Gossiping For Communication-Efficient Broadcast

Georgios Tsimos[1], Julian Loss[2], and Charalampos Papamanthou[3]

[1] University of Maryland, tsimos@umd.edu
[2] University of Maryland, jloss@umiacs.umd.edu
[3] University of Maryland, cpap@umd.edu

Abstract. Broadcast (BC) is a crucial ingredient for a plethora of cryptographic protocols such as secret sharing and multiparty computation. In this paper we apply *gossiping* (propagating a message by sending to a few random parties who in turn do the same, until the message is delivered) to design new randomized BC protocols with improved communication complexity and which are secure against an adversary controlling the majority of parties. We make progress on two fronts. First, we propose a protocol for single-sender BC in the static model of corruption that achieves $\tilde{O}(n^2 \cdot \kappa^2)$ bits of communication and where no trusted setup is required—parties just need to generate their own cryptographic keys. All prior protocols in this setting exhibit $O(n^3 \cdot \kappa)$ communication. Using insights from our single-sender BC protocol, we then propose the first adaptively-secure parallel BC protocol with $\tilde{O}(n^2 \cdot \kappa^4)$ communication complexity, significantly improving existing parallel BC protocols of $\tilde{O}(n^3)$ communication. To the best of our knowledge, our parallel BC protocol is the first non-trivial one, i.e., one that is not using a single-sender BC protocol n times and in a black box fashion, thus leading to the improved complexity.

1 Introduction

Since its formalization by Lamport et al. [16], the problem of broadcast (BC) has been studied in countless works and remains at the pinnacle of interest in the era of cryptocurrencies and blockchains. In its *single-sender* version, BC involves a designated sender s that distributes a value v such that (1) all parties output the same value v' (consistency); (2) all parties output v in case s is honest (validity). *Parallel* BC (PBC) (also known as interactive consistency [20]) is a generalization of the single-sender setting where all parties act as designated senders, and in the end each party outputs values, satisfying the above conditions, from every sender. Broadcast protocols lie at the core of many cryptographic protocols such as secret sharing and multiparty computation and have also recently been proven useful in the context of blockchain consensus protocols (e.g., [17]). Interestingly, in many of the above applications, BC is mostly used as a library that is invoked by *every* party simultaneously, which implicitly gives a (trivial) PBC protocol. An important metric for any distributed protocol is its *communication complexity*, i.e., how many bits are exchanged during the protocol. Often, it is closely related to the communication efficiency of the underlying protocols using BC or

protocol	model	communication	adversary	malicious parties	type
Abraham et al. [1]	trusted PKI	$\tilde{O}(n \cdot \kappa)$	adaptive	$< n/2$	BC
Chan et al. [5]	trusted PKI	$O(n^2 \cdot \kappa^2)$	adaptive	$< (1-\epsilon) \cdot n$	BC
Dolev and Strong [7]	bulletin PKI	$O(n^3 \cdot \kappa)$	adaptive	$< n$	BC
Momose and Ren [19]	bulletin PKI	$\tilde{O}(n^2 \cdot \kappa)$	adaptive	$< n/2$	BC
RandomBroadcast §3	bulletin PKI	$O(n^2 \cdot \kappa^2)$	static	$< (1-\epsilon) \cdot n$	BC
ParallelBroadcast §5	trusted PKI	$\tilde{O}(n^2 \cdot \kappa^4)$	adaptive	$< (1-\epsilon) \cdot n$	PBC

Table 1. Comparison of RandomBroadcast and ParallelBroadcast, in terms of communication (number of bits), to existing work. n the number of parties and $\epsilon < 1$.

PBC. However, while there is a lot of work on the communication complexity of simple BC (see Table 1) no results exist (to the best of our knowledge) on the communication complexity of PBC *specifically*.

Contributions. In this work, we revisit the communication complexity of both BC and PBC. See Table 1. We focus on the dishonest majority setting with both a static and an adaptive adversary and *we provide the first subcubic protocols (under certain assumptions) for both BC and PBC*. Additionally, to the best of our knowledge, our PBC protocol is the first to not use BC as a black box, leading to the improved communication bound.

Message Propagation. We achieve our improvements by optimizing one of the fundamental aspects of BC protocols: their *message propagation*. When an honest party sends out a message at some round, propagation ensures that all honest parties receive this message soon. In most protocols, this is implemented via an expensive SEND-ALL instruction delivering the message in a single round [5,7]. However, such instructions might not always be needed. For example, it might not be crucial to deliver the message strictly in the next round. Our proposed protocols for BC and PBC minimize the use of SEND-ALL instructions via *gossiping*, eventually yielding much better communication for the proposed protocols. In particular, in message propagation via gossiping, a party first sends the message M only to a small random set of parties, who in turn do the same, until M is delivered. Somewhat surprisingly, the effect of this simple technique on the complexity of broadcast had not been explored before.

Bulletin PKI versus Trusted PKI. Our proposed protocols are designed in two distinct trust models, the *bulletin PKI* model and the *trusted PKI* model. In bulletin PKI, *no trusted setup* is required: All parties register their public keys to a public bulletin board before the start of the protocol and no assumption is made on how parties generate their keys. In trusted PKI, *trusted setup* is required: A trusted party, Alice, generates all keys honestly and distributes them to the parties prior to protocol execution while at the same time she must remain uncompromised for the rest of the protocol. Clearly, trusted PKI is a *stronger* assumption than bulletin PKI.

1.1 Communication-Efficient BC in the Bulletin PKI Model

In our first contribution (Section 3) we apply gossiping to achieve communication-efficient BC protocols *in the bulletin PKI model*. When using bulletin PKI, it is widely known that BC can be solved for arbitrary $t < n$ malicious parties with $O(n^3 \cdot \kappa)$ bits of communication using the seminal result of Dolev and Strong [7] (Here κ is the security parameter and represents the size of a digital signature.) However, to the best of our knowledge no protocol with better communication complexity exists. We resolve this question by introducing RandomBroadcast (see Figure 2), the first BC protocol that achieves communication complexity of $\tilde{O}(n^2 \cdot \kappa^2)$ bits in the bulletin PKI model. Our protocol is randomized and works for $t < (1 - \epsilon) \cdot n$ corrupted parties where $\epsilon \in (0, 1)$ is a constant. On the downside, we assume a static model of corruption where the adversary must decide which t parties to corrupt before the execution—but the corrupted parties are byzantine.

Technical Highlights. Our protocol follows a similar framework with the Dolev-Strong protocol [7]. Recall that in Dolev-Strong, for all rounds $r < n$, whenever an honest party p has observed r signatures on a bit b for the first time, p adds her signature and sends a message x of $r + 1$ signatures *to all parties*, using a $\mathsf{SEND\text{-}ALL}(x)$ instruction. Our proposed protocol *just replaces* the $\mathsf{SEND\text{-}ALL}(x)$ instruction with a $\mathsf{SEND\text{-}RANDOM}(m, x)$ instruction and runs for an additional $O(\log n)$ rounds. Our $\mathsf{SEND\text{-}RANDOM}(m, x)$ instruction is implemented by sending message x to party i (for all $i \in [n]$) with probability m/n for some fixed $m = \Theta(\kappa)$. It is important to note, however, that our protocol does not merely implement $\mathsf{SEND\text{-}ALL}(x)$ via a sequence of $\mathsf{SEND\text{-}RANDOM}(m, x)$ instructions as this would still lead to cubic communication complexity—again, it just replaces the instructions. Unfortunately our gossiping technique does not protect against adaptive adversaries who can simply wait and corrupt recipients of $\mathsf{SEND\text{-}RANDOM}(m, x)$ commands during the protocol. Still, this is not fundamental: We show next that gossiping, when used in PBC can (quite surprisingly) overcome this issue.

1.2 Adaptively-Secure PBC in the Trusted PKI Model

In our second contribution (Section 5) we use gossiping to improve the communication complexity of PBC. Recall that in PBC all n parties simultaneously act as a sender and wish to consistently distribute their message. To the best of our knowledge, all PBC protocols that are used in the literature are derived trivially by calling BC multiple times in a black-box fashion, which leads to a multiplicative n-factor in the communication complexity. For example, deriving PBC for $t < (1 - \epsilon) \cdot n$ via n parallel executions of the best BC protocol so far [5], would yield $\tilde{O}(n^3)$ communication complexity. The main question we are considering here is whether gossiping can help in deriving a non-black-box version of PBC, possibly with better communication complexity. We answer this question by providing the first adaptively-secure protocol for PBC, ParallelBroadcast (see

Figure 9), with $\tilde{O}(n^2 \cdot \kappa^4)$ overall communication complexity. Thus, we achieve $\tilde{O}(n \cdot \kappa^4)$ amortized communication complexity per sender.

Technical Highlights. Our starting point is the recent protocol by Chan et al. [5] that solves BC in the trusted PKI model with $O(n^2 \cdot \kappa^2)$ communication complexity. We call this protocol ChBC from now on. ChBC is essentially a Dolev-Strong protocol run among a random committee of κ parties—for the rest of honest parties to agree with the committee, a distribution phase takes place. More concretely, ChBC at round $r < \kappa$ instructs parties to perform the following.

1. (Voting) Whenever an honest committee party p has observed r signatures on a bit b for the first time, p adds her signature and sends a message of $r+1$ signatures *to all parties*, using a SEND-ALL instruction. Note this step is executed only by the committee and the size of the message sent is at most κ signatures of κ bits each; hence the induced communication complexity is $O(n \cdot \kappa^3)$;
2. (Distribution) Whenever an honest party p has observed r signatures on a bit b for the first time, p just forwards the r signatures *to all parties*, using a SEND-ALL instruction. Note that this step is executed by all parties and therefore induces $O(n^2 \cdot \kappa^2)$ communication complexity (It is $n^2 \cdot \kappa^2$ since every party sends once to all n parties a list of at most κ signatures of κ bits each.)

Our Approach: ConvergeRandom Protocol. We first observe that if we naively use the ChBC protocol for the parallel case, the communication complexity of the distribution phase will grow to $O(n^3 \cdot \kappa^2)$, since the SEND-ALL instruction would have to be used at most $2 \cdot n$ times (instead of one), for each bit b and for each party p. To overcome this issue we abstract away what this naive modification of ChBC achieves by introducing the Converge problem. In the Converge problem, all honest parties p begin with a set of messages M_p; In the end, all remaining honest parties q (we consider an adaptive adversary) should output a set S_q that is a *superset* of $\bigcup_p M_p$ (We use superset and not equality because the adversary can inject arbitrary messages.) For a formal definition of the Converge problem, see Definition 5.

We are able to come up with an efficient and quite simple protocol that solves the Converge problem, the ConvergeRandom protocol—see Figure 6. The main idea is very simple:

1. In round 1, honest party p, for every message $x \in M_p$, picks recipient $i \in [n]$ with probability m/n (Here $m = \Theta(\kappa)$.) Then party p constructs lists $\mathcal{L}_j$ (for $j = 1, \ldots, n$) containing messages $x \in M_p$ that were assigned to recipient j in the previous step. Padding is used to ensure an adaptive adversary does not gain any advantage and list $\mathcal{L}_j$ is sent out to party j for $j = 1, \ldots, n$;
2. In rounds $i = 2, \ldots, \lceil \log \epsilon \cdot n \rceil$, every honest party p collects all lists $\mathcal{L}_p$ from the round $i-1$, compiles *their union* into a new message set M_p and performs the same task (random assigning of elements in M_p, compilation into lists and sending) as before.

We prove that with overwhelming probability, the above protocol delivers every element contained in the sets of the initially-honest parties to all remaining honest parties (see Lemma 9). To compute the communication complexity of the above protocol we make an important observation: The number of elements in M_p, for any round and for any honest party is always $O(n)$. This is because in our setting, messages are valid r-batches (a set of r valid signatures) for party-bit pairs and there can only be at most $2n$ of them! In particular, if an honest party receives, at some round i two different valid r-batches on the same party-bit pair, it can safely ignore one of them. Therefore the communication complexity of ConvergeRandom is $O(n^2 \cdot \kappa^4 \cdot \log n)$ since ConvergeRandom is run for κ rounds, and for every inner iteration of ConvergeRandom, every honest party sends a message of at most $|M_p| \cdot m/n = \Theta(\kappa)$ elements to all parties and each element is at most κ signatures of κ bits each.

Adaptive Security of ConvergeRandom. It turns out that proving ConvergeRandom is adaptively secure (which translates into adaptive security of our ParallelBroadcast protocol) requires a lot of care. In particular, the crux of our proof rests on Lemma 5 stating that during the propagation process, an adversary corrupting a party that just sent does not get any further information about who to corrupt next. Note that this was trivial to establish in the original ChBC protocol since honest parties send to all. However, in our case a corruption can expose the party's state and therefore who the party *actually* sent to, guiding the adversary's next choice. To address this, we require our protocol be secure in the erasure model (e.g., as in [14,2]), where honest parties can erase their states at appropriate times in the protocol—see Line 11 in the code of our Propagate procedure in Figure 5.

