Gossiping for Communication-Efficient Broadcast

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

[1] University of Maryland
[2] CISPA Helmholtz Center for Information Security
[3] Yale University

Abstract. Byzantine Broadcast is crucial for many cryptographic protocols such as secret sharing, multiparty computation and blockchain consensus. 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) and propose new communication-efficient protocols, under dishonest majority, for Single-Sender Broadcast (BC) and Parallel Broadcast (PBC), improving the state-of-the-art in several ways.

As our warm-up result, we present a randomized protocol for BC which achieves $O(n^2\kappa^2)$ communication complexity from plain public key setup assumptions. This is the first protocol with subcubic communication in this setting, but operates only against static adversaries.

Using ideas from our BC protocol, we move to our central contribution and present two protocols for PBC that are secure against adaptive adversaries. To the best of our knowledge we are the first to study PBC *specifically*: All previous approaches for Parallel Broadcast naively run n instances of single-sender Broadcast, increasing the communication complexity by an undesirable factor of n. Our insight of avoiding black-box invocations of BC is particularly crucial for achieving our asymptotic improvements. In particular:

1. Our first PBC protocol achieves $\tilde{O}(n^3\kappa^2)$ communication complexity and relies only on plain public key setup assumptions.
2. Our second PBC protocol uses trusted setup and achieves nearly optimal communication complexity $\tilde{O}(n^2\kappa^4)$.

Both PBC protocols yield an almost linear improvement over the best known solutions involving n parallel invocations of the respective BC protocols such as those of Dolev and Strong (SIAM Journal on Computing, 1983) and Chan et al. (Public Key Cryptography, 2020). Central to our PBC protocols is a new problem that we define and solve, which we name "Converge". In Converge, parties must run an adaptively-secure and *efficient* protocol such that by the end of the protocol, all honest parties that remain possess a superset of the union of the initial honest parties' inputs.

1 Introduction

Since its formalization by Lamport et al. [19], the broadcast problem has been studied in many works and lies at the core of many cryptographic protocols such as secret sharing, multiparty computation, and blockchain consensus protocols (e.g., [23]). 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 [26]) 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. PBC comprises a central component of protocols like verifiable secret sharing and multiparty computation (e.g., [2,20]), where whenever broadcast is used in these protocols, it is always a *parallel broadcast*. Any improvement in PBC therefore yields an improvement in these protocols as well.

A crucial distinction among broadcast protocols is which *implementation model* they use: In the *bulletin PKI* model, 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 the stronger *trusted PKI* model, trusted setup is required: A trusted party generates all keys honestly and distributes them to the parties prior to protocol execution. An important efficiency metric for broadcast protocols, which is the focus of this paper, is their *communication complexity*, i.e., how many bits are exchanged during the protocol. Often, it depends on the implementation model and directly affects the efficiency of the underlying protocols using BC or PBC.

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. Our first (warm-up) result provides the first statically-secure BC protocol with subcubic communication in the bulletin PKI model. We then continue with our central contributions comprising two new PBC protocols. Our main observation is that no results exist on the communication complexity of PBC *specifically*. In particular, all protocols for PBC, to the best of our knowledge, are implemented via simultaneous calls to n BC instances, leading to increased communication complexity. Leveraging this insight, we use ideas of our first warmup BC protocol to build the first adaptively-secure PBC protocol with cubic communication complexity in the bulletin PKI model (Again, one defining feature of our PBC protocol is that it is non-black-box, in that it does not use n simultaneous calls to BC.) Our final PBC result shows how to improve the PBC complexity to quadratic, by switching to the trusted PKI model. To the best of our knowledge, all three results comprise significant improvements (i.e., by a linear factor) in the communication complexity of the state-of-the-art.

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 [8,10]. 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 minimize the use of SEND-ALL instructions via *gossiping*, eventually yielding much better communication. 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 technique on the complexity of broadcast has not been explored before. Our current protocols are for the binary case only and might become inefficient when messages become longer.

Table 1. Comparison of BulletinBC, BulletinPBC and TrustedPBC, in terms of communication complexity in bits (CC) and round complexity (RC), to existing work. Number of parties is n. Also $\epsilon < 1$.

