coincides with the pollster.

When we describe communication between two responders (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 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 channel 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 it 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 does not use its privileged position in the communication graph in any way except to pass messages).

4.2.2 Adversary and Corruption Model

We consider two types of scenario: a scenario in which the pollster is malicious (e.g. he attempts to violate the soundness guarantee of our protocol), and one in which the pollster is actually honest. In the latter case, we still need to account for possible malicious responders who can attempt to violate the completeness guarantee of our protocol.

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, modelling "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 and can also copy the content of some messages send it forging the originator (as long as the fake source of the message is a corrupt party — it cannot 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).

4.2.3 Full Protocol

We divide the protocol into six phases (as described at the beginning of section 4.1). 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.

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

Signature Scheme In all of the subprotocols, we make use of a signature scheme, $(Gen, Sign, \mathrm{Ver})$, denoting the key generation, signing algorithm and verification algorithm respectively. As seen in Section 2, $Gen(1^\kappa)$ outputs a key pair (sk, vk), $Sign(msg, sk)$ outputs a signature $\sigma \in \{0,1\}^*$, and $\mathrm{Ver}(msg, \sigma, vk)$ outputs either 1 (if σ is a valid signature on msg with verification key vk) or 0 if not (if Ver outputs 1 we say the signature is accepted).

Pseudonyms and Network Address. 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 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 $(sid, vk_{src}, vk_{dst}, serial)$. In addition, sid will be used as the poll identifier in the PPE executions.

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 received 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 party A and B engage in a PPE execution, and party A does not successfully verify party B, it treats this as if B has aborted).

Dealing with Registration-Refusal Attacks. One attach that our protocol does not handle robustly is a registration-refusal attack: this is when a malicious pollster refuses to *register* a subset of honest voters. The honest voter will detect this immediately, but it may be hard to generate a proof that will convince other verifiers that this is malicious behavior by the pollster. We suggest several directions for dealing with these type of attacks:

1. *Require registration complaints to include publicly-verifiable proofs of effort*. In this case, we modify the global verification protocol to count registration complaints with valid proofs of effort, and fail if too many are received (where "too many" is determined by the effort bound for malicious responders). We note that it could make sense to use publicly-verifiable proofs here even though the point of the protocol is to use privately-verifiable proofs, since a failure of this assumption impacts completeness, not soundness.

2. *Use "semi-verifiable" one-sided PPEs.* For example, if the one-sided PPE is a CAPTCHA, the correctness of the solution can be publicly verified (although possibly requiring human effort), even if it may not be possible to verify that effort was expended. In this case, the registration complaint can contain the transcript of the session which includes the CAPTCHA signed by the pollster, as well as the solution signed by the responder. If the pollster receives such a broadcast in the registration phase, but does not include the responder in the responder list, this is a publicly-verifiable proof of malicious behaviour by the pollster. (If the pollster aborts before giving the one-sided PPE in the registration phase, the responder can try again without spending effort - so this only has an effect if the pollster refuses to register users with overwhelming probability).

3. *Use anonymous channels*. If responders communicate with the pollster via anonymous channels, the pollster cannot base the refusal on the responder's identity, and so should

not be able to bias the poll unless a significant fraction of honest responders are refused. This is the case because our protocol guarantees that the pollster cannot control "too many" responders under the assumption that at least α fraction of the responders are indeed honest. Thus, in order to bias the results he would need to refuse either specific votes or a lot of them. In the latter case, honest parties who try to register to the poll will be refused with high probability. Therefore, in this case, honest parties who receive a registration complaint can try to register themselves to check if they are refused (whether or not they are already registered); unfortunately, this does not allow post-factum global verification.

Protocol 1 Parameter Announcement

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

1: $(\mathrm{sk}, \mathrm{vk}) \leftarrow Gen(1^\kappa)$ ▷ Generate pollster's key pair

2: Broadcast $(sid, \kappa, p, \mathrm{vk}, questions)$ to all parties.