1.3 Related Work

The problem of BC was originally introduced in the celebrated work of Lamport, Shostak, and Pease [16]. Their work also gave the first (setup-free) protocol for $t < n/3$ and showed optimality of their parameters. However, their solution required an exponential amount of communication and was soon improved upon by protocols requiring only polynomial amounts of communication [8,11]. More recently, a line of work initiated by King et al. [15,13,3] gave setup-free protocols for the case of $t < n/3$ that require $\tilde{O}(n^{3/2})$ communication and Momose and Ren [19] provide a protocol in the bulletin PKI model with $\tilde{O}(n^2 \cdot \kappa)$ communication complexity but in the honest majority setting. For the setting of $t < n$ corruptions, Dolev and Strong [7] gave the first protocol with polynomial efficiency. Their protocol uses a bulletin board PKI, requires $O(n^3 \cdot \kappa)$ bits of communication, and solves BC for any $t < n$. Much more recently, the work of Chan et al. [5] gives a protocol that requires $\tilde{O}(n^2 \cdot \kappa)$ bits of communication and requires trusted setup. In the range of $t < n/3$ and $t < n/2$, the works of Micali [17], Micali and Vaikuntnathan [18], and Abraham et al. [1] present solution with subquadratic communication complexity using trusted setup. Somewhat surprisingly, in the setting with setup (for $t < n$), any efficiency improvement to

the early work of Dolev and Strong has been aimed exclusively at improving the *round complexity* rather than the communication complexity. This has been the subject of several works [10,9,5,22]. Finally, the problem of interactive consistency or parallel broadcast was originally introduced by Pease et al. [20]. Another line of works studies the *round complexity* of BC, examples are [10,5,23,21]. We give an overview over communication efficient protocols in Table 1.

2 Preliminaries and Notation

We denote as $X \leftarrow \Pi$ the random variable X output by probability experiment Π. Our protocols are run among a set of n parties out of which $t = (1-\epsilon) \cdot n$ can be malicious, for constant $\epsilon < 1$. We use κ to indicate the security parameter.

Bulletin PKI. For the first part of our work, we assume that parties share a *public key infrastructure* (PKI). That is, each party i has a secret key sk_i and a public key pk_i, where pk_i is known to all parties. The secret key sk_i and the public key pk_i are not assumed to be computed in a trusted manner. Instead, we assume only that each party i generates its keys (sk_i, pk_i) locally and then makes pk_i known by using a public bulletin board. Each party i can compute a *signature* σ on a message m via $\sigma \leftarrow sig(sk_i, m)$. Later, anybody can verify σ via calling $ver(pk_i, \sigma, m)$. As is standard for this line of work, we assume that signatures are *idealized*, that is, we treat signatures as *perfectly unforgeable* in the sense that it is impossible, without sk_i, to create a signature σ on a message m such that $ver(pk_i, \sigma, m) = 1$. We also assume *perfect correctness*, meaning that for any m, $ver(pk_i, sig(sk_i, m), m) = 1$. To simplify notation, we write $sig_i(m)$ to indicate $sig(sk_i, m)$ and $ver_i(\sigma, m)$ to indicate $ver(pk_i, \sigma, m)$.

Network and Synchrony Model. We consider the standard synchronous model of communication. In this model, parties are assumed to share a global clock that progresses at the same rate for all parties. Furthermore, they are connected via pairwise, authenticated channels. Any message that is sent by an honest party at time T is guaranteed to arrive at every honest party at time $T + \Delta$, where Δ is the maximum network delay. In particular, this means that messages of honest parties can not be dropped from the network and are always delivered. It is assumed that all parties know the parameter Δ. As such we consider protocols that execute in a round based fashion, where every round in the protocol is of length Δ and parties start executing the r-th round of a protocol at time $(r-1) \cdot \Delta$. Let $\mathcal{M}$ be a set of messages and $\mathcal{P}$ be a set of parties. When a party i calls $\textsc{Send}(\mathcal{M}, \mathcal{P})$ at round r, then the set of messages $\mathcal{M}$ is delivered to parties in $\mathcal{P}$ by round $r+1$. Finally, when a party i calls $\textsc{Receive}()$ in round r, then all messages that were sent to i in round $r-1$ via $\textsc{Send}$ commands are stored in i's local storage.

t-Secure Broadcast and t-Secure Parallel Broadcast. We now begin with the definitions of t-Secure broadcast and t-Secure Parallel Broadcast which are the focus of Section 3 and Section 5 respectively.

Definition 1 (t-Secure Broadcast). *A protocol Π executed by n parties, where a designated party $s \in [n]$ (the sender) holds an input v and parties terminate*

with an output is a t-secure broadcast protocol *if the following properties are satisfied with probability* $1 - \mathsf{negl}(\kappa)$ *whenever at most* t *parties are corrupted:*

> t-***validity:*** *if the sender is honest, all honest parties output* v.
> t-***consistency:*** *all honest parties output the same value* v'.

Definition 2 (*t*-Secure Parallel Broadcast). *A protocol* Π *executed by* n *parties, where each party* p_i *holds an input* v_i *and defines a slot* i *and all parties terminate with some output, is a* t-secure parallel broadcast protocol *if the following properties are satisfied with probability* $1 - \mathsf{negl}(\kappa)$ *whenever at most* t *parties are corrupted:*

> t-***validity:*** *for all slots* $s \in [n]$ *with* p_s *honest, all honest parties output* v_s.
> t-***consistency:*** *for all slots* $s \in [n]$, *all honest parties output same value* v'_s.

Adversary Models. In general for all our protocols we consider a polynomial-time adversary that can corrupt up to t parties in a malicious fashion. The adversary can make them deviate from the protocol description arbitrarily. Our adversary is also rushing, being able to observe the honest parties' messages in any synchronous round r of a protocol, and delay them until the end of that round. In this way, it can choose its own messages for that round before delivering any of the honest messages.

Now, specifically for our RandomBroadcast in Section 3, the adversary is *static* in that the adversary chooses what parties to corrupt *before* the execution of the protocol. For our ParallelBroadcast in Section 5, the adversary is *adaptive*, being able to corrupt up to t parties, each at any point during the execution of the protocol, learning its internal state which consists of any longterm secret keys and ephemeral values that have not been deleted at that point. We do not consider *strongly* adaptive adversaries that could, after corrupting a party, observe what message that party attempted to send during that round and then replace its message with another one (or simply delete it).

MPC Model. One part of our parallel broadcast protocol will be modeled using an ideal functionality $\mathcal{F}_{\mathsf{prop}}$ (See Figure 4.) To show that a specific implementation realizes $\mathcal{F}_{\mathsf{prop}}$, we will be using the definition of synchronous MPC in the standalone model by Canetti [4] which we briefly recall here. Let f be a (possibly randomized) function that takes n inputs. We denote the input of party i as x_i. All parties also hold some common auxiliary input z. The goal of running a protocol Π is for all honest parties to learn outputs $[y_1, \ldots, y_n] \leftarrow f(x_1, \ldots, x_n)$. During the execution, the adversary $\mathcal{A}$ can corrupt any party adaptively, upon which it learns the internal state of this party. The adversary $\mathcal{A}$ outputs its entire view. We write $\mathbf{Real}_{\Pi,\mathcal{A}}(1^\kappa, \mathbf{x}, z)$ to denote the distribution of the adversary's view that results from the above experiment.

We define security of Π relative to an ideal world where a trusted party securely computes f and outputs the result to all parties. As before, parties hold input vector $\mathbf{x}$; the adversary in the ideal world is denoted as $\mathcal{S}$. The ideal-world execution now works as follows.

Initial corruptions. $\mathcal{S}$ adaptively corrupts parties and learns their inputs.

Evaluation by trusted party. The inputs $\mathbf{x}$ of honest parties are sent to the trusted party. The ideal-world adversary $\mathcal{S}$ can specify the inputs on behalf of any of the parties it has corrupted. The trusted party evaluates the function f and returns the computed outputs y_i to the respective party i.
Additional corruptions. At any point in time (after output has been provided by the trusted party), $\mathcal{S}$ may adaptively corrupt additional parties i.
Output. The honest parties output their view.
Post-execution corruptions. $\mathcal{S}$ may corrupt additional parties and output (any function of) its view.

We write $\mathbf{Ideal}_{f,\mathcal{S}}(1^\kappa, \mathbf{x}, z)$ to denote the distribution of $\mathcal{S}$ that results from the above experiment.

Definition 3 (Secure Computation.). *Π is said to t-securely compute f if for all PPT adversaries $\mathcal{A}$ that corrupt at most t parties, there exists a PPT simulator $\mathcal{S}$ such that*

$$\{\mathbf{Real}_{\Pi,\mathcal{A}}(1^\kappa, \mathbf{x}, z)\}_{\kappa\in\mathbb{N},\mathbf{x},z\in\{0,1\}^*} \approx \{\mathbf{Ideal}_{f,\mathcal{S}}(1^\kappa, \mathbf{x}, z)\}_{\kappa\in\mathbb{N},\mathbf{x},z\in\{0,1\}^*}.$$

3 Single-Sender Broadcast

We are now ready to describe our proposed communication-efficient BC protocol in detail. As we mentioned in the introduction, our protocol replaces Dolev-Strong's SEND-ALL instruction with a SEND-RANDOM instruction. We model SEND-RANDOM with a randomized procedure that we call AddRandomEdges (see Figure 1) that simulates the propagation of messages from honest nodes to the rest of the network in our protocol, between two consecutive rounds. Separately analyzing it allows us to argue about consistency and validity of our protocol RandomBroadcast in a structured manner.

3.1 The Procedure AddRandomEdges

AddRandomEdges works over a graph G whose n vertices V are partitioned into three disjoint sets and which is initially empty, i.e., has no edges. Given an arbitrary partition of V into three disjoint sets S_1, S_2, S_3, and a set $S \subseteq S_1$, AddRandomEdges adds the edge (v, u) to the graph G with probability m/n for every pair of nodes $v \in S$ and $u \in V$. Note that S_1 is fully defined by S_2, S_3 and V as $S_1 = V - (S_2 \cup S_3)$, therefore S_1 is not an input to AddRandomEdges.

The procedure outputs the resulting graph G (i.e., with all the added edges). Looking ahead, S will represent the set of parties that send a message q at a specific round r and S_2 will represent the set of parties that have not received q in a previous round. An edge from $v \in S$ to $u \in V$ represents that party v sends q to party u in round r. Since we are trying to figure out how many parties in S_2 will receive q (for the first time) during round r, we can study the degree of nodes in S_2. We now prove a useful property of AddRandomEdges. First, let us define the following indicator random variables.

```
1: procedure G ← AddRandomEdges(V, S_2, S_3, S, m)
   Input: Set of n nodes V; disjoint sets S_2, S_3 both subsets of V; S ⊆ V − (S_2 ∪ S_3);
   Integer m ≤ n.
   Output: A graph G.

2:     Let G be an empty graph with node set V;
3:     for every node v ∈ S do
4:         for every node u ∈ V do
5:             Add an edge (v, u) to G with probability m/n;
6:     return G;
```

Fig. 1. The AddRandomEdges procedure.

Definition 4. *Let $G \leftarrow \text{AddRandomEdges}(V, S_2, S_3, S, m)$. For all $u \in S_2$ let $Z_u \in \{0, 1\}$ such that $Z_u = 1$ if and only if u has nonzero degree in G.*

In Lemma 1 we show that the number of nodes in S_2 that acquire an edge in G is at least twice the number of nodes in S. Intuitively, this will allow us to show that messages propagate very quickly in our broadcast protocol.

Lemma 1. *Let (S_1, S_2, S_3) be a partition of n nodes into disjoint sets with $\tau = |S_1| \leq \epsilon \cdot n/3$, $|S_2| = \epsilon \cdot n - |S_1|$, $|S_3| = n - \epsilon \cdot n$, where $\epsilon \in (0, 1)$ is a constant. Let also $S \subseteq S_1$ with $|S| \geq 2 \cdot \tau/3$ and let $\{Z_u\}_{u \in S_2}$ be the random variables defined by $\text{AddRandomEdges}(V, S_2, S_3, S, m)$ per Definition 4. Then for $m \geq 15/\epsilon$,*

$$\Pr\left[\sum_{u \in S_2} Z_u \geq 2 \cdot \tau\right] \geq 1 - p, \text{ where } p = \max\left\{\epsilon \cdot n \cdot e^{-\epsilon \cdot m/9}, \left(\frac{e}{2}\right)^{-\epsilon \cdot m/4}\right\}.$$

Proof. Denote E the set of edges in G (note that E is a random variable). First, note that by definition of the propagation process, the random variables Z_u, $(u \in S_2)$ are independent and identically distributed with

$$\Pr[Z_u = 0] = \Pr[\forall v \in S : (v, u) \notin E] = \prod_{v \in S} \Pr[(v, u) \notin E] = (1 - m/n)^{|S|}.$$

The proof will consider two cases, one for $|S_1| > \epsilon \cdot n/6$ and one for $|S_1| \leq \epsilon \cdot n/6$.