protocol	model	CC	RC	adversary	dishonest	type
Abraham et al. [1]	trusted PKI	$\tilde{O}(n \cdot \kappa)$	$O(1)$	adaptive	$< n/2$	BC
Momose and Ren [25]	bulletin PKI	$\tilde{O}(n^2 \cdot \kappa)$	$O(n)$	adaptive	$< n/2$	BC
Chan et al. [8]	trusted PKI	$O(n^2 \cdot \kappa^2)$	$O(\kappa)$	adaptive	$< (1-\epsilon) \cdot n$	BC
Dolev and Strong [10]	bulletin PKI	$O(n^3 \cdot \kappa)$	$O(n)$	adaptive	$< n$	BC
BulletinBC §3	bulletin PKI	$O(n^2 \cdot \kappa^2)$	$O(n)$	static	$< (1-\epsilon)n$	BC
BulletinPBC §5	bulletin PKI	$\tilde{O}(n^3 \cdot \kappa^2)$	$O(n \log n)$	adaptive	$< (1-\epsilon)n$	PBC
TrustedPBC §6	trusted PKI	$\tilde{O}(n^2 \cdot \kappa^4)$	$O(\kappa \log n)$	adaptive	$< (1-\epsilon)n$	PBC

1.1 Communication-Efficient BC in the Bulletin PKI Model

In our first contribution (Section 3) we use gossiping to achieve communication-efficient BC protocols *in the bulletin PKI model*. When using bulletin PKI, it is 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 [10] (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 BulletinBC (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 [10]. 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 SEND-ALL(x) instruction. Our proposed protocol *just replaces* the

SEND-ALL(x) instruction with a SEND-RANDOM(m, x) instruction and runs for an additional $O(\log n)$ rounds. Our SEND-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 SEND-ALL(x) via a sequence of SEND-RANDOM(m, x) instructions as this would still lead to cubic communication complexity—again, it just replaces the instructions directly. We remark that while the resulting changes in the protocol are minimal and only cause the protocol to run for roughly an additional $\log n$ many steps, the security proof is affected substantially. In particular, our gossiping technique does not protect against adaptive adversaries who can simply wait and corrupt all recipients of particular SEND-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 Communication-Efficient PBC for Bulletin and Trusted PKI

Recall that in PBC all n parties simultaneously act as a sender and wish to consistently distribute their message. As we mentioned before, all PBC protocols 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 Dolev-Strong protocol [10], would yield $\tilde{O}(n^4)$ communication complexity. Instead, we do not use a BC protocol as a black-box but we instantiate protocols specifically for PBC. This leads to our next contributions (Sections 5, 6).

The $\mathcal{M}$-*Converge* Problem. Central to our PBC protocols is the $\mathcal{M}$-*Converge* problem (Section 4) that we define for the first time and could be of independent interest. In the $\mathcal{M}$-Converge problem, there is a fixed message set $\mathcal{M}$ and all initial honest parties $p \in \mathcal{H}$ begin with a set of messages $M_p \subseteq \mathcal{M}$ and a constraint set of messages $\mathcal{C}_p \subseteq \mathcal{M}$. In the end, all remaining honest parties q (we consider an adaptive adversary) should output a set S_q that is a *superset* of $\bigcup_p M_p - \bigcup_p \mathcal{C}_p$. (We consider superset since the adversary can inject messages as well.) Clearly, there is a simple protocol that solves the $\mathcal{M}$-Converge problem in the presence of an adaptive adversary and in a single round: Have all honest parties send their sets M_p and $\mathcal{C}_p$ to all other n parties—however this protocol leads to $O(n^2|\mathcal{M}|)$ communication since both M_p and $\mathcal{C}_p$ are subsets of $\mathcal{M}$. Instead, in our $\mathcal{M}$-ConvergeRandom protocol (see Figure 6), every honest party runs in a few more rounds (around $O(\log n)$) and in every round sends every message in her local set to a randomly-chosen subset of parties and *not to all of them*. This reduces the communication to $\tilde{O}(n|\mathcal{M}| + n^2)$, which, for the case of $|\mathcal{M}| = \Omega(n)$ (as in our applications), leads to a much improved worst-case communication complexity! The main idea of our protocol is as follows.

1. In round 1, honest party p, for every message $x \in M_p - \mathcal{C}_p$, picks $i \in [n]$ as a recipient with probability m/n (Again, m is $\Theta(\kappa)$.) Then p constructs lists $\mathcal{L}_j$ (for $j = 1, \ldots, n$) containing messages $x \in M_p - \mathcal{C}_p$ that were assigned to recipient j. Each list $\mathcal{L}_j$ is first padded to at most $2m\lceil|\mathcal{M}|/n\rceil$ elements

(which is enough to ensure no overflows by Chernoff) and then encrypted to ensure an adaptive adversary does not gain any advantage. Then $\mathcal{L}_j$ 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 adds to her constraint set $\mathcal{C}_p$ all messages sent in the previous round $i-1$, collects all lists $\mathcal{L}_p$ from round $i-1$ into a new set M_p and performs the same task as in round 1 (random assigning of elements in $M_p - \mathcal{C}_p$, compilation into lists and sending).