4.3 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 neighbouring nodes allows the honest user to verifiably complain about being omitted from the count despite

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 $i \in [m]$, R_i and the pollster execute:

1: **(Responder R_i):** $(\mathrm{sk}_i, \mathrm{vk}_i) \leftarrow Gen(1^\kappa)$ ▷ Generate responder's key pair

2: **(Responder R_i):** send $(addr_i, \mathrm{vk}_i)$ 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 the verification was successful, add $(addr_i, \mathrm{vk}_i)$ to the responder list and send $\mathrm{Sign}_{\mathrm{sk}}(addr_i || \mathrm{vk}_i)$ to the responder (note that this serves as a registration confirmation).

5: **(Responder R_i):** If the verification was successful but the responder did *not* receive the confirmation from the pollster, broadcast a "registration complaint".

When the registration phase is completed:

1: **(Pollster):** Choose a random permutation $\pi : [m] \to [m]$ and broadcast the shuffled list $(\mathrm{vk}_{\pi(1)}, \ldots, \mathrm{vk}_{\pi(m)})$. ▷ The shuffle is used only as defense against malicious responders; to reduce clutter we ignore it except in the completeness proof; from now on, when we write i we actually mean $\pi(i)$.

2: **(Responder R_i):** Verify that vk_i appears in the list. If not, broadcast an "identity missing" complaint, that includes the registration confirmation message with the pollster's signature (as received from the pollster).

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 section 4.3.

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

Protocol 3 Responder Certification

This protocol is executed between every responder R_i and its neighbours in the certification graph: $N_i = \{R_j | j \in [m], j \neq i, H(i,j) < p\}$. Note that N_i can be computed independently by R_i given the public parameters H, p, and the list $(\mathrm{vk}_1, \ldots, \mathrm{vk}_m)$. 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 $(sid, \mathrm{vk}_{min\{i,j\}}, \mathrm{vk}_{max\{i,j\}})$.
3: If $\Pi^E(R_i, R_j) = 1$, R_i computes $\sigma \leftarrow \mathrm{Sign}_{\mathrm{sk}_i}(sid || \mathrm{vk}_j)$ and sends σ to R_j.
4: Let σ_j be the corresponding signature received from R_j
5: **end for**
6: Let $Good_i = \{\sigma_j | \mathrm{Ver}(sid || \mathrm{vk}_i, \sigma_j, \mathrm{vk}_j) = 1\}$ be the set of valid signatures from R_i's neighbours.

Protocol 4 Poll Response

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

For every $i \in [m]$:

1: R_i sends $resp_i = (x_i, Good_i, \mathrm{Sign}_{\mathrm{sk}_i}(sid || x_i || Good_i))$ to the pollster.

Protocol 5 Results and Proof

This protocol is executed by the pollster.

1: **for all** $i \in [m]$ **do**
2: Broadcast $resp_i$
3: **end for**

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.

Protocol 6 Verification

This protocol is executed by the responders and verifiers.

Local Verification. Executed by every responder. For every $i \in [m]$:

1: Verify that $resp_i$ was published correctly.

2: If $resp_i$ was not published, broadcast a "response-missing" complaint.

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

1: **if** a valid "identity-missing" complaint was broadcast by vk^* ▷ i.e. $\mathrm{vk}^* \notin \{\mathrm{vk}_1, \ldots, \mathrm{vk}_m\}$, but the complaint contains a registration-confirmation message signed with key sk **then**

2: Output $\perp$ // Malicious Pollster

3: **end if**

4: Set $\mathrm{badNodes} \leftarrow 0$.

5: Set $Complaints \leftarrow 0$.

6: Initialize an array $count[]$ indexed by the poll answers.

7: **for all** $i \in [m]$ **do**

8: Verify that $resp_i$ contains a valid signature from R_i.

9: Verify that $Good_i$ is valid (all signatures are valid).

10: **if** $|N_i| - |Good_i| < \eta_E d$ **then**

11: Increment $count[x_i]$. ▷ We don't count nodes if they failed more than $\eta_E d$ PPEs

12: **else**

13: Increment $\mathrm{badNodes}$

14: **end if**

15: **if** R_i broadcast a response-missing complaint **then**

16: Increment $complaints$.