Case $|S_1| > \epsilon \cdot n/6$: For this case, we have that $|S| \geq 2 \cdot \tau/3 = 2 \cdot |S_1|/3 > \epsilon \cdot n/9$ and therefore, according to the random process of AddRandomEdges, the probability that $Z_u = 0$ for a fixed $u \in S_2$ is

$$\Pr[Z_u = 0] = (1 - m/n)^{|S|} < (1 - m/n)^{\epsilon \cdot n/9} \leq e^{-\epsilon \cdot m/9},$$

where the last step is due to $1+x \le e^x$. Since $|S_2| = \epsilon \cdot n - |S_1| \ge 2 \cdot \epsilon \cdot n/3 \ge 2 \cdot \tau$,

$$\Pr\left[\sum_{u \in S_2} Z_u \ge 2 \cdot \tau\right] \ge \Pr\left[\sum_{u \in S_2} Z_u = |S_2|\right] = \Pr\left[\bigcap_{u \in S_2} Z_u = 1\right]$$
$$= 1 - \Pr\left[\bigcup_{u \in S_2} Z_u = 0\right] \ge 1 - (\epsilon \cdot n - |S_1|) \cdot e^{-\epsilon \cdot m/9} > 1 - \epsilon \cdot n \cdot e^{-\epsilon \cdot m/9}\,.$$

Case $|S_1| \le \epsilon \cdot n/6$:
Since variables Z_u are iid, we can use the lower tail Chernoff bound (see Appendix) for $Z = \sum_{u \in S_2} Z_u$, i.e.,

$$\Pr[Z < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}\,.$$

Note that $\mu = \mathbb{E}[Z] = \sum_{u \in S_2} \Pr[Z_u = 1] = (\epsilon \cdot n - |S_1|)\left(1 - (1 - m/n)^{|S|}\right)$. Therefore

$$\begin{aligned}\mu &\ge (5/6) \cdot \epsilon \cdot n \cdot \left(1 - (1-m/n)^{|S|}\right) \text{ (since } |S_1| \le \epsilon \cdot n/6) \\ &\ge (5/6) \cdot \epsilon \cdot n \cdot (1 - (1-m/n)) \text{ (since } |S| \ge 1) \\ &= (5/6) \cdot \epsilon \cdot n \cdot \frac{m}{n} = (5/6) \cdot \epsilon \cdot m > 0.5 \cdot \epsilon \cdot m\,.\end{aligned}$$

For $\delta = 1/2$ this yields

$$\Pr[Z < \mu/2] < \left(\frac{e^{-0.5}}{(0.5)^{0.5}}\right)^{0.5 \cdot \epsilon \cdot m} = \left(\frac{e}{2}\right)^{-\epsilon \cdot m/4}\,.$$

Recall that we want to bound the probability $\Pr[Z < 2 \cdot \tau]$. Therefore it is enough to show that $\mu \ge 4 \cdot \tau$. Indeed, starting with the expression $\mu = (\epsilon \cdot n - |S_1|)\left(1 - (1 - \frac{m}{n})^{|S|}\right)$ we have that

$$\begin{aligned}&\mu \ge (5/6) \cdot \epsilon \cdot n \cdot \left(1 - e^{-\frac{m}{n}|S|}\right) \text{ (by } 1 + x \le e^x) \\ &\ge 5 \cdot |S_1| \cdot \left(1 - e^{-\frac{m \cdot \epsilon}{6|S_1|}|S|}\right) \text{ (since } n \ge \frac{6|S_1|}{\epsilon} \text{ and for } \alpha > 0,\ n \cdot \left(1 - e^{-\frac{\alpha}{n}}\right) \nearrow \text{ in } n) \\ &\ge 5 \cdot \tau \cdot \left(1 - e^{-\frac{m \cdot \epsilon}{6\tau} 2 \cdot \tau/3}\right) \text{ (since } |S_1| = \tau \text{ and } |S| \ge 2 \cdot \tau/3) \\ &= 5 \cdot \tau \cdot \left(1 - e^{-m \cdot \epsilon/9}\right) > 4 \cdot \tau \text{ (since } m \ge 15/\epsilon)\,.\end{aligned}$$

Therefore the statement holds with the stated probability. □

3.2 The Protocol RandomBroadcast

We now describe our protocol RandomBroadcast. For simplicity, we will describe our protocol for the case where values agreed upon are from the binary

domain, but we discuss how to generalize to an arbitrary domain of values in Section 6.

Intuition: From SEND-ALL to Gossiping. As we mentioned in the introduction, our protocol is inspired by the protocol of Dolev and Strong [7] that achieves $O(n^3 \cdot \kappa)$ communication complexity in the bulletin board PKI model. We give a detailed description of the Dolev-Strong protocol here. Each party $i \in [n]$ maintains a set $\mathsf{Extracted}_i$ that is initialized as empty. The protocol proceeds in $t+1$ rounds as follows (Again, $t < n$ is the number of corrupted parties.) In the first round, the designated sender s signs her input bit and sends the signature to all $n-1$ parties. In rounds $2 \leq r \leq t$, for each bit $b \in \{0,1\}$, if an honest party i has seen at least r signatures on b (including a signature from the designated sender) and b is not in her extracted set, then party i adds b to her local extracted set, signs b and sends the $r+1$ signatures to all $n-1$ parties. In the final round $t+1$, i accepts a bit b iff b is the only bit in $\mathsf{Extracted}_i$.

What makes the above protocol work is the fact that when party i sends the $r+1$ signatures, *all* honest parties see these signatures in the next round, since these signatures are sent to *all* other $n-1$ parties. In the final round, it is not necessary to send again, since holding $t+1$ signatures on b means that at least one honest party has sent $r+1$ signatures upon receiving r signatures on b in round $r < t+1$. Hence, all parties must have received these $r+1$ signatures in round $r+1$ and added b to their extracted sets in that round. In terms of communication complexity, note that all honest parties send an $O(t \cdot \kappa)$-sized message to $n-1$ parties (κ is due to the size of the signature), which results in $O(n^2 \cdot t \cdot \kappa)$ communication. Given that $t = O(n)$, this is $O(n^3 \cdot \kappa)$.

Our protocol does away with the SEND-ALL instructions, and introduces a form of gossiping: it does not require an honest party to send the $r+1$ signatures to all $n-1$ parties. Instead, an honest party sends the $r+1$ signatures to each other party with probability $\frac{m}{n}$. The hope is that after a certain number of rounds, enough honest parties will see these messages. As expected, the total number of rounds must now increase. Fortunately, our protocol requires just an additional $R = O(\log n)$ rounds, yielding a communication complexity of $O(n^2 \cdot m \cdot \kappa) = O(n^2 \cdot \kappa^2)$ and a round complexity of $O(n)$.

Formal Description and Proof of RandomBroadcast. Figure 2 contains the pseudocode of our protocol from the view of an honest party p (i.e., this is the algorithm that runs at each distributed (honest) node p). It takes as input an initial bit b in the case of the designated sender (no input else) and returns the final bit b'. Note three major differences from the Dolev-Strong protocol [7]: (1) we increase the number of rounds from t to $t+R+1$ (Line 3); (2) instead of sending to all parties we send to each party randomly with probability m/n (Lines 4 and 12); (3) for all rounds $r \geq t+1$ we do not require $r+1$ signatures to add to the extracted set but just $t+1$—that is why we use the expression $\min\{r, t+1\}$ in Line 10. We now continue with the proof of consistency and validity of RandomBroadcast. We first define, using notation consistent with AddRandomEdges, the following sets of parties (wrt a bit b and a round r):

1. $S(b,r)$: honest parties i that added b to their $\mathsf{Extracted}_i$ set *at* round r;

```
1: procedure b' ← RandomBroadcast_p()
       Input: A bit b if p = s, no input otherwise.
       Output: Decision bit b'.

2:     Extracted_p = Local_p = ∅;
3:     if p is the designated sender s then
4:         Send(sig_s(b), [n]);
5:     for round r = 2 to t + R do
6:         Local_p ← Local_p ∪ Receive();
7:         for bit x ∈ {0, 1} do
8:             S ← DistinctSigs(x, Local_p, s);
9:             if |S| ≥ min{r, t + 1} ∧ x ∉ Extracted_p then
10:                Extracted_p = Extracted_p ∪ x;
11:                for party i = 1 to n do
12:                    Send(sig_p(x) ∪ S, i) with probability m/n;
13:    return b' ∈ Extracted_p if |Extracted_p| = 1, otherwise return canonical bit 0;
```

Fig. 2. Our RandomBroadcast$_p$ protocol for party p. DistinctSigs(x, Local_p, s) returns the set of valid signatures from distinct signers on x contained in Local_p if this set includes a signature from s, otherwise it returns $\emptyset$. Note that only the designated sender receives an input bit in the protocol.

2. $S_1(b, r)$: honest parties i that added b to their $\mathsf{Extracted}_i$ set *by* round r;
3. $S_2(b, r)$: honest parties i that have *not* added b to their $\mathsf{Extracted}_i$ set by round r.

Also, define S_3 contain the set of malicious parties ($|S_3| = n - \epsilon \cdot n$). We now prove our main technical lemma showing (roughly) that the number of parties that receive a message at round r' that was sent at round $r < r'$ increases exponentially with $r' - r$ with overwhelming probability.

Lemma 2 (Gossiping bounds). *For a specific bit b, let r be the first round of* RandomBroadcast *where an honest party i adds b to $\mathsf{Extracted}_i$. Let $R = \lceil \log_3(\epsilon \cdot n) \rceil$. Let p be the probability defined in Lemma 1. Then:*

1. *For all rounds ρ such that $r \leq \rho \leq r + R$ and $|S_1(b, \rho - 1)| \leq \epsilon \cdot n/3$ we have*

$$|S(b, \rho)| \geq (2/3) \cdot |S_1(b, \rho)| \text{ and } |S_1(b, \rho)| \geq 3^{\rho - r}$$

   with probability at least $(1 - p)^{\rho - r}$.
2. *Let $r^* > r$ be a round such that $|S_1(b, r^* - 1)| > \epsilon \cdot n/3$. Then*

$$|S_1(b, r^*)| = \epsilon \cdot n$$

   with probability at least $(1 - p^*) \cdot (1 - p)^{r^* - r - 1}$, where $p^* = \epsilon \cdot n \cdot e^{-2\epsilon \cdot m/9}$.

Proof. We first prove (1) by induction on ρ. For the base case where $\rho = r$, we have that by definition of $S(b, r + 1)$ and $S_1(b, r + 1)$, $i \in S(b, r + 1)$ and $i \in S_1(b, r + 1)$. Therefore $|S(b, \rho)| = |S(b, r)| \geq 1$ and $|S_1(b, \rho)| = |S_1(b, r)| \geq 1$,

with probability 1, and the base case holds. For the inductive step, assume the claim holds for some round $\rho \leq r+R-1$, i.e., with probability at least $(1-p)^{\rho-r}$:

$$|S(b,\rho)| \geq (2/3)|S_1(b,\rho)| \text{ and } |S_1(b,\rho)| \geq 3^{\rho-r}\,. \tag{1}$$

Recall now that the protocol proceeds from round ρ to round $\rho+1$ by having parties in $S(b,\rho)$ send a valid message on b to party i ($i \in [n]$) with probability m/n. We define the events $A : |S(b,\rho+1)| \geq 2 \cdot |S_1(b,\rho)|$ and

$$B : |S(b,\rho)| \geq (2/3)|S_1(b,\rho)| \text{ and } |S_1(b,\rho)| \geq 3^{\rho-r}.$$

To figure out a bound on how many new honest parties will receive this message in the next round (which is the set $S(b,\rho+1)$), and since $|S_1(b,\rho)| \leq \epsilon \cdot n/3$ by hypothesis, it is enough to call AddRandomEdges$(V, S_2(b,\rho), S_3, S(b,\rho), m)$ from Figure 1. By applying Lemma 1 for $\tau = S_1(b,\rho)$, we get that, assuming $m \geq 15/\epsilon$, $\Pr[A|B] \geq 1-p$, and thus $\Pr[|S(b,\rho+1)| \geq 2 \cdot |S_1(b,\rho)||B] \geq 1-p$. By the inductive hypothesis in Equation 1 we know that $\Pr[B] \geq (1-p)^{\rho-r}$, and therefore by the identity $\Pr[A] \geq \Pr[B] \cdot \Pr[A|B]$ we have that

$$|S(b,\rho+1)| \geq 2 \cdot |S_1(b,\rho)|\,, \tag{2}$$

with probability at least $(1-p)^{\rho-r} \cdot (1-p) = (1-p)^{\rho+1-r}$. Now by Equation 2 we have $|S(b,\rho+1)| \geq 2 \cdot |S_1(b,\rho)| \geq (2/3) \cdot |S_1(b,\rho+1)|$, as required. The last part is because, by definition of $S_1()$ and $S()$ we have

$$|S_1(b,\rho+1)| = |S(b,\rho+1)| + |S_1(b,\rho)|\,.$$

Finally, by Equation 2, the above can be written again as

$$|S_1(b,\rho+1)| = |S(b,\rho+1)| + |S_1(b,\rho)| \geq 3 \cdot |S_1(b,\rho)| \geq 3 \cdot 3^{\rho-r} = 3^{(\rho+1)-r}\,,$$

as required.