We prove that with overwhelming probability, the above gossiping protocol delivers every element contained in the sets of the initially-honest parties (except the ones in the initial constraint sets) to all remaining honest parties, see Lemma 9. The communication complexity is indeed $\tilde{O}(n|\mathcal{M}| + n^2)$ since in every round all n parties send to all n parties a message of size $2m\lceil |\mathcal{M}|/n \rceil \leq 2|\mathcal{M}| \cdot m/n + 2$.

We emphasize that constraint sets used in $\mathcal{M}$-ConvergeRandom are crucial in reducing the communication complexity of our $\mathcal{M}$-ConvergeRandom protocol, in particular when $\mathcal{M}$-ConvergeRandom has to be called a large number of times by another protocol. This is because any message m that is sent by party p in round 1 of $\mathcal{M}$-ConvergeRandom enters p's constraint set and is therefore never sent again by p. A protocol can thus avoid resending messages during future calls of $\mathcal{M}$-ConvergeRandom, by initializing future constraint sets to contain already sent messages. We finally note that one of our protocols presented in this paper (TrustedPBC) invokes $\mathcal{M}$-ConvergeRandom a small number of times, so for simplicity we do not use constraint sets.

While the above protocol achieves the desired communication complexity bounds, the crucial question is whether it achieves *adaptive security* (It does.) Recall that the danger with picking a few random parties to send the message in an adaptive setting (as in our statically secure broadcast protocol) was that the adversary can corrupt the specific set of recipients for the sent message. This cuts off the propagation and leaves the sender as the only honest party who has the message! Our protocol avoids that because it never reveals which are the true recipients of the message by padding each list $\mathcal{L}_j$ to the same size and then encrypting it (Secure state erasures are required as well for adaptive security.) More intuitively, this can be understood as having the (different) messages that are sent by a sender in a particular round of this process pose as "cover traffic" for one another. This cover traffic hides to whom a particular message is being sent (since all the recipients of a single sender obtain the same amount of encrypted information), and hence makes it impossible to trace the path of that message.

Technical Highlights for BulletinPBC (Section 5). This protocol follows a similar framework as the Dolev-Strong protocol [10]. 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 bit b to the a local set (which we call Extracted), adds her signature and sends a message x of $r+1$ signatures *to all parties*, using a SEND-ALL(x) instruction. As r goes to n, this instruction incurs $O(n^3 \cdot \kappa)$ communication complexity, since all parties potentially send $O(n)$ sized lists of signatures to all. Combining n of these protocols naively would yield a PBC protocol with prohibitive communication complexity of $O(n^4 \cdot \kappa)$.

To overcome this issue, we observe that the main communication overhead in this naive protocol comes from steps where all parties send lists of $O(n)$ signatures *for the same designated sender*. For example, it could be possible that all parties hold the same list of r signatures in some round r that made them accept a message s for some slot i in that round. Clearly, it is extremely wasteful for *all* honest parties to send *all* of these signatures to *all* parties via SEND-ALL(x) (after appending their own respective signatures), since there is a large amount of redundancy among these lists.

Our proposed protocol deals with this situation by viewing it as an instance of the $\mathcal{M}$-*Converge* problem, where input sets consist of the signatures in the parties' lists and the constraint sets ensure that signatures which were already sent will not be repeated. Hence, we use our $\mathcal{M}$-ConvergeRandom protocol to propagate lists in that case much more efficiently, namely with $O(n^2 \cdot \kappa^2)$ complexity (rather than naively within $O(n^3 \cdot \kappa)$). However, in the above discussion we focused on only a single slot i. Yet, there is an issue when applying this strategy to a single instance of Dolev-Strong BC: since there are $O(n)$ rounds where parties might call $\mathcal{M}$-ConvergeRandom, the total communication complexity is $O(n^3 \cdot \kappa^2)$, even worse than the original DS protocol! It is here that we once again leverage the inherent parallel structure of PBC. Namely, in our protocol, we use $\mathcal{M}$-ConvergeRandom to propagate messages for all slots *simultaneously*, regardless of what slot they belong to. Moreover, we run one instance of $\mathcal{M}$-ConvergeRandom per step of the protocol, for a total of $O(n)$ many instances. While it is possible that for some of these instances, the complexity increases to $O(n^3 \cdot \kappa^2)$, thanks to the constraint sets we bound the number of such instances by $O(1)$. This results in a total of $O(n^3 \cdot \kappa^2)$ complexity, or an amortized $O(n^2 \cdot \kappa^2)$ complexity per slot.