17: **end if**

18: **end for**

19: **if** $complaints > \theta m$ **then**

20: Output $\perp$. ▷ Too many complaints.

21: **end if**

22: **if** $\mathrm{badNodes} > \eta_V m$ **then**

23: Output $\perp$. ▷ Too many deleted nodes.

24: **end if**

25: Output the array $count$.

4.3.1 Large-Set Expanding Property

The LSE property is similar to the "jumbled" graphs of Thomason [54, 55], but is weaker since we do not 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 (formally, we use the $G(n,p)$ model for random graphs; a graph is distributed according to $G(n,p)$ if it has n vertices and for each pair of vertices the corresponding edge exists with probability p).

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 $G \in G(m,p)$, 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 $K \leq |A| \leq \frac{m}{2}$ and $m - |A| - \rho \leq |B| \leq m - |A| - 1$, 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:

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

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

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

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

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

Hence

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

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

Applying the union bound again, we get

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

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. □

4.3.2 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 $K = \kappa + (\eta_V m + 2)\ln m + \eta_V m$.[2] 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 4.1 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

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

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

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 fails. 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 a lower bound on the number of crossing edges between a possible set of bad nodes and the set of honest nodes.
- We conclude by noticing that the pollster, in order to control a set of nodes larger than B, would have had to succeed 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 labelled 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:

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

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

□

4.4 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 in the poll. We think of the number of maliciously controlled nodes as bounded by θm, where θ depends on the "effort" invested by the malicious voters. Theorem 4.7 gives a bound on η_V as a function of θ and κ that enables 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 require a corollary to the following lemma:

Lemma 4.5. *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,

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

□

Corollary 4.6. *Assuming static corruption, the probability that malicious responders with a θm upper bound on effort (in total) can control $\max\{3\theta md, 3\kappa\}$ edges in the certification graph is bounded by $e^{-\kappa}$.*

Proof. 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 Lemma 4.5. □

We are now ready to prove the Completeness Theorem:

Theorem 4.7 (Completeness)**.** *Let θm denote the maximum number of effort-oracle calls that can be made by malicious responders and*

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

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.

Proof. Let $G = (V, E)$, with $p = d/m$, be the random graph generated by the pollster. Recall that an edge (i, j) is labelled as *failing* whenever the double-sided 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. Recall that the certification graph is an undirected graph and that an edge between two nodes denotes the requirement of running a double-sided PPE between the responders represented by those two nodes. Thus, the event "$(i, j) \in E$ is a failing edge" is

actually equivalent to the event "$(j,i) \in E$ is a failing edge". Therefore, X is double counting the number of failing edges. Since each edge affect 2 nodes, X is actually the cardinality of the set containing (with repetitions) all the nodes affected by $X/2$ failing edges. Note that for a node to be labelled 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 X. Thus, the expected number of deleted nodes in case of honest responders is bounded by $X/d\eta_E$. 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 deleted nodes;
2. Enough bad edges will cause an honest node to be marked deleted. However, by Corollary 4.6, 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 labelled as deleted.

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

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

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 $\Pr\left[X > (1+\delta)md\sigma\right] \leq e^{-\kappa}$. □

As previously stated, our proofs hold in the random oracle model (e.g. the analysis has been done assuming that the certification graph G has been generated using $G(n,p)$). Under the

reasonable assumption that the cryptographic random function H used to generate the certification graph would indeed produce a graph with good expansion parameters, our proofs would hold in the standard model as well. In fact, note that to carry forward both the soundness and the completeness proofs, we relied only on the expansion properties of the certification graph. Thus, the construction of the graph can potentially be derandomized; we could use an explicit graph construction with the appropriate expansion properties. We would have a more elaborate protocol, but we could remove the assumption on H and improve the protocol parameters. In practice, the pollster will still broadcast a parameter which will give a lower bound on the expected degree of the graph and, after the registration phase he will simply broadcast a specific graph G. As part of the verification procedure, the verifiers will make sure that G is indeed LSE.