For (2), take r^* to be the first round with $|S_1(b,r^*-1)| > \epsilon \cdot n/3$. It has to be the case that $|S_1(b,r^*-2)| \leq \epsilon \cdot n/3$. Thus, we can apply **Item (1)** for round r^*-1 and we have that $|S(b,r^*-1)| \geq (2/3) \cdot |S_1(b,r^*-1)| > (2/3) \cdot \epsilon \cdot n/3$, with probability at least $(1-p)^{r^*-r-1}$, where $p = \epsilon \cdot n \cdot e^{-\epsilon \cdot m/9}$. The set $S(b,r^*-1)$ contains all honest parties i who added b to $\mathsf{Extracted}_i$ at round r^*-1. These are all the parties who will propagate b during round r^*. For each honest party $1,\ldots,\epsilon \cdot n$ (without loss of generality, we assume $\epsilon \cdot n$ is an integer) we define an indicator random variable $Z_i = 1$, if party i adds b to $\mathsf{Extracted}_i$ at round r^*, i.e., if i receives a valid message of b within this round and $Z_i = 0$ otherwise. We want to bound the probability $\Pr\left[\bigcup_{i=1}^{\epsilon \cdot n}\{Z_i = 0\}\right]$, since this is the probability with which at least one honest party i will not add b to $\mathsf{Extracted}_i$ at round r^*. From the union bound of this probability we get that

$$\Pr\left[\bigcup_{i=1}^{\epsilon \cdot n}\{Z_i = 0\}\right] \leq \sum_{i=1}^{\epsilon \cdot n} \Pr[Z_i = 0] = \epsilon \cdot n \cdot \left(1 - \frac{m}{n}\right)^{|S(b,r^*-1)|}$$

$$\leq \epsilon \cdot n \cdot \left(1 - \frac{m}{n}\right)^{(2/3)\cdot\epsilon\cdot n/3} = \epsilon \cdot n \cdot \left(\left(1 - \frac{m}{n}\right)^{\frac{n}{m}}\right)^{(2/3)\cdot\epsilon\cdot m/3} \leq \epsilon \cdot n \cdot e^{-2\epsilon\cdot m/9}\,.$$

Therefore, all honest parties will receive b at round r^*, i.e. $|S_1(b, r^*)| = \epsilon \cdot n$, with probability at least $(1 - \epsilon \cdot n \cdot e^{-2\epsilon \cdot m/9}) \cdot (1-p)^{r^*-r-1}$, as required. For any round after r^*, the same holds with the same probability and it holds with certainty, conditioned that it holds for r^*. □

Lemma 3 (t-consistency of RandomBroadcast). *Let $R = \lceil \log_3(\epsilon \cdot n) \rceil$ and $m = 15/\epsilon + \kappa$.* RandomBroadcast *satisfies t-consistency, per Definition 1, with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Suppose an honest party i adds bit b to $\mathsf{Extracted}_i$ at some round r. We prove by the end of the protocol all honest parties j will add b to their $\mathsf{Extracted}_j$ sets with probability $1 - \mathsf{negl}(\kappa)$—this will mean that all honest parties will have identical $\mathsf{Extracted}$ sets by the end of the protocol, which is equivalent to consistency. We distinguish the two cases:
Case $r < t + 1$: We distinguish two cases. If $S_1(b, r) > \epsilon \cdot n/3$, then by Item (2) of Lemma 2, all $\epsilon \cdot n$ honest parties will add bit b in their extracted set by the next round with probability at least $1 - \epsilon \cdot n \cdot e^{-2\epsilon \cdot m/9} = 1 - \mathsf{negl}(\kappa)$, since $n = poly(\kappa)$. Now, if $S_1(b, r) \leq \epsilon \cdot n/3$, let $\mathbf{r}$ be the round $r + R - 1$. If $S_1(b, \mathbf{r}) > \epsilon \cdot n/3$, the previous case applies. Otherwise, Item (1) of Lemma 2 applies, saying that at round $\mathbf{r} + 1 = r + R$,

$$S_1(b, r + R) \geq 3^R = 3^{\lceil \log_3(\epsilon \cdot n) \rceil} \geq 3^{\log_3(\epsilon \cdot n)} = \epsilon \cdot n$$

with probability at least $(1-p)^R \geq 1 - R \cdot p$, by Bernoulli's inequality[4] and since $-p \geq -1, R \geq 1$. Note however that for $m = 15/\epsilon + \kappa$ we get that $R \cdot p$ is $\mathsf{negl}(k)$, again since $n = poly(\kappa)$.
Case $r \geq t + 1$: Suppose an honest party i adds bit b to $\mathsf{Extracted}_i$ at some round $r \geq t+1$. This means that i has received valid signatures on b from $t+1$ distinct parties. Thus an honest party j added bit b to $\mathsf{Extracted}_j$ at some round $r'' < t+1$. So, the case $r'' < t + 1$ from above applies for honest party j and therefore all honest parties will add b to their $\mathsf{Extracted}$ sets by the end of the protocol, with probability $1 - \mathsf{negl}(\kappa)$. □

Lemma 4 (t-validity of RandomBroadcast). *Let $R = \lceil \log_3(\epsilon \cdot n) \rceil$ and $m = 15/\epsilon + \kappa$.* RandomBroadcast *satisfies t-validity, per Definition 1, with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Follows from the proof of consistency in Theorem 3. After $R = \lceil \log_3(\epsilon \cdot n) \rceil$ rounds, all honest parties will have received the bit of the honest sender, with probability $1 - \mathsf{negl}(\kappa)$. □

Theorem 1 (t-security of RandomBroadcast). *The protocol* RandomBroadcast *is a t-secure broadcast protocol according to Definition 1 and against a static adversary.*

[4] For every $x, r \in \Re, x \geq -1, r \geq 1$ it holds that $(1 + x)^r \geq 1 + rx$.

Proof. The protocol RandomBroadcast runs for a deterministic and fixed number of rounds, equal to $t + R + 1$, thus every party terminates with a value (which could be the canonical 0 bit in cases). As proven in Lemmata 3, 4, the protocol achieves consistency and validity for any $t = (1 - \epsilon) \cdot n < n$. Therefore, the protocol satisfies Definition 1. □

Theorem 2 (Communication complexity of RandomBroadcast). *Let $R = \lceil \log_3(\epsilon \cdot n) \rceil$ and $m = 15/\epsilon + \kappa$. The total number of bits exchanged by all parties in* RandomBroadcast *is $\tilde{O}(n^2 \cdot \kappa^2)$.*

Proof. Every honest party sends at most one time to $m = 15/\epsilon + \kappa$ parties a message of at most t signatures. Since there are $O(n)$ honest parties, $t = O(n)$ and the size of each signature is κ, the total number of bits exchanged is $O(n^2 \cdot m \cdot \kappa) = \tilde{O}(n^2 \cdot \kappa^2)$. □

4 ConvergeRandom Protocol

In this section we introduce the Converge problem, an efficient solution of which will be used as a black box in our parallel broadcast protocol in Section 5. Informally, in the Converge problem, honest parties begin with individual subsets (of a fixed set $\mathcal{M}$) as inputs and in the end of the protocol all honest parties must have a superset of the union of all the initial honest-owned subsets. We want to design Converge protocols in the presence of an adversary that can corrupt at most t parties, adaptively. We now define the Converge problem formally.

Definition 5 (t-secure Converge protocol). *Let $\mathcal{M}$ be a fixed set of messages. A protocol Π executed by n parties, where every honest party p initially holds a set $M_p \subseteq \mathcal{M}$, is a* t-secure Converge protocol *if all honest parties, upon termination, output a set*

$$S_p \supseteq \bigcup_{p \in \mathcal{H}} M_p \,,$$

whenever at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of the protocol, with probability $1 - \mathsf{negl}(\kappa)$.

We note that there is a very simple t-secure Converge protocol: All honest parties just send their local sets to all other parties, in one round. Unfortunately, since every local set has size at most $|\mathcal{M}|$, the communication complexity of such a protocol is $O(n^2 \cdot |\mathcal{M})|$. In this section we propose a protocol, ConvergeRandom (see Figure 6) that uses gossiping to reduce communication to $\tilde{O}(n \cdot |\mathcal{M}|)$. Before we present our protocol, we will be introducing two necessary tools that will help with the analysis and clear exposition respectively, the procedure AddRandomEdgesAdaptive and the functionality $\mathcal{F}_{\mathsf{prop}}$.

4.1 The Procedure AddRandomEdgesAdaptive

Recall that central in the analysis of our single-sender broadcast result was the standalone randomized procedure AddRandomEdges that was presented in Section 3.1. Roughly speaking, AddRandomEdges was adding edges from a set S of nodes (i.e., senders of a message x) to other nodes (i.e., recipients of message x) of the graph. Among recipients, some nodes were malicious and we were interested in computing the number of honest nodes with a nonzero degree (i.e., those receiving the message). Crucially for our analysis, the set of malicious nodes in AddRandomEdges was *fixed* before the edges were drawn.

Here we will be working with an adaptive adversary, and we will introduce the respective standalone randomized procedure which we call AddRandomEdgesAdaptive—see Figure 3. Analyzing AddRandomEdgesAdaptive will help us with the proof of our ConvergeRandom procotol later.

In AddRandomEdgesAdaptive, a set of honest senders $S \subseteq H$ (H is the set of initial honest nodes) are sending a message x using AddRandomEdges, as in Section 3.1. However, the set of malicious nodes are not fixed a priori (as in AddRandomEdges) and the adversary $\mathcal{A}$ decides who to corrupt *after* the edges have been placed. Our goal is to prove that after the adversary has finished with the adaptive corruptions, still a good amount (in particular $2 \cdot |S|$) among the remaining honest nodes (i.e., nodes in H') will be receiving message x with overwhelming probability. Clearly if we do not confine the view of the adversary, we cannot prove something meaningful since the adversary can go ahead and corrupt exactly the nodes that received x. For this reason, in AddRandomEdgesAdaptive we strategically allow the adversary to access just the list RevealedEdges (see Line 5) before making the next corruption—these are the edges that correspond to nodes that *have already been corrupted* (Note that this restriction of the adversary will be enforced by the implementation of our protocol later.) We now formalize this intuition.

Definition 6. *Let* $(G, H') \leftarrow \text{AddRandomEdgesAdaptive}_{\mathcal{A}}(V, S, H, m)$. *For all* $u \in H$ *define random variables* $Z_u \in \{0, 1\}$ *such that* $Z_u = 1$ *if and only if* u *has nonzero degree in* G *and* $u \in H'$.

Lemma 5. *Let* $(G, H') \leftarrow \text{AddRandomEdgesAdaptive}_{\mathcal{A}}(V, S, H, m)$. *Fix* $\epsilon \in (0, 1)$. *Then for* $m \geq 10/\epsilon$ *and* $|S| \leq \epsilon \cdot n/2$,

$$\Pr\left[\sum_{u \in H} Z_u \geq 2|S|\right] \geq 1 - p\,,$$

where $p = \max\{n \cdot e^{-\epsilon \cdot m/5}, (\frac{e}{2})^{-\epsilon \cdot m/2}\}$.

Proof. For any set H^*, define E_{H^*} to be the event that AddRandomEdgesAdaptive$_{\mathcal{A}}(V, S, H, m)$ outputs $(\cdot, H^*)$. Also, $\text{supp}(\mathsf{E}) = \{H' : \Pr[\mathsf{E}_{H'}] > 0\}$. Fix $H^* \in \text{supp}(\mathsf{E})$ such that

$$\Pr\left[\sum_{u \in H} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H'}\right] \geq \Pr\left[\sum_{u \in H} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H^*}\right],$$

```
1: procedure (G, H') ← AddRandomEdgesAdaptive_A(V, S, H, m)
   Input: Set of n nodes V; sets S and H with S ⊆ H ⊆ V; Integer m ≤ n.
   Output: A graph G a set of nodes H' ⊆ H.

2:     G ← AddRandomEdges(V, S, m);
3:     Initialize RevealedEdges to be all edges incident to nodes in v ∈ V − H;
4:     while |H| ≥ ε · n do
5:         v_i ← A(V, H, RevealedEdges) such that v_i ∈ H;
6:         if v_i ≠ null then
7:             Add {(u, v_i) : u ∈ neighbor(v_i)} to RevealedEdges;
8:             H = H − v_i;
9:         else
10:            break;
11:    Set H' = H;
12:    return (G, H');
```

Fig. 3. Experiment AddRandomEdgesAdaptive.

for all $H' \in \mathrm{supp}(\mathsf{E})$. Therefore

$$\Pr\left[\sum_{u\in H} Z_u \geq 2|S|\right] = \sum_{H'\in\mathrm{supp}(\mathsf{E})} \Pr\left[\sum_{u\in H} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H'}\right] \Pr[\mathsf{E}_{H'}]$$

$$\geq \sum_{H'\in\mathrm{supp}(\mathsf{E})} \Pr\left[\sum_{u\in H} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H^*}\right] \Pr[\mathsf{E}_{H'}]$$

$$= \Pr\left[\sum_{u\in H} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H^*}\right] \geq \Pr\left[\sum_{u\in H^*} Z_u \geq 2|S| \;\middle|\; \mathsf{E}_{H^*}\right],$$

since $H^* \subseteq H$. The remaining proof lower-bounds $\Pr[\sum_{u\in H^*} Z_u \geq 2|S| \mid \mathsf{E}_{H^*}]$. This will be done in three steps.