Technical Highlights for TrustedPBC (Section 6). Our starting point for this protocol is the recent protocol by Chan et al. [8] 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. This step is executed only by the committee and the message's size is at most κ signatures of κ bits each; hence the induced communication complexity is $O(n \cdot \kappa^3)$;
2. (Distribution) When an honest party p observes 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.)

We then observe that if we naively use the ChBC protocol for the parallel case, the communication complexity of the distribution phase grows 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 twice), for each bit b and for each sender slot p. To overcome this issue we once again observe that we can abstract each step (there are κ many) of the protocol as an instance of the $\mathcal{M}$-Converge problem to improve communication complexity. This leads to the final version of our TRUSTEDPBC protocol, with amortized linear complexity $O(n \cdot \kappa^4)$ per sender slot.

1.3 Related Work

The problem of BC was originally introduced in the celebrated work of Lamport, Shostak, and Pease [19]. 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 [11,14]. More recently, a line of work initiated by King et al. [5,17,18] gave setup-free protocols for the case of $t < n/3$ that require $\tilde{O}(n^{3/2})$ communication and Momose and Ren [25] 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 [10] 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. [8] gives a protocol that requires $\tilde{O}(n^2 \cdot \kappa)$ bits of communication and requires trusted setup. What can be seen as a statically secure version of Chan et al.'s protocol was suggested by Buterin [6] and gives a protocol with $O(n \cdot \kappa)$ communication complexity and also requires trusted setup (to elect a random committee at the onset of the protocol). In the range of $t < n/3$ and $t < n/2$, the works of Micali [23], Micali and Vaikuntnathan [24], 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 [8,12,13,27]. Finally, the problem of interactive consistency or parallel broadcast was originally introduced by Pease et al. [26]. Another line of works studies the *round complexity* of BC, e.g., [8,13,27]. See Table 1 for overview of communication-efficient protocols.

Gossip protocols (also known as flooding protocols) are a simple, efficient type of information propagation and are used as a background layer in many infrastructures (e.g., blockchain protocols). Loosely speaking, they are based on the principle of exponential graph expansion: a party sends its message to a small random sample of its neighbours who in turn do the same. It is well-known [9] and easy to see that within $O(\log(n))$ rounds, all n parties learn the message. Some works [16] consider more refined versions of data dissemination, where parties keep sending until they hear from a certain number of their neighbours. These protocols work well in the presence of random, benign (i.e., crash) faults, but fail completely in the presence of malicious (byzantine) faults. This was addressed in a line of works such as [4,21]. However, as recently noticed by [22],

none of these protocols work in the presence of a fully adaptive adversary that can corrupt parties based on the propagation of a specific message. It is easy to see why: an adaptive adversary can simply corrupt all the (randomly elected) neighbours an honest party sent to and inhibit further propagation of a message in this manner. Such attacks are sometimes referred to as "eclipsing attacks" in the cryptocurrency space. Hence, in [22] they considered flooding protocols in the presence of an adaptive adversary who dynamically chooses who to corrupt, but whose corruptions take a while to become active. This gives the nodes time to propagate the messages as necessary. An example of gossip used in a byzantine consensus protocol is [15]: this work shows how to obtain, using gossip, a reliable broadcast protocol with quasi-linear (in n) communication for an asynchronous network. Being asynchronous, their protocol heavily relies on the fact that fewer than $n/3$ parties are corrupted.

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 vs. Trusted PKI. We briefly recall the difference between Bulletin PKI and Trusted PKI here. For the first part of our work, we assume that parties share a *public key infrastructure* (Bulletin 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. For the final result of this work, we use the Trusted PKI setting, where a trusted party computes and distributes secret/public key pairs to the protocol participants, enabling the use use of more powerful cryptographic primitives in our constructions, such as verifiable random functions.

Signatures. Party i computes a *signature* σ on a message m via $\sigma \leftarrow sig(sk_i, m)$. Later σ can be verified via $ver(pk_i, \sigma, m)$. As is standard for this line of work, we assume signatures are *idealized* 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*, i.e., for any m, $ver(pk_i, sig(sk_i, m), m) = 1$. We write $sig_i(m)$ to indicate $sig(sk_i, m)$ and $ver_i(\sigma, m)$ to indicate $ver(pk_i, \sigma, m)$.

Communication 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 $r+1$. 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. Finally, we assume secure erasures, namely a party p can safely erase her state so that the adversary cannot access it whenever the adversary corrupts p (after the state has been erased), e.g. [3,17].

t-Secure Broadcast and t-Secure Parallel Broadcast. We now provide the definitions of t-Secure Broadcast and t-Secure Parallel Broadcast which are the focus of the paper's main body.