Chapter 5

Conclusion

5.1 Summary

We introduced a formal model defining verifiable effort-based polling. The idea was to construct a protocol whose results can be publicly verified and whose security did not depend on a trustful central authority. We achieved our goal by introducing the new concept of Privately verifiable Proof of Efforts (PPE). Our protocol generates a "responders certification graph" where nodes are associated to responders, and edges denote the requirement of cross-certification between adjacent nodes. Responders who appear to be connected in the graph will have to certify each others "honesty" through the use of PPEs. We guarantee that if enough honest users participate to the poll, a cheating pollster will be detected with high probability. Note that in our analysis the certification graph is randomly generated, but to prove our results we simply considered a specific expansion property of the graph. Therefore, a deterministic construction of a certification graph would still allow us to obtain the same results provided that the expansion property needed is achieved.

5.2 Future Work

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 neighbourhood and a path from that neighbourhood to the root in the tree. Thus, if we can assume that enough honest responders participate in the verification, the total amount of 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.

Appendix A

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 A.1) - those that depend on assumptions about adversarial behavior and the effectiveness of the PPEs, *tunable* parameters (in Table A.2) - those that can be set by the poll designer (subject to certain constraints), and *computed* parameters (in Table A.3) - those that are functions of the previous parameters.

TABLE A.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.
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.

Continued on next page

Table A.1 – continued from previous page

Symbol	Description
ϵ	Probability of a PPE succeeding when one party does not make at least one oracle call.

TABLE A.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 A.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 PPEs 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 A.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).

A.1 Constraints on Parameters

First, from Theorem 4.4 we have:

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

which implies that

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

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

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

Isolating η_V instead of α, we have:

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

Combining this with the bound on η_V from Theorem 4.7, 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:

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

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:

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

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

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

A.2 Examples of Parameter Settings

For simplicity we consider PPEs for which the soundness error ϵ is negligible and we 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 Section 3.3, 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-retrievability 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), require fewer (or different) resources.

In Table A.4 the reader can find example parameter settings for two parameter regimes: in Scenario 1 and 2, there are 5000 responders and PPEs are error-free, while Scenario 3 and 4 have 100000 responders with PPEs that have 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 A.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	$1/1000$	$1/1000$	$1/10000$	$1/10000$

Continued on next page

Table A.4 – continued from previous page

Symbol	Scenario 1	Scenario 2	Scenario 3	Scenario 4
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
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.

Bibliography

[1] B. Adida and R. L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40. ACM, 2006.

[2] G. Alberini, T. Moran, and A. Rosen. Public verification of private effort. In *Theory of Cryptography*, pages 169–198. Springer, 2015.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[4] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged byzantine impostors. *Department of Computer Science, Yale University, New Haven, CT, Tech. Rep*, 2005.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. Remote data checking using provable data possession. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):12, 2011.

[6] B. Awerbuch and C. Scheideler. Group spreading: a protocol for provably secure distributed name service. In *Automata, Languages and Programming*, pages 183–195. Springer, 2004.

[7] A. Back. Hashcash-a denial of service counter-measure. `http://www.cypherspace.org/hashcash`, 2002.

[8] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory A*, 24:296–307, 1978.

[9] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

[10] B. Bollobás. *Random Graphs*. Springer, 1998.

[11] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[12] R. Canetti, S. Halevi, and M. Steiner. Mitigating dictionary attacks on password-protected local storage. In *Advances in Cryptology-CRYPTO 2006*, pages 160–179. Springer, 2006.

[13] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[14] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, (1):38–47, 2004.

[15] D. Chaum, P. Y. A. Ryan, and S. Schneider. *A practical voter-verifiable election scheme*. Springer, 2005.

[16] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132. ACM, 2005.

[17] G. Danezis and C. Diaz. A survey of anonymous communication channels. Technical report, Technical Report MSR-TR-2008-35, Microsoft Research, 2008.