Step 1: Computing the probabilities $\Pr[Z_u = 1 \mid \mathsf{E}_{H^*}]$ *for all* $u \in H^*$. For $u \in H^*$ let Y_u be a random variable such that $Y_u = 1$ iff u has nonzero degree in G. By definition of Z_u ($Z_u = (Y_u = 1) \cap (u \in H')$) we have that, for all $u \in H^*$,

$$\Pr[Z_u = 1 \mid \mathsf{E}_{H^*}] = \Pr[Y_u = 1 \mid \mathsf{E}_{H^*}].$$

Suppose now $H^* = H - \{v_1, \ldots, v_\ell\}$, where $v_1, \ldots, v_\ell$ are nodes chosen by the adversary in Line 5 of the experiment. Because the adversary, in his picking $v_1, \ldots, v_\ell$, never accesses information related to nodes in H^* (his access is confined to information in $V - H^*$ through the RevealedEdges list), it follows that E_{H^*} (the event of the adversary picking $v_1, \ldots, v_\ell$) and Y_u (for every $u \in H^*$) are independent events. Therefore for all $u \in H^*$ it is

$$\Pr[Z_u = 1 \mid \mathsf{E}_{H^*}] = \Pr[Y_u = 1 \mid \mathsf{E}_{H^*}] = \Pr[Y_u = 1] = 1 - (1 - m/n)^{|S|}.$$

Step 2: Showing $\{Z_u\}_{u\in H^*}$, conditioned on E_{H^*}, are independent. By using the above findings and by the independence of Y_u we have that for all $b_u \in \{0,1\}$

$$\Pr\left[\left(\bigcap_{u\in H^*} Z_u = b_u\right) \mid \mathsf{E}_{H^*}\right] = \Pr\left[\left(\bigcap_{u\in H^*} Y_u = b_u\right) \mid \mathsf{E}_{H^*}\right]$$
$$= \Pr\left[\bigcap_{u\in H^*} Y_u = b_u\right] = \prod_{u\in H^*} \Pr[Y_u = b_u] = \prod_{u\in H^*} \Pr[Z_u = b_u \mid \mathsf{E}_{H^*}]$$

and therefore $\{Z_u\}_{u\in H^*}$ are independent conditioned on E_{H^*}.
Step 3: Applying a Chernoff bound. We consider two cases, one for $|S| > \epsilon \cdot n/5$ and one for $|S| \leq \epsilon \cdot n/5$. For $|S| > \epsilon \cdot n/5$, we have that for $u \in H^*$

$$\Pr[Z_u = 0 \mid \mathsf{E}_{H^*}] = (1 - m/n)^{|S|} < (1 - m/n)^{\epsilon\cdot n/5} \leq e^{-\epsilon\cdot m/5}\,.$$

Note now that since $|H^*| \geq \epsilon \cdot n$ and $|S| \leq \epsilon \cdot n/2$ it is $|H^*| \geq 2 \cdot |S|$. Therefore

$$\Pr\left[\sum_{u\in H^*} Z_u \geq 2\cdot|S| \mid \mathsf{E}_{H^*}\right] \geq \Pr\left[\sum_{u\in H^*} Z_u = |H^*| \mid \mathsf{E}_{H^*}\right]$$
$$= \Pr\left[\bigcap_{u\in H^*} Z_u = 1 \mid \mathsf{E}_{H^*}\right] = 1 - \Pr\left[\bigcup_{u\in H^*} Z_u = 0 \mid \mathsf{E}_{H^*}\right]$$
$$\geq 1 - \sum_{u\in H^*} \Pr\left[Z_u = 0 \mid \mathsf{E}_{H^*}\right] = 1 - |H^*|\cdot e^{-\epsilon\cdot m/5}$$
$$> 1 - n\cdot e^{-\epsilon\cdot m/5}, \text{ since } |H^*| < n\,.$$

For the case $|S| \leq \epsilon \cdot n/5$, since Z_u ($u \in H^*$) are independent random variables conditioned on E_{H*} we can use the lower tail Chernoff bound (Inequality 6.2) for $Z = \sum_{u\in H^*} Z_u$, i.e.,

$$\Pr[Z < (1-\delta)\mu \mid \mathsf{E}_{H^*}] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}\,.$$

Now $\mu = \mathbb{E}\left[Z \mid \mathsf{E}_{H^*}\right] = \sum_{u\in H^*} \Pr[Z_u = 1 \mid \mathsf{E}_{H^*}] \geq \epsilon\cdot n\cdot(1-(1-m/n)^{|S|})$ since $|H^*| \geq \epsilon \cdot n$. Now since $|S| \geq 1$ it is

$$\mu \geq \epsilon\cdot n\cdot(1-(1-m/n)^{|S|}) \geq \epsilon\cdot n\cdot(1-(1-m/n)) = \epsilon\cdot m\,.$$

For $\delta = 1/2$ this yields

$$\Pr[Z < \mu/2 \mid \mathsf{E}_{H^*}] < \left(\frac{e^{-0.5}}{(0.5)^{0.5}}\right)^{\epsilon\cdot m} = \left(\frac{e}{2}\right)^{-\epsilon\cdot m/2}\,.$$

Recall however that we must bound the probability $\Pr[Z < 2|S| \mid \mathsf{E}_{H^*}]$. Therefore it is enough to also show that $\mu \geq 4\cdot|S|$. Starting with the expression

Functionality: $\mathcal{F}_{\mathsf{prop}}$

Let n be the number of parties and $m = 10/\epsilon + \kappa$. For every party $i \in [n]$, $\mathcal{F}_{\mathsf{prop}}$ keeps a set O_i which is initialized to $\emptyset$. Let M_i be input messages of party i.

On input $(\mathsf{SendRandom}, M_i)$ by honest party i:
- For all $x \in M_i$ and for all $j \in [n]$ add (i, x) to O_j with prob. m/n;
- **return** M_i to adversary $\mathcal{A}$;
- **return** O_i to party i.

On input $(\mathsf{SendDirect}, \mathbf{x}, J)$ by adversary $\mathcal{A}$ (for a corrupted party i):
- Add $(i, x[j])$ to O_j for all $j \in J$;
- **return** O_i to adversary $\mathcal{A}$.

Fig. 4. Functionality $\mathcal{F}_{\mathsf{prop}}$.

$\mu \geq \epsilon \cdot n \cdot \left(1 - (1 - m/n)^{|S|}\right)$ we have that

$$\begin{aligned}
\mu &\geq \epsilon \cdot n \cdot (1 - e^{-\frac{m}{n}|S|}) \text{ (by } 1 + x \leq e^x) \\
&\geq 5 \cdot |S| \cdot \left(1 - e^{-\frac{m \cdot \epsilon}{5}}\right) \text{ (since } n \geq \frac{5|S|}{\epsilon} \text{ and for } \alpha > 0,\ n\left(1 - e^{-\frac{\alpha}{n}}\right) \nearrow \text{ in } n) \\
&> 4 \cdot |S| \text{ (since } m \geq 10/\epsilon).
\end{aligned}$$

□

4.2 Ideal Functionality $\mathcal{F}_{\mathsf{prop}}$

To facilitate the exposition of our ConvergeRandom protocol, we will be using an ideal functionality $\mathcal{F}_{\mathsf{prop}}$—see Figure 4. In summary, functionality $\mathcal{F}_{\mathsf{prop}}$ enables a party i to send a set of messages M to, on average, m out of n randomly selected parties without leaking *which those parties are* to the adversary. (We stress that each message in M is sent to different parties.) In our protocol $\mathcal{F}_{\mathsf{prop}}$ is called, via $(\mathsf{SendRandom}, M)$, by all honest parties i in the beginning of every round ρ and returns a set O_i, in the end of round ρ, which contains messages sent from other parties to i in the beginning of round ρ[5]. The adversary in $\mathcal{F}_{\mathsf{prop}}$ gets a special interface to the functionality via the instruction $(\mathsf{SendDirect}, \mathbf{x}, J)$ by which it can send messages in $\mathbf{x}$ to parties specified in the vector J directly rather than randomly. We also assume the adversary learns the input set of an honest party to $\mathcal{F}_{\mathsf{prop}}$. Finally the adversary gets access to all the sets O_i for the parties i that have been corrupted. We note here that $\mathcal{F}_{\mathsf{prop}}$ does not maintain state across calls and therefore all $O_i = \emptyset$, in the beginning of every round.

[5] We slightly abuse the term *round* here; in our protocol, a round ρ will actually consist of two synchronous rounds. If ρ were considered, instead, as a synchronous round, $\mathcal{F}_{\mathsf{prop}}$ is assumed to return at the end of (synchronous) round $\rho + 2\Delta$, after all honest parties provide input at (synchronous) round ρ.

Implementing $\mathcal{F}_{\mathsf{prop}}$ with Propagate(). In Figure 5 we define the process that will instantiate $\mathcal{F}_{\mathsf{prop}}$ and give it the fitting name Propagate(). As usual, we describe the protocol Propagate() from the view of a party p. Consistent with the interface of $\mathcal{F}_{\mathsf{prop}}$, Propagate() takes as input a set of messages M_p (that are to be sent out to other parties) and returns a set of messages O_p that were sent to p. In the first step of Propagate(), every party creates a fresh pair of secret and public keys.

Next, note that Propagate() does not send a message $x \in M_p$ directly to party j (with probability m/n) since this would reveal the recipient of x to the adversary. Instead, before sending, it locally computes a list $\mathcal{L}_j$, for every party $j \in [n]$, and adds x to $\mathcal{L}_j$ with probability m/n. All lists $\mathcal{L}_j$ are padded to a maximum bound $\Lambda = 2|\mathcal{M}| \cdot \frac{m}{n}$ and are encrypted using the fresh public keys (Recall that $\mathcal{M}$ is the fixed set of the Converge problem.) Then *the plaintext lists are erased from memory* and only after erasure the encrypted lists are sent out. In the end, the fresh secret key is also erased from memory. Intuitively, security is guaranteed because (i) the communication graph does not reveal anything (irrespectively of who the recipient of x is, the adversary just sees equally-sized encrypted lists sent to all parties); and (ii) even if the adversary adaptively corrupts a party, no information about who that party sent to is revealed (because of erasure)—the only information that is revealed is from whom that party received, which is harmless. We now show that the lists $\mathcal{L}_j$ constructed by Propagate() will overflow with negligible probability.

Lemma 6. *Let $\mathcal{M}$ be the fixed set of the* Converge *problem with $|\mathcal{M}| = \Omega(n)$. The number of elements x added to list $\mathcal{L}_j$ at Line 7 of* Propagate() *is at most Λ with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Fix a recipient party j. For $i = 1, \ldots, |M|$, let X_i be a 0-1 random variable such that $X_i = 1$ iff the i-th message in M is assigned to list $\mathcal{L}_j$. Note that $\Pr[X_i = 1] = m/n$. Let us also define Y_i, for $i = 1, \ldots, |\mathcal{M}| - |M|$ to be a 0-1 random variable such that $\Pr[Y_i = 1] = m/n$. Define $X = \sum_{i=1}^{|M|} X_i$ and $Y = \sum_{i=1}^{|\mathcal{M}|-|M|} Y_i$. Note that $\mathbb{E}[X + Y] = \mu = |\mathcal{M}| \cdot m/n = \Lambda/2$. By Chernoff,

$$\Pr[X > \Lambda] \leq \Pr[X + Y \geq \Lambda] = \Pr[X + Y \geq 2 \cdot \mu] \leq e^{-\mu/3} \leq e^{-c \cdot m} = \mathsf{negl}(\kappa),$$

for some constant c and since $\mu = |\mathcal{M}| \cdot m/n$, $|\mathcal{M}| = \Omega(n)$ and $m = \Theta(\kappa)$. □

Lemma 7. *Let $\mathcal{M}$ be the fixed set of the* Converge *problem with $|\mathcal{M}| = \Omega(n)$. Let s be the number of bits of a message in $\mathcal{M}$. The communication complexity induced by one call* Propagate() *is $O(|\mathcal{M}| \cdot m \cdot s)$.*

Proof. When a party calls Propagate(), it first sends the fresh public key to all parties and then it creates n lists $\mathcal{L}_i$, each of size $\Lambda \cdot s$, by Lemma 6. Thus, the communication complexity of one call of the process is $O(|\mathcal{M}| \cdot m \cdot s)$. □

Security of Propagate(). Below, we prove that Propagate() securely instantiates $\mathcal{F}_{\mathsf{prop}}$. The key property of Propagate() that we leverage is that all

```
1:  procedure PROPAGATE_p(SendRandom, M_p)
    Input: A set of of messages M_p.
    Output: A set of messages O_p.
2:      Set (sk_p, pk_p) ← KeyGen(1^κ);                    ▷ at time 0
3:      SEND(pk_p, [n]);
4:      RECEIVE();                                          ▷ at time Δ
5:      for all x ∈ M_p do
6:          for j = 1 to n do
7:              Add x to list L_j with probability m/n;
8:      for j = 1 to n do
9:          Pad list L_j to maximum size Λ = 2|M| · m/n;
10:         ct_j ← Enc(pk_j, L_j);
11:         Erase L_j from memory;
12:     for j = 1 to n do
13:         SEND(ct_j, j);
14:     C ← RECEIVE();                                      ▷ at time 2Δ
15:     for all ct ∈ C do
16:         Decrypt ct using sk_p and output a list L;
17:         Add L to O_p;
18:     Erase sk_p from memory;
19:     return O_p;
```

Fig. 5. PROPAGATE(), our implementation of $\mathcal{F}_{\mathsf{prop}}$. We note that the secret and public keys that are generated in Line 2 are one-time and are never used again.

parties send the same amount of information to every other party, regardless of their inputs. This trivializes the simulation of honest parties' communication in the protocol, since it is independent of the actual values they input to PROPAGATE().