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 all parties terminate upon output, is a* t-secure broadcast protocol *if the following properties hold 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 i holds an input v_i and terminates upon outputting an n-value vector $\mathbf{V}_i$, is a* t-secure parallel broadcast protocol *if the following properties hold with probability $1-\mathsf{negl}(\kappa)$ whenever at most t parties are corrupted:*

*t-**validity:** If party s is honest, then all honest parties i have $\mathbf{V}_i(s)=v_s$.*

*t-**consistency:** all honest parties output the same vector $\mathbf{V}'$.*

We define a "*slot*" s to denote in PBC the equivalent of the execution of a single sender BC, if party p_s was the designated sender. Thus, the output bit of a party p_i for a slot s is $\mathbf{V}_i(s)$. A slot s is honest, if p_s is honest.

Adversary Model. 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. For Section 3, we consider a *static adversary*, who chooses all the parties to corrupt before the execution of the protocol. For Sections 4 to 6, we consider an *adaptive adversary*, 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. However, 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). Moreover, we assume *atomic sends [3]*: an honest party p can send to multiple parties simultaneously, without the adversary being able to corrupt p in between sending to two parties.

MPC Model. One part of our parallel broadcast protocol is modeled using the ideal functionality $\mathcal{F}_{\mathsf{prop}}$ (See Figure 4.) To show that a specific implementation realizes $\mathcal{F}_{\mathsf{prop}}$, we are using the definition of synchronous MPC in the standalone model by Canetti [7] which we briefly recall here. Let f be a (possibly randomized) function that takes n inputs. In the real world execution of a protocol Π for computing f, party P_i initially holds 1^κ and an input x_i. The adversary holds 1^κ and an auxiliary input z. The goal of running Π is for all honest parties P_i to learn output y_i, where $[y_1, \ldots, y_n] \leftarrow f(x_1, \ldots, x_n)$. During the execution, the adversary $\mathcal{A}$ can corrupt any party adaptively, upon which it learns the internal state of this party. At the end of the execution, each honest party outputs its local output (as instructed by Π) and the adversary $\mathcal{A}$ outputs its entire view. We write $\mathbf{Real}_{\Pi,\mathcal{A}}(1^\kappa, \mathbf{x}, z)$ to denote the distribution over the vector of outputs of honest parties and the set of corrupted parties in the above (real-world) 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. Parties hold inputs as above; 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.[4]
Output. The honest parties output their view.
Post-execution corruptions. $\mathcal{S}$ may corrupt additional parties and output (any function of) its view.

$\mathbf{Ideal}_{f,\mathcal{S}}(1^\kappa, \mathbf{x}, z)$ denotes the distribution over the vector of outputs of honest parties and the set of corrupted parties in the above (ideal-world) experiment.

Definition 3 (Secure Computation). *Π t-securely computes f if for all PPT adversaries $\mathcal{A}$ corrupting 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. Aanalyzing it seperately, allows us to argue about the consistency and validity of our protocol BulletinBC in a structured manner.

[4] Since we assume erasure, the adversary learns nothing new from such corruptions.

1: **procedure** $G \leftarrow$ AddRandomEdges(V, S_2, S_3, S, m)
Input: Set of n nodes V; disjoint subsets of V: S_2, S_3; $S \subseteq V - (S_2 \cup S_3)$; Integer $m \leq n$.
Output: A graph G.

2: Let G be an empty graph with node set V;
3: **for** every node $v \in S$ **do**
4: **for** every node $u \in V$ **do**
5: Add an edge (v, u) to G with probability $\frac{m}{n}$;
6: **return** G;

Fig. 1. The AddRandomEdges procedure.

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 represents the set of parties that send a message q at a specific round r and S_2 represents 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. We want to compute how many parties in S_2 receive q (for the first time) during round r, so we study the degree of nodes in S_2. We define the following indicator random variables.

Definition 4. *Let $G \leftarrow$ 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 allows 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* AddRandomEdges(V, S_2, S_3, S, m) *per Definition 4. Then for $m \geq 15/\epsilon$,*

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

3.2 The Protocol BulletinBC

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

Intuition: From SEND-ALL to Gossiping. As we mentioned in the introduction, our protocol is inspired by the protocol of Dolev and Strong [10] 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 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 BulletinBC. 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 [10]: (1) we increase the number of rounds from t to $t+R$ (Line 5); (2) instead of sending to all parties we send to each party randomly with probability m/n (Line 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 9. We now continue with the proof of consistency and validity of BulletinBC. We first define, using notation consistent with AddRandomEdges, the following sets of parties (w.r.t. 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;
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 r.

1: **procedure** $b' \leftarrow \text{BULLETINBC}_p()$
Input: A bit b if $p = s$, no input otherwise.
Output: Decision bit b'.