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

[19] S. Diaz-Santiago and D. Chakraborty. On securing communication from profilers. In *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 154–162, 2012.

[20] J. R. Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[21] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology—CRYPTO'92*, pages 139–147. Springer, 1993.

[22] S. Dziembowski. How to pair with a human. In *Security and Cryptography for Networks*, pages 200–218. Springer, 2010.

[23] C. M. Ellison. Establishing identity without certification authorities. In *USENIX Security Symposium*, pages 67–76, 1996.

[24] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[25] P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

[26] P. Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, 12(1):261–267, 1961.

[27] I. Haitner, O. Horvitz, J. Katz, C.-Y. Koo, R. Morselli, and R. Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. *Journal of Cryptology*, 22(3):283–310, 2009.

[28] I. Haitner, M.-J. Nguyen, S. J. Ong, O. Reingold, and S. P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[29] I. Haitner, O. Reingold, S. P. Vadhan, and H. Wee. Inaccessible entropy. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 611–620. ACM, 2009.

[30] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[31] S. Janson, T. Luczak, and A. Rucinski. *Random Graphs*. New York: Wiley & Sons, 2000.

[32] A. Juels and J. G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, volume 99, pages 151–165, 1999.

[33] A. Juels and B. S. Kaliski. PORs: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. ACM, 2007.

[34] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

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

[36] A. Kumarasubramanian, R. Ostrovsky, O. Pandey, and A. Wadia. Cryptography using captcha puzzles. In *Public-Key Cryptography–PKC 2013*, pages 89–106. Springer, 2013.

[37] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *Usenix Security*, 1998.

[38] B. N. Levine, C. Shields, and N. B. Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 2006.

[39] M. D. Lillibridge, M. Abadi, K. Bharat, and A. Z. Broder. Method for selectively restricting access to computer systems, February 27, 2001. US Patent 6,195,698.

[40] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[41] G. A. Margulis. Explicit constructions of graphs without short cycles and low density codes. *Combinatorica*, 2(1):71–78, 1982.

[42] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problemy peredachi informatsii*, 24(1):51–60, 1988.

[43] R. Meshulam and A. Wigderson. Expanders from symmetric codes. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 669–677. ACM, 2002.

[44] T. Moran and M. Naor. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In *Advances in Cryptology-EUROCRYPT 2006*, pages 88–108. Springer, 2006.

[45] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Unpublished*, 2008.

[46] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[47] M. Naor. Verification of a human in the loop or identification via the Turing Test. *Unpublished draft from* http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human_abs.html, 1996.

[48] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

[49] M. Naor and G. Rothblum. The complexity of online memory checking. In *Proceedings of FOCS*, pages 573–84. IEEE Computer Society, 2005.

[50] C. A. Neff. Practical high certainty intent verification for encrypted votes. http://www.votehere.com/documents.php, 2004.

[51] R. L. Rivest. The ThreeBallot voting system. http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf, 2006.

[52] E. Rozenman, A. Shalev, and A. Wigderson. A new family of Cayley expanders (?). In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 445–454. ACM, 2004.

[53] H. Shacham and B. Waters. Compact proofs of retrievability. In *Advances in Cryptology-ASIACRYPT 2008*, pages 90–107. Springer, 2008.

[54] A. Thomason. Pseudo-random graphs. *North-Holland Mathematics Studies*, 144:307–331, 1987.

[55] A. Thomason. Random graphs, strongly regular graphs and pseudorandom graphs. *Surveys in combinatorics*, 123:173–195, 1987.

[56] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings NSDI*, 2009.

[57] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311. Springer, 2003.

[58] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[59] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[60] N. C. Wormald. The asymptotic connectivity of labelled regular graphs. *Journal of Combinatorial Theory, Series B*, 31(2):156–167, 1981.

[61] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE, 2008.

[62] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, no. 4, pages 267–278. ACM, 2006.

[63] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. Technical report, Intel Research, Pittsburgh, PA, June 2006. http://www.pittsburgh.intel-research.net/people/gibbons/papers/sybilguard-tr.pdf.