Lemma 8. *Assume a CPA-secure PKE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. *Then procedure* PROPAGATE() *t-securely computes* $\mathcal{F}_{\mathsf{prop}}$ *according to Definition 3.*

Proof. We describe a simulator $\mathcal{S}$. To simulate an honest party p the simulator first generates $(sk_p, pk_p) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends pk_p to all parties. Then for each $j \in n$, the simulator samples some c_j of size Λ as a ciphertext with respect to pk_j as follows. Let M_p be p's input to $\mathcal{F}_{\mathsf{prop}}$. Then c_j is computed as $\mathsf{Enc}(pk_j, L_j)$, where each message in M_p is added to L_j with probability m/n. Note that due to Lemma 6, the size of L_j will always be less than Λ with overwhelming probability. Then the simulator sends c_j to p_j.

Static corruptions. When corrupted party p sends ciphertext c to honest party p_j, $\mathcal{S}$ attempts to decrypt c using sk_j. If this succeeds, store the resulting plaintext. At the end of the protocol, $\mathcal{S}$ constructs a vector $\mathbf{x}$ of all so obtained plaintexts and stores the intended recipients in J. Then, it inputs $(\mathsf{SendDirect}, \mathbf{x}, J)$ to $\mathcal{F}_{\mathsf{prop}}$ on behalf of party p_i (It inputs nothing if all decryptions fail.)

1: **procedure** $S_p \leftarrow \textsc{ConvergeRandom}_p(M_p, \mathcal{M})$
Input: A set $M_p \subseteq \mathcal{M}$.
Output: A set S_p.

2: **for** round $\rho = 1$ to $\lceil \log(\epsilon \cdot n) \rceil$ **do**
3: $\textsc{Receive}^{\mathcal{F}_{\mathsf{prop}}} \leftarrow \mathcal{F}_{\mathsf{prop}}(\mathsf{SendRandom}, M_p)$;
4: $\mathsf{Local}_p \leftarrow \mathsf{Local}_p \cup \textsc{Receive}^{\mathcal{F}_{\mathsf{prop}}}$;
5: $M_p = \mathsf{Local}_p \cap \mathcal{M}$;
6: **return** Local_p;

Fig. 6. Our $\textsc{ConvergeRandom}_p$ protocol.

Adaptive corruptions. The first time a party can be corrupted is when it sends its first message, in Line 13 (If it is corrupted before, there is no state to simulate.) There are two subcases:

> p **is corrupted before the output step.** Simulation of p's internal state apart from the received ciphertexts is trivial, as p keeps only its own honestly generated ciphertexts as well as its secret key sk (which $\mathcal{S}$ knows) in the protocol prior to sending; everything else, including the plaintext lists, is erased at this point. All the received ciphertexts (prior to corruption) already have the proper distribution.
> p **is corrupted after the output step.** In this case, $\mathcal{S}$ has to ensure that the ciphertexts received by p over the course of the protocol match what it has output (i.e., what it received from $\mathcal{F}_{\mathsf{prop}}$). This, however, follows directly from the CPA security of the public key encryption scheme, because p erases its secret key prior to outputting.

It is easy to verify that $\mathcal{S}$ provides a perfect simulation of $\mathcal{A}$'s view in a real-world execution of the protocol $\textsc{Propagate}()$. This view is also consistent with the outputs of parties which remain honest throughout $\textsc{Propagate}()$, as $\mathcal{S}$ uses the inputs of the honest senders to $\mathcal{F}_{\mathsf{prop}}$ to simulate the internal states of adaptively corrupted parties. □

4.3 Our ConvergeRandom Protocol

We are now ready to present and analyze our ConvergeRandom protocol, depicted in Figure 6, for solving the Converge problem with improved communication complexity. Our protocol proceeds in $\lceil \log(\epsilon \cdot n) \rceil$ rounds where in each round every honest party uses $\mathcal{F}_{\mathsf{prop}}$ to send his local set to a few randomly selected parties (Line 3). To decide what to send in the next round, each honest party takes the union of the received messages (Line 4), and from the resulting set of messages, keeps only messages that *could* have originated by honest parties (Line 5). For example, messages $m \notin \mathcal{M}$, sent by the adversary can be safely discarded.

We now continue with proving t-security. Recall our adversary observes the execution of round ρ and decides who to corrupt next, adaptively. Clearly, it is

important to conceal recipients of messages (as achieved by $\mathcal{F}_{\mathsf{prop}}$) since otherwise the adversary can block the propagation of a certain message $m \in M_p$ for some honest party p and therefore the final set $S_{p'}$ output by some other party p' will not be a superset of $\cup_{p \in \mathcal{H}} M_p$, as required. We now prove the following.

Lemma 9 (Propagation in presence of an adaptive adversary). *Fix an initially-honest party p and a message $\mathsf{m}^* \in M_p$. Let $h_\rho \geq \epsilon \cdot n$ be the number of honest parties that remain after the adversary has performed corruptions for round ρ of* ConvergeRandom. *At the end of round $\rho \leq \lceil \log(\epsilon \cdot n) \rceil$, with probability $1 - \mathsf{negl}(\kappa)$, there are $\geq \min\{2^\rho, h_\rho\}$ honest parties that receive m^*.*

Proof. Recall that in ConvergeRandom, a message m^* at round ρ is propagated in a randomized fashion using $\mathcal{F}_{\mathsf{prop}}$. Therefore the adversary does not see which party receives the message unless this party is already corrupted—this is exactly what is modeled by AddRandomEdgesAdaptive with the use of $\mathsf{RevealedEdges}$. Therefore, since $h_\rho \geq \epsilon \cdot n$, we can compute the number of honest receivers R_ρ of message m in the end of round ρ by (almost) directly applying Lemma 5. Denote with S_ρ the number of senders in the beginning of some round ρ. We use induction. For $\rho = 1$, we have one sender ($S_1 = 1$) and therefore by applying Lemma 5 we have

$$\Pr[R_1 \geq 2|S_1|] = \Pr[R_1 \geq \min\{2, h_1\}] \geq 1 - \mathsf{negl}(m) \geq 1 - \mathsf{negl}(\kappa)\,,$$

since $m \geq \kappa$ and therefore the base case holds. Let us assume the claim holds for round $\rho \leq \lceil \log(\epsilon \cdot n) \rceil - 1$, i.e., the number of honest receivers R_ρ of message m in the end of round ρ is at least $\min\{2^\rho, h_\rho\}$ with overwhelming probability, i.e.,

$$\Pr[R_\rho \geq 2^\rho] = \Pr[R_\rho \geq \min\{2^\rho, h_\rho\}] \geq 1 - \mathsf{negl}(\kappa)\,,$$

We want to prove that $\Pr[R_{\rho+1} \geq \min\{2^{\rho+1}, h_{\rho+1}\}] \geq 1 - \mathsf{negl}(\kappa)$. According to how ConvergeRandom works, all remaining honest receivers in round ρ are senders in round $\rho + 1$ and therefore $S_{\rho+1} \geq R_\rho$. We distinguish two cases.
Case $S_{\rho+1} \leq \epsilon \cdot n/2$: Using the identity $\Pr[x \geq \min\{a, b\}] \geq \Pr[x \geq a]$ we have that $\Pr[R_{\rho+1} \geq \min\{2^{\rho+1}, h_{\rho+1}\}] \geq \Pr[R_{\rho+1} \geq 2^{\rho+1}]$.
From the inductive hypothesis, we know that $\Pr[S_{\rho+1} \geq 2^\rho] \geq \Pr[R_\rho \geq 2^\rho] \geq 1 - \mathsf{negl}(\kappa)$. So, by Lemma 5 and the inductive hypothesis we have that

$$\begin{aligned}
\Pr[R_{\rho+1} \geq \min\{2^{\rho+1}, h_{\rho+1}\}] &\geq \Pr[R_{\rho+1} \geq 2^{\rho+1}] \\
&\geq \Pr[R_{\rho+1} \geq 2 \cdot S_{\rho+1}, S_{\rho+1} \geq 2^\rho] \\
&= \Pr[R_{\rho+1} \geq 2 \cdot S_{\rho+1} \mid S_{\rho+1} \geq 2^\rho] \cdot \Pr[S_{\rho+1} \geq 2^\rho] \\
&\geq (1 - \mathsf{negl}(\kappa)) \cdot (1 - \mathsf{negl}(\kappa)) \\
&= 1 - \mathsf{negl}(\kappa)\,.
\end{aligned}$$

Case $S_{\rho+1} > \epsilon \cdot n/2$: We have $\Pr[R_{\rho+1} \geq \min\{2^{\rho+1}, h_{\rho+1}\}] \geq \Pr[R_{\rho+1} \geq h_{\rho+1}]$. Now note that the probability that at least one honest party (from the $h_{\rho+1}$ parties) will not receive when $S_{\rho+1} > \epsilon \cdot n/2$ nodes are sending is at most

$$h_{\rho+1}(1 - m/n)^{S_{\rho+1}} < h_{\rho+1}(1 - m/n)^{\epsilon \cdot n/2} \leq h_{\rho+1} \cdot e^{-\epsilon \cdot m/2} = \mathsf{negl}(\kappa)\,,$$

since $m \geq \kappa$. As such all $h_{\rho+1}$ nodes receive with overwhelming probability. Therefore $\Pr[R_{\rho+1} \geq h_{\rho+1}] \geq 1 - \mathsf{negl}(\kappa)$ and this completes the proof. □

Theorem 3. *Protocol* ConvergeRandom *from Figure 6 is an adaptively t-secure* Converge *protocol. Moreover its communication complexity is $O(n \cdot \log n \cdot |\mathcal{M}| \cdot m \cdot s)$ where s is the size of a message in $\mathcal{M}$.*

Proof. Follows by applying Lemma 9 for all initially-honest parties and for all of their initial messages. Every message m^* will be delivered to all honest parties that remain after the termination of the protocol, as required by Definition 5. The communication complexity follows directly from Lemma 7, since every party needs to call Propagate() $O(\log n)$ times. □

4.4 An Extension: The DistinctConverge Protocol

Recall that the Converge problem was defined with respect to a message set $\mathcal{M}$. For our application, we will need a slightly different version of the Converge problem, namely the DistinctConverge problem, defined with respect to a parameter k. In DistinctConverge, some elements of the set $\mathcal{M}$, while different, are considered the "same" because their k-bit prefixes are the same. Looking ahead, our set $\mathcal{M}$ is the set of all possible valid r-batches (a valid r-batch is a set of at least r signatures) on bit-slot pairs (b, s), denoted (b, s, r). In this set, two valid r-batches with different set of signatures but on the same (b, s) are considered the same. Our prior analysis applies as is to DistinctConverge but presenting Converge first simplified our exposition. We now give some definitions.

Definition 7 ($\texttt{distinct}_k$ function). *For any set M, $\texttt{distinct}_k(M)$ is a subset of M that contains all messages in M with distinct k-bit prefixes.*

E.g., for $M = \{01001, 01111, 11000, 10000\}$ we have that $\texttt{distinct}_2(M) = \{01001, 11000, 10000\}$. Note that $\texttt{distinct}_k$ is an one-to-many function. For example, $\texttt{distinct}_2(M)$ is also $\{01111, 11000, 10000\}$. We are now ready to present the DistinctConverge problem.

Definition 8 (t-secure DistinctConverge protocol). *Let $\mathcal{M}$ be a fixed set of messages and $k > 0$. A protocol Π executed by n parties, where every honest party p initially holds a set $M_p \subseteq \mathcal{M}$, is a* t-secure DistinctConverge protocol *if all honest parties, upon termination, output a set*

$$S_p \supseteq \texttt{distinct}_k \left(\bigcup_{p \in \mathcal{H}} M_p \right) ,$$

whenever at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of the protocol, with probability $1 - \mathsf{negl}(\kappa)$.

Our DistinctConvergeRandom, a slight modification of ConvergeRandom, is shown in Figure 7.

1: **procedure** $S_p \leftarrow \text{DISTINCTCONVERGERANDOM}_p(M_p, \mathcal{M}, k)$
Input: A set $M_p \subseteq \mathcal{M}$ and a parameter k.
Output: A set S_p.