2: $\mathsf{Extracted}_p = \mathsf{Local}_p = \emptyset$;
3: **if** p is the designated sender s **then**
4: SEND($\mathsf{sig}_s(b), [n]$);
5: **for** round $r = 1$ to $t + R$ **do**
6: $\mathsf{Local}_p \leftarrow \mathsf{Local}_p \cup$ RECEIVE();
7: **for** bit $x \in \{0, 1\}$ **do**
8: $S \leftarrow$ DISTINCTSIGS(x, Local_p, s);
9: **if** $|S| \geq \min\{r, t + 1\} \wedge x \notin \mathsf{Extracted}_p$ **then**
10: $\mathsf{Extracted}_p = \mathsf{Extracted}_p \cup x$;
11: **for** party $i = 1$ to n **do**
12: SEND($\mathsf{sig}_p(x) \cup S, i$) with probability $\frac{m}{n}$;
13: **return** $b' \in \mathsf{Extracted}_p$ if $|\mathsf{Extracted}_p| = 1$, otherwise **return** canonical bit 0;

Fig. 2. Our BULLETINBC_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.

Define S_3 be the set of malicious parties ($|S_3| = n - \epsilon n$). We show (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* BULLETINBC *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 that with probability at least $(1 - p)^{\rho - r}$:*

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

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 $p^* = \epsilon \cdot n \cdot e^{-2\epsilon \cdot m/9}$.*

Lemma 3 (t-consistency: BULLETINBC). *Let* $R = \lceil \log_3(\epsilon n) \rceil$, $m = \Theta(\kappa)$. BULLETINBC *satisfies t-consistency (Definition 7), 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 that by the end of the protocol all honest parties j add b to their $\mathsf{Extracted}_j$ sets with probability $1 - \mathsf{negl}(\kappa)$—this means that all honest parties 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 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, meaning that at round $\mathbf{r}+1 = r+R$ we have $S_1(b, r+R) \geq 3^R = 3^{\lceil \log_3(\epsilon \cdot n) \rceil} \geq 3^{\log_3(\epsilon \cdot n)} = \epsilon \cdot n$, with probability at least $(1-p)^R \geq 1-R \cdot p$, by Bernoulli's inequality[5] and since $-p \geq -1, R \geq 1$. Note that for $m = \Theta(\kappa)$, $R \cdot p$ is $\mathsf{negl}(k)$, since $n = poly(\kappa)$.
Case $r \geq t+1$: Suppose an honest party i adds bit b to $\mathsf{Extracted}_i$ at some round $r \geq t+1$. Then, 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, case $r'' < t+1$ applies to honest party j. Thus all honest parties add b to their $\mathsf{Extracted}$ sets by the end of the protocol, with probability $1-\mathsf{negl}(\kappa)$. □

Lemma 4 (t-validity: BulletinBC). *Let $R = \lceil \log_3(\epsilon \cdot n) \rceil$, $m = \Theta(\kappa)$.* BulletinBC *satisfies t-validity, per Definition 7, with probability $1-\mathsf{negl}(\kappa)$.*

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

Theorem 1 (Communication complexity: BulletinBC). *Let $m = \Theta(\kappa)$ and $R = \lceil \log_3(\epsilon n) \rceil$. The total number of bits exchanged by all parties in* BulletinBC *is $O(n^2 \cdot \kappa^2)$, with probability $1-\mathsf{negl}(\kappa)$.*

Proof. Every honest party sends at most one time for each bit to a number of $O(m) = O(\kappa)$ parties with overwhelming probability in κ, 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 $O(n^2 \cdot m \cdot \kappa) = O(n^2 \cdot \kappa^2)$. □

4 The $\mathcal{M}$-Converge Problem

In this section we introduce the $\mathcal{M}$-Converge problem, an efficient solution of which is used as a black box in our parallel broadcast protocols. Informally, in the $\mathcal{M}$-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 $\mathcal{M}$-Converge protocols in the presence of an adversary that can corrupt at most t parties, adaptively. We now define the $\mathcal{M}$-Converge problem formally.

Definition 5 (t-secure $\mathcal{M}$-Converge problem). *Let $\mathcal{M} \subseteq \{0,1\}^*$ be an efficiently recognizable set (This is a set for which membership can be efficiently decided.) A protocol Π executed by n parties where every honest party p initially holds input set $M_p \subseteq \mathcal{M}$ and constraint set $\mathcal{C}_p \subseteq \mathcal{M}$ is a* t-secure $\mathcal{M}$-Converge protocol *if all remaining honest parties upon termination, with probability $1-\mathsf{negl}(\kappa)$, output a set S_p s.t.*

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

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

whenever at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of the protocol.

We note that the trivial solution of outputting $\mathcal{M}$ does not work: We cannot necessarily enumerate the elements of $\mathcal{M}$ since we only assume that membership can be efficiently decided. An example of such a setting is where $\mathcal{M}$ consists of signatures of n parties on a bit b, computed with the parties' secret keys. In such a setting, PPT parties can easily verify membership in $\mathcal{M}$ but cannot enumerate $\mathcal{M}$, even if $\mathcal{M}$ is polynomial-sized (Note that the latter could be true depending on the signature scheme.) As we already noted in the introduction, there is a very simple t-secure $\mathcal{M}$-Converge protocol: All honest parties just send their local sets M_p to all other parties, in one round. Unfortunately, the communication complexity of this protocol is $\Omega(n^2 \cdot \min_{p\in\mathcal{H}}\{|M_p - \mathcal{C}_p|\})$, which would be prohibitively expensive. In this section we propose a protocol, $\mathcal{M}$-ConvergeRandom (see Figure 6) that uses gossiping to reduce communication to $\tilde{O}(n \cdot |\mathcal{M}| + n^2)$. First, we introduce some tools necessary for the analysis and clear exposition of our protocol, the functionality $\mathcal{F}_{\mathsf{prop}}$ and the procedure AddRandomEdgesAdaptive, required for analyzing $\mathcal{F}_{\mathsf{prop}}$.

4.1 The Procedure AddRandomEdgesAdaptive

Here we face an adaptive adversary and introduce the respective standalone randomized procedure which we call AddRandomEdgesAdaptive(Figure 3). Analyzing this process helps with proving the security of $\mathcal{M}$-ConvergeRandom.

In AddRandomEdgesAdaptive, a set of honest senders $S \subseteq H$ (H is the set of initial honest nodes) are sending a message x to the rest of the parties, by picking each party independently to be a recipient of the message with probability m/n. The set of malicious nodes are not fixed a priori and the adversary $\mathcal{A}$ decides who to corrupt *after* the edges have been placed. We define a specific set of honest parties $S_2 \subseteq H$ as the *target set* which does not contain any of the senders in S (The meaning of target set will depend on our application.) Our goal is to prove that after the adversary has finished with the adaptive corruptions, still a good amount (in particular $2 \cdot |H - S_2|$) among the remaining honest nodes of S_2 is receiving message x with overwhelming probability. Clearly if we do not confine the view of the adversary, we cannot prove anything 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 8) 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 is enforced by the implementation of our protocol later.) We formalize this intuition in Lemma 5.

Definition 6. *Let* $(G, H') \leftarrow \text{AddRandomEdgesAdaptive}_{\mathcal{A}}(V, S, H, m)$. *Let* $S_2 \subseteq H$. *Then for all* $u \in S_2$ *define random variables* $Z_u \in \{0, 1\}$ *such that* $Z_u = 1$ *if and only if* u *has nonzero degree in* G *and* $u \in H'$.

procedure $(G, H') \leftarrow \text{AddRandomEdgesAdaptive}_{\mathcal{A}}(V, S, H, m)$
Input: Set of n nodes V; sets S and H with $S \subseteq H \subseteq V$; Integer $m \leq n$.
Output: A graph G a set of nodes $H' \subseteq H$.

Let G be an empty graph with node set V;
for every node $v \in S$ **do**
  for every node $u \in V$ **do**
    Add a directed edge (v, u) to G with probability m/n;
Initialize RevealedEdges to be all edges incident to nodes in $v \in V - H$;
while $|H| \geq \epsilon \cdot n$ **do**
  $v_i \leftarrow \mathcal{A}(V, H, \text{RevealedEdges})$ such that $v_i \in H$;
  if $v_i \neq null$ **then**
    Add $\{(u, v_i) : u \in \text{in\_neighbor}(v_i)\}$ to RevealedEdges;
    $H = H - v_i$;
  else break;
Set $H' = H$;
return (G, H');

Fig. 3. The experiment with an adaptive adversary, AddRandomEdgesAdaptive.

Lemma 5. *Let $\epsilon \in (0, 1)$, $S_2 \subseteq H$ with $|H - S_2| \leq \epsilon \cdot n/3$ and $S \subseteq H - S_2$ with $|S| \geq 2|H - S_2|/3$. Let $(G, H') \leftarrow \text{AddRandomEdgesAdaptive}_{\mathcal{A}}(V, S, H, m)$. Then for $m \geq 19/\epsilon$,*

$$\Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2|\right] \geq 1 - p,$$

where $p = \max\{n \cdot e^{-4\epsilon \cdot m/45}, (\frac{e}{2})^{-\epsilon \cdot m/2}\}$.