2: **for** round $\rho = 1$ to $\lceil \log(\epsilon \cdot n) \rceil$ **do**
3: $\text{RECEIVE}^{\mathcal{F}_{\mathsf{prop}}} \leftarrow \mathcal{F}_{\mathsf{prop}}(\mathsf{SendRandom}, \mathtt{distinct}_k(M_p))$;
4: $\mathsf{Local}_p \leftarrow \mathsf{Local}_p \cup \text{RECEIVE}^{\mathcal{F}_{\mathsf{prop}}}$;
5: $M_p = \mathsf{Local}_p \cap \mathcal{M}$;
6: **return** Local_p;

Fig. 7. Our $\text{DISTINCTCONVERGERANDOM}_p$ protocol.

Theorem 4. *Let $k > 0$. Protocol* DISTINCTCONVERGERANDOM *from Figure 7 is an adaptively t-secure* DISTINCTCONVERGE *protocol. Moreover its communication complexity is*

$$O(n \cdot \log n \cdot |\mathtt{distinct}_k(\mathcal{M})| \cdot m \cdot s)$$

where s is the size of a message in $\mathcal{M}$.

Proof. Follows from Theorem 3 and since in DISTINCTCONVERGERANDOM an honest party always uses the function $\mathtt{distinct}_k$ before sending. □

5 Our Parallel Broadcast Protocol

In this section we present our PBC protocol PARALLELBROADCAST by using our protocol DISTINCTCONVERGERANDOM from Section 4.4. We recall that in PBC we have a set of n parties and each party p_i has an input bit b_i and acts as the designated sender. Therefore for each party p_i, a slot i is naturally defined as the underlying (single sender) broadcast with respect to the designated sender p_i. For the definition of a t-secure PBC protocol see Definition 2 in Section 2. We note that the PBC protocol that we will present here is secure against an adaptive adversary, as defined in Section 2.

To facilitate the exposition and our proof, our protocol PARALLELBROADCAST in Figure 9 is given in a hybrid world where two functionalities exist. The first one is $\mathcal{F}_{\mathsf{prop}}$ which was presented and instantiated in Section 4.2.

The second functionality is $\mathcal{F}_{\mathsf{mine}}$ (Figure 8), which was also presented and used in Chan et al. [5] and was shown to be instantiable from standard assumptions (with setup) by Abraham et al. [1]. We assume that when a party calls $\mathcal{F}_{\mathsf{mine}}$ then it returns instantaneously. A party p in our protocol queries $\mathcal{F}_{\mathsf{mine}}$ on input (Mine, b, s) in some round i, where $s \in [n]$ refers to one of the slots and $b \in \{0, 1\}$. If it receives response 1, it considers itself a member of a randomly selected subset of all the parties, which we will refer to as the "(b, s)-*committee*". More concretely, when $\mathcal{F}_{\mathsf{mine}}$ receives such a query, it flips a random coin to decide whether the party p is in that committee. $\mathcal{F}_{\mathsf{mine}}$ keeps the information and returns the same answer to all future identical queries by any party. We now give some necessary definitions.

Functionality: $\mathcal{F}_{\mathsf{mine}}$

$\mathcal{F}_{\mathsf{mine}}$ is parameterized by parties $1, \ldots, n$ and "mining" probability p_{mine}. Let $s \in [n]$ and $b \in \{0, 1\}$. Let *call* be vector of n entries initialized with -1.

On input (Mine, b, s) from party i:
- If $call_i = -1$ output $b = 1$ with probability p_{mine} or $b = 0$ with probability $1 - p_{\mathrm{mine}}$ and set $call_i = b$;
- Else output $call_i$.

On input $(\mathsf{Verify}, b, s, j)$ from party i output 1 if $call_j = 1$ and 0 otherwise.

Fig. 8. Functionality $\mathcal{F}_{\mathsf{mine}}$.

Definition 9 ((b, s)-committee). *For each pair of bit b and slot s, the (b, s)-committee is a subset of parties such that for each party c in the (b, s)-committee, whenever the $\mathcal{F}_{\mathsf{mine}}$ is queried on input $(\mathsf{Verify}, b, s, c)$, $\mathcal{F}_{\mathsf{mine}}$ outputs 1.*

Lemma 10 (Honest Committees). *Let $p_{\mathrm{mine}} = \min\{1, \kappa/(\epsilon \cdot n)\}$ be the probability of success for $\mathcal{F}_{\mathsf{mine}}$. Also set $R = 2\kappa/\epsilon$. Then, with probability $1 - \mathsf{negl}(\kappa)$, for each bit $b \in \{0, 1\}$ and slot $s \in [n]$, the (b, s)-committee will contain (i) at least one honest party and (ii) at most R dishonest parties.*

The proof of the above lemma is given in the Appendix.

Definition 10 (Valid r-batch). *A valid r-batch on pair (b, s) is the element*

$$b||s||\mathsf{SIG}_r\,,$$

where SIG_r is a set of at least r signatures on $[b, s]$ consisting of one signature from party s and at least $r - 1$ signatures from parties in the (b, s)-committee.

Definition 11. *We define $\mathcal{M}_r$ to be a set that contains all possible valid r-batches for all $b \in \{0, 1\}$ and for all $s \in [n]$.*

Lemma 11. *It is $|\mathtt{distinct}_{k^*}(\mathcal{M}_r)| = 2 \cdot n$, where k^* is the number of bits needed to represent $b||s$ and where $\mathtt{distinct}_{k^*}$ is defined in Definition 7.*

Proof. Follows from the fact that $\mathcal{M}_r$ contains exactly $2 \cdot n$ elements with unique $b||s$ prefixes, since $b \in \{0, 1\}$ and $s \in [n]$. □

5.1 Detailed Description of ParallelBroadcast

Our protocol (see Figure 9) is inspired by the single-sender protocol of Chan et al. [5] (ChBC protocol), of which we gave a detailed overview in the introduction. In particular, our protocol works in $R+1$ rounds as follows (Round $R+1$ is only used for updating the local sets and no sending takes place.)

First of all, every party p maintains n $\mathsf{Extracted}_s^p$ and Voted_p^s sets, $s \in [n]$, that are initialized as $\emptyset$. Roughly speaking, a bit $b \in \mathsf{Extracted}_s^p$ if p has observed

```
1: procedure {b_1, ..., b_n} ← ParallelBroadcast_p(b_p)
       Input: Local bit b_p.
       Output: Decision bits b_1, ..., b_n.

2:     Extracted_p^i = ∅, for i = 1, ..., n;                    ▷ global variable
3:     Voted_p^i = ∅, for i = 1, ..., n;                        ▷ global variable
4:     Local_p = ∅;                                             ▷ global variable
5:     Send(sig_p([b_p, p]), [n]);
6:     for round r = 1 to R + 1 do                              ▷ R is also a global variable
7:         Distribute_p(r);
8:         Vote_p(r);
9:     for slot i = 1, ..., n do
10:        return b_i ∈ Extracted_p^i if |Extracted_p^i| = 1 else return canonical bit 0;
```

Fig. 9. Our protocol. All global variables can be accessed by Distribute and Vote.

a valid r-batch on $[b, s]$; a bit $b \in \mathsf{Voted}_s^p$ if it has already been revealed that p is part of the (b, s)-committee (i.e., p has "voted").

In round 0, party i (for all $i \in [n]$) signs her input bit b_i, adds it to her $\mathsf{Extracted}_i^i$ set and sends b_i to all $n-1$ parties along with its signature on b_i, $\mathsf{sig}_i([b_i, i])$. From that point on, the distribution (Figure 10) and voting (Figure 11) phases follow, for every round $r = 1, \ldots, R$. These are non-trivial modifications (for many slots and using parallel gossiping) of the distribution and voting phases of ChBC. Our new distribution/voting phases, for round $r \leq R$, work as follows.

1. (Distribution) An honest party p first collects the set $\mathcal{V}$ of valid r-batches $v_1, \ldots, v_w$ on messages $[b_1, s_1], \ldots, [b_w, s_w]$ that are not in the respective p's Extracted sets. Then, instead of every node using a SEND-ALL to send each v_i (as ChBC would do), all nodes run DistinctConvergeRandom from Figure 7. As we showed, running DistinctConvergeRandom assures that all parties will eventually see the inputs from other parties but because there is overlap between input messages, it achieves its goal using less communication complexity. The pseudocode of Distribute is in Figure 10.
2. (Voting) An honest party p checks which valid r-batches in their local set correspond to pairs $(b, s) : b \notin \mathsf{Voted}_p^s$. For each such pair, they check whether they are members of the respective committee by using the functionality $\mathcal{F}_{\mathsf{mine}}$. If they are, they add their own signature to extend the valid r-batch to a valid $(r+1)$-batch. See the pseudocode of Vote in Figure 11.

We are now ready to prove the consistency and the validity of our protocol.

Lemma 12 (t-consistency of ParallelBroadcast). *Let $R = 2\kappa/\epsilon$ and $m = 10/\epsilon + \kappa$. ParallelBroadcast satisfies t-consistency, per Definition 2, in the $(\mathcal{F}_{\mathsf{mine}}, \mathcal{F}_{\mathsf{prop}})$-hybrid world with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Suppose for some slot s, an honest party p adds bit b to $\mathsf{Extracted}_p^s$ at some round r. We prove by the end of the protocol all honest parties j will add b

```
1: procedure DISTRIBUTE_p(r)

2:     Local_p ← Local_p ∪ RECEIVE();
3:     Let V = {v_i} be valid r-batches (in Local_p) on {[b_i, s_i]} s.t. b_i ∉ Extracted_p^{s_i};
4:     for all v_i ∈ V do
5:         Add b_i to Extracted_p^{s_i};
6:     if r ≤ R then
7:         Local_p ← DISTINCTCONVERGERANDOM_p(V, M_r, k*);
```

Fig. 10. The DISTRIBUTE procedure for a round r. Note that k^* is the number of bits required to represent $b||s$.

```
1: procedure VOTE_p(r)

2:     if r ≤ R then
3:         Let V = {v_i} be valid r-batches (in Local_p) on {[b_i, s_i]} s.t. b_i ∉ Voted_p^{s_i};
4:         for all v_i ∈ V s.t. F_mine(Mine, b_i, s_i) = 1 do
5:             Add b_i to Voted_p^{s_i};
6:             Add b_i in Extracted_p^{s_i} if b_i ∉ Extracted_p^{s_i};
7:             Extend v_i to a valid (r+1)-batch v'_i by adding p's signature on [b_i, s_i];
8:             SEND(v'_i, [n]);
```

Fig. 11. The VOTE procedure for a round r.

to their $\mathsf{Extracted}_j^s$ sets with probability at least $1 - \mathsf{negl}(\kappa)$—this will mean that for every slot s all honest parties will have identical $\mathsf{Extracted}^s$ sets by the end of the protocol, which is equivalent to consistency. We distinguish the following cases according to the step when p adds bit b to $\mathsf{Extracted}_p^s$ (We sometimes omit "with probability $1 - \mathsf{negl}(\kappa)$" when it is clear from the context.)

1. $r \leq R$ **and** p **adds** b **to** $\mathsf{Extracted}_p^s$ **in Line 6 of** $\textbf{VOTE}_p(r)$**.** This means that p sends a valid-$(r+1)$ batch v'_i for (b, s) to all parties via $\text{SEND}(v'_i, [n])$ in Line 8 of $\text{VOTE}_p(r)$ and therefore all parties j will add b to their $\mathsf{Extracted}_j^s$ sets during $\text{DISTRIBUTE}_j(r+1)$.
2. $r \leq R$ **and** p **adds** b **to** $\mathsf{Extracted}_p^s$ **in Line 5 of** $\textbf{DISTRIBUTE}_p(r)$: For this to happen, p has received, at round r, a valid r-batch v for (b, s) in Line 2 of $\text{DISTRIBUTE}_p(r)$. Valid r-batch v belongs to the set $\mathcal{V}$ provided as input to DISTINCTCONVERGERANDOM in Line 7 of $\text{DISTRIBUTE}_p(r)$. Since protocol DISTINCTCONVERGERANDOM is t-secure (Theorem 4), all honest parties output a set that contains v after $\text{DISTRIBUTE}_p(r)$ ends. By Lemma 10(i), there will be at least one honest voter ℓ in the (b, s)-committee. We distinguish two cases.
   (a) ℓ has not voted before for (b, s), i.e., $b \notin \mathsf{Voted}_\ell^s$. This means that ℓ sends a valid-$(r+1)$ batch v'_i for (b, s) to all parties via $\text{SEND}(v'_i, [n])$ in Line 8 of $\text{VOTE}_p(r)$ and therefore all parties j will add b to their $\mathsf{Extracted}_j^s$ sets during $\text{DISTRIBUTE}_j(r+1)$;

(b) ℓ voted before for (b, s), i.e., $b \in \mathsf{Voted}^s_\ell$. Let $r' < r$ be the round when ℓ voted for (b, s). This means that ℓ sent a valid-$(r'+1)$ batch v'_i for (b, s) to all parties via $\textsc{Send}(v'_i, [n])$ in Line 8 of $\textsc{Vote}_p(r')$ and therefore all parties j added b to their $\mathsf{Extracted}^s_j$ sets during $\textsc{Distribute}_j(r'+1)$.

3. p **adds** b **to** $\mathsf{Extracted}^s_p$ **in Line 5 of** $\textsc{Distribute}_p(R+1)$: In this case, p observes a valid $(R+1)$-batch for (b, s). By Lemma 10(ii), at least one of the voters, say voter ℓ, will be honest. Let $r' < R+1$ be the round when ℓ voted for (b, s). This means that ℓ sent a valid-$(r'+1)$ batch v'_i for (b, s) to all parties via $\textsc{Send}(v'_i, [n])$ in Line 8 of $\textsc{Vote}_p(r')$ and therefore all parties j added b to their $\mathsf{Extracted}^s_j$ sets during $\textsc{Distribute}_j(r'+1)$.

This completes the proof. □

Lemma 13 (t-validity of ParallelBroadcast). *Let $R = 2\kappa/\epsilon$ and $m = 10/\epsilon + \kappa$.* ParallelBroadcast *satisfies t-validity, per Definition 2, in the $(\mathcal{F}_{\mathsf{mine}}, \mathcal{F}_{\mathsf{prop}})$-hybrid world with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Follows directly from ParallelBroadcast and Distribute and the security of the signature scheme. Every honest sender s with input bit b_s, will execute Line 5 of ParallelBroadcast and thus will send to all parties message a valid 1-batch (their signature) for (b_s, s) at the beginning of the protocol. So, all honest parties h will add b_s to their $\mathsf{Extracted}^s_h$ sets at the beginning of $\textsc{Distribute}(1)$. Also, by the security of the signature scheme, with probability at least $1 - \mathsf{negl}(\kappa)$ no other bit b' could bare a valid signature from designated sender p_s, thus no other bit b' could be in an honest party's $\mathsf{Extracted}^s$ set. □

Theorem 5 (t-security of ParallelBroadcast). *Let $R = 2\kappa/\epsilon$ and $m = 10/\epsilon + \kappa$.* ParallelBroadcast *satisfies t-security, per Definition 2, in the $(\mathcal{F}_{\mathsf{mine}}, \mathcal{F}_{\mathsf{prop}})$-hybrid world.*

Proof. Adaptive security follows directly from Lemmata 12 and 13. □

Theorem 6 (t-security and communication of ParallelBroadcast). *Let $R = 2\kappa/\epsilon$ and $m = 10/\epsilon + \kappa$. The total number of bits exchanged by all parties in* ParallelBroadcast *is $\tilde{O}(n^2 \cdot \kappa^4)$.*

Proof. The commuication complexity of ParallelBroadcast is dominated by the communication complexity of DistinctConvergeRandom, which by Theorem 4 and Lemma 11 is $O(n^2 \cdot \log n \cdot m \cdot s)$. Since m is $\Theta(\kappa)$ and s, the size of every message/batch, is at most $R = 2\kappa/\epsilon$ (each element of the batch being a κ-bit signature) and DistinctConvergeRandom is run for R rounds, it follows that the communication complexity is $\tilde{O}(n^2 \cdot \kappa^4)$. □

6 Discussion and Conclusions

In this paper, we studied the communication complexity of BC and PBC with dishonest majority. We showed two protocols that achieve close to $O(n)$ improvement in communication complexity over the best existing protocols in their respective settings. We believe that there is room for future work in this direction.

6.1 Complexity for the Multivalued Case

Our protocols, as presented, work only for the case of binary domain. To extend our protocols to messages of ℓ bits, the trivial approach of broadcasting an ℓ-bit message bit by bit yields a communication complexity of $\tilde{O}(n^2 \cdot \kappa^3 \cdot \ell)$ (for our PBC protocol) and $\tilde{O}(n^2 \cdot \kappa^2 \cdot \ell)$ for our BC protocol. It would be interesting to see whether one can obtain protocols that achieve a better complexity, i.e., $O(n^2 \cdot \ell + \mathsf{poly}(n, \kappa))$ by combining our ideas with approaches from the literature on *broadcast extension* for dishonest majority (see, e.g., [12,6]). Namely, this line of work uses broadcast on short messages and techniques from coding theory to obtain BC protocols with asymptotically optimal complexity $O(n\ell)$, given ℓ is large enough. It is interesting whether such techniques can also be used to improve the complexity for the specific case of PBC (where the optimal complexity is $O(n^2\ell)$, asymptotically).

6.2 Further Work on PBC

We have initiated the study of PBC protocols which leverage parallelity to obtain better communication complexity and security than what is possible from naively combining n single sender BCs. There are many open questions in this area; we elaborate on a few of them here. Most closely related to our own work is the question of studying PBC under different trust assumptions (bulletin PKI vs. trusted PKI). For example: is it possible to get an adaptively secure PBC protocol under *bulletin PKI* which has $o(n^4)$ communication complexity in the dishonest majority setting? (Note that a naive version of PBC via n Dolev-Strong BCs would lead to $O(n^4\kappa)$ communication complexity.) With regards to adaptivity, another interesting question for follow up work would be to consider the communication efficiency of PBC protocols in the *strongly adaptive* adversarial model, where the adversary can observe a party p's messages for round r, then adaptively corrupt p and *delete any of p's messages for round r*. In this manner, the adversary can replace p's messages with its own, or send conflicting messages to the messages that p sent prior to being corrupted. Similar to [5], our protocol would become insecure in such a scenario. However, recent work of Wan et al. [21] shows how to overcome this issue by using time-locked puzzles. It would be interesting to see if their approach could also be applied to our PBC protocol to yield a protocol with $o(n^3)$ communication complexity in the dishonest majority setting. Finally, studying PBC under different corruption thresholds (i.e., $t < n/2$ or $t < n/3$) is also a completely open direction for future research.

References

1. Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In Peter Robinson and Faith Ellen, editors, *38th ACM Symposium Annual on Principles of Distributed Computing*, pages 317–326. Association for Computing Machinery, July / August 2019.

2. Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 353–380. Springer, 2020.
3. Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $O(\sqrt{n})$-bits barrier: Balanced byzantine agreement with polylog bits per-party. Cryptology ePrint Archive, Report 2020/130, 2020. `https://eprint.iacr.org/2020/130`.
4. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.
5. T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 246–265. Springer, Heidelberg, May 2020.
6. Wutichai Chongchitmate and Rafail Ostrovsky. Information-theoretic broadcast with dishonest majority for long messages. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 370–388. Springer, Heidelberg, November 2018.
7. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
8. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th Annual ACM Symposium on Theory of Computing*, pages 148–161. ACM Press, May 1988.
9. Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *DISC*, volume 5805 of *LNCS*, pages 449–463. Springer, 2009.
10. Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual Symposium on Foundations of Computer Science*, pages 658–668. IEEE Computer Society Press, October 2007.
11. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement in t+1 rounds. In *25th Annual ACM Symposium on Theory of Computing*, pages 31–41. ACM Press, May 1993.
12. Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: The $t < n$ case. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 448–465. Springer, Heidelberg, December 2014.
13. Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In Andréa W. Richa and Rachid Guerraoui, editors, *29th ACM Symposium Annual on Principles of Distributed Computing*, pages 420–429. Association for Computing Machinery, July 2010.
14. Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. *J. ACM*, 58(4):18:1–18:24, 2011.
15. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 990–999. ACM-SIAM, January 2006.

16. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(382–401), July 1982.
17. Silvio Micali. Very simple and efficient byzantine agreement. In Christos H. Papadimitriou, editor, *ITCS 2017: 8th Innovations in Theoretical Computer Science Conference*, volume 4266, pages 6:1–6:1, 67, January 2017. LIPIcs.
18. Silvio Micali and Vinod Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. Technical report, MIT, 2017.
19. Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement, 2021.
20. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
21. Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 412–456. Springer, Heidelberg, November 2020.
22. Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. Cryptology ePrint Archive, Report 2020/590, 2020. `https://eprint.iacr.org/2020/590`.
23. Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 381–411. Springer, Heidelberg, November 2020.

Appendix

Chernoff Bound for independent random variables

Lemma 14. *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables, with $\Pr[X_i = 1] = p_i$, for each $i = 1, \ldots, n$. Also, let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for any $\delta > 0$, it holds that:*

1. *(Upper tail Chernoff Bound)* $\Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$,
2. *(Lower tail Chernoff Bound)* $\Pr[X < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}$.

Deferred Proofs

Proof of Lemma 10. The trivial case $\kappa \geq \epsilon n$ is of no interest, since then $R \geq 2 \cdot n$, so we will focus on the case $\kappa < \epsilon n$ and thus $p_{\text{mine}} < 1$.
For each party i (honest or dishonest), we define indicator random variables $X_i = 1$ if F_{mine} has i in the respective committee. The r.v.s are independent, since F_{mine} throws an independent random coin for each choice, with $\Pr[X_i = 1] = \frac{\kappa}{\epsilon \cdot n}$.
We prove each case separately:

1. We bound the probability of the event that no honest party is elected to the respective committee.
   Define $X_1 = \sum_{i \text{ honest}} X_i$, then:

$$\begin{aligned}
\Pr[X_1 = 0] &= \Pr\left[\bigcap_{i \text{ honest}} X_i = 0\right] \\
&= \prod_{i \text{ honest}} \Pr\left[X_i = 0\right] \text{ (due to independence)} \\
&\leq \prod_{i=1}^{\epsilon \cdot n} \Pr\left[X_i = 0\right] \text{ (at least } \epsilon \cdot n \text{ honest parties)} \\
&= (1 - p_{\text{mine}})^{\epsilon \cdot n} \\
&\leq e^{-\epsilon \cdot n \cdot p_{\text{mine}}} \text{ (by use of } 1 + x \leq e^x) \\
&= e^{-\kappa} \text{ (since } p_{\text{mine}} = \frac{\kappa}{\epsilon \cdot n})
\end{aligned}$$

2. We bound the probability of the event that more than R dishonest parties are elected to the respective committee.
   Define $X_2 = \sum_{j \text{ dishonest}} X_j$, with $\mathbb{E}\left[X_2\right] = \sum_{j \text{ dishonest}} \Pr[X_j = 1] = |S_3| \cdot p_{\text{mine}} = |S_3| \cdot \frac{\kappa}{\epsilon \cdot n} \leq \frac{1-\epsilon}{\epsilon} \cdot \kappa$.
   Also, $R = \frac{2\kappa}{\epsilon} \geq (1+\delta)(1-\epsilon) \cdot n \cdot \frac{\kappa}{\epsilon \cdot n} \geq (1+\delta) \cdot \mathbb{E}\left[X_2\right]$, for any $\delta \in \left(0, \frac{1+\epsilon}{1-\epsilon}\right]$.
   So, if we define as X_2^* the random variable $X_2^* = \sum_{j=1}^{(1-\epsilon)n} X_j$, from applying a Chernoff bound, we have

$$\begin{aligned}
\Pr\left[X_2 > R\right] &\leq \Pr\left[X_2^* > R\right] \\
&\leq \Pr\left[X_2^* \geq (1+\delta)\mathbb{E}\left[X_2^*\right]\right] \\
&\leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^{\frac{1-\epsilon}{\epsilon} \cdot \kappa}. \qquad (3)
\end{aligned}$$

   We distinguish the cases:

   $\epsilon < 1/2$: Then, $1 - \epsilon > 1/2$ and $1/\epsilon > 2$, thus $(1-\epsilon)/\epsilon > 1$. Pick $\delta = 1/2$, then Equation 3 becomes

$$\begin{aligned}
\Pr\left[X_2 > R\right] &\leq \left[\left(\frac{8 \cdot e}{27}\right)^{1/2}\right]^{\frac{1-\epsilon}{\epsilon} \cdot \kappa} \\
&< \left[\left(\frac{8 \cdot e}{27}\right)^{1/2}\right]^{\kappa}.
\end{aligned}$$

$\underline{\epsilon \geq 1/2}$: Set $\delta = \frac{1+\epsilon}{1-\epsilon}$, thus $1+\delta = \frac{1+\epsilon+1-\epsilon}{1-\epsilon} = \frac{2}{1-\epsilon}$. Then, Equation 3 becomes

$$\begin{aligned}
\Pr[X_2 > R] &\leq e^{-(1-\epsilon)\frac{\kappa}{\epsilon}} \cdot \left(\frac{e}{1+\delta}\right)^{(1+\delta)(1-\epsilon)\frac{\kappa}{\epsilon}} \\
&\leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)(1-\epsilon)\frac{\kappa}{\epsilon}}, \text{ (since } \frac{1-\epsilon}{\epsilon} \cdot \kappa > 0) \\
&= \left(\frac{e \cdot (1-\epsilon)}{2}\right)^{\frac{2\kappa}{\epsilon}}, \text{ (since } 1+\delta = \frac{2}{1-\epsilon}) \\
&\leq \left(\frac{e}{4}\right)^{2\kappa}, \text{ (since } \epsilon \geq 1/2 \Rightarrow \frac{e \cdot (1-\epsilon)}{2} \leq \frac{e}{4} \text{ and } 1/\epsilon > 1).
\end{aligned}$$

□