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

$$\Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2| \;\middle|\; \mathsf{E}_{H'}\right] \geq \Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2| \;\middle|\; \mathsf{E}_{H^*}\right],$$

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

$$\Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2|\right] = \sum_{H' \in \text{supp}(\mathsf{E})} \Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2| \;\middle|\; \mathsf{E}_{H'}\right] \Pr[\mathsf{E}_{H'}]$$

$$\geq \sum_{H' \in \text{supp}(\mathsf{E})} \Pr\left[\sum_{u \in S_2} Z_u \geq 2|H - S_2| \;\middle|\; \mathsf{E}_{H^*}\right] \Pr[\mathsf{E}_{H'}] \geq \Pr\left[\sum_{u \in S_2 \cap H^*} Z_u \geq 2|H - S_2| \;\middle|\; \mathsf{E}_{H^*}\right],$$

since $S_2 \cap H^* \subseteq S_2$. The remaining proof lower-bounds the probability $\Pr[\sum_{u \in S_2 \cap H^*} Z_u \geq 2|H - S_2| \mid \mathsf{E}_{H^*}]$. This is done in three steps.

Step 1: Computing the probabilities $\Pr[Z_u = 1 \mid \mathsf{E}_{H^*}]$ for all $u \in S_2 \cap H^*$.

For $u \in S_2 \cap H^*$ let Y_u be a random variable such that $Y_u = 1$ iff u has nonzero degree in G. By definition of Z_u ($Z_u = (Y_u = 1) \cap (u \in H')$, for $u \in S_2$) we have

that, for all $u \in S_2 \cap H^*$: $Pr[Z_u = 1 \mid \mathsf{E}_{H^*}] = \Pr[Y_u = 1 \mid \mathsf{E}_{H^*}]$.
Suppose now $H^* = H - \{v_1, \ldots, v_\ell\}$, where $v_1, \ldots, v_\ell$ are nodes chosen by the adversary in Line 8 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 $u \in S_2 \cap H^*$) are independent events. Therefore for all $u \in S_2 \cap H^*$ 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 $\{Z_u\}_{u \in S_2 \cap H^*}$, conditioned on E_{H^*}, are independent. By using the above findings and by the independence of Y_u we have that for all $b_u \in \{0, 1\}$

$$\Pr\left[\left(\bigcap_{u \in S_2 \cap H^*} Z_u = b_u\right) \mid \mathsf{E}_{H^*}\right] = \Pr\left[\left(\bigcap_{u \in S_2 \cap H^*} Y_u = b_u\right) \mid \mathsf{E}_{H^*}\right]$$

$$= \Pr\left[\bigcap_{u \in S_2 \cap H^*} Y_u = b_u\right] = \prod_{u \in S_2 \cap H^*} \Pr[Y_u = b_u] = \prod_{u \in S_2 \cap H^*} \Pr[Z_u = b_u \mid \mathsf{E}_{H^*}]$$

and therefore $\{Z_u\}_{u \in S_2 \cap H^*}$ are independent conditioned on E_{H^*}.
Step 3: Applying a Chernoff bound. We consider two cases, one for $|S| > 4\epsilon \cdot n/45$ and one for $|S| \leq 4\epsilon \cdot n/45$. For $|S| > 4\epsilon \cdot n/45$, we have that for $u \in S_2 \cap H^*$

$$\Pr[Z_u = 0 \mid \mathsf{E}_{H^*}] = (1 - m/n)^{|S|} < (1 - m/n)^{4\epsilon \cdot n/45} \leq e^{-4\epsilon \cdot m/45}\,,$$

where the last step is derived from the inequality $(1 - x) \leq e^{-x}, \forall x \in \mathbb{R}$ and the fact that $(1 - m/n) > 0$ and $4\epsilon \cdot n/45 > 0$. Note that we have $|H^* - S_2| \leq |H - S_2| \leq \epsilon n/3$ and $|H^*| \geq \epsilon n$. So, by using the set equality $A = (A - B) \cup (A \cap B)$, for $A = H^*$ and $B = S_2$, we get that $|S_2 \cap H^*| \geq 2\epsilon n/3$. Also, $|H - S_2| \leq \epsilon \cdot n/3$, so it is $|S_2 \cap H^*| \geq 2|H - S_2|$. Therefore

$$\Pr\left[\sum_{u \in S_2 \cap H^*} Z_u \geq 2 \cdot |H - S_2| \mid \mathsf{E}_{H^*}\right] \geq \Pr\left[\sum_{u \in S_2 \cap H^*} Z_u = |S_2 \cap H^*| \mid \mathsf{E}_{H^*}\right]$$

$$= \Pr\left[\bigcap_{u \in S_2 \cap H^*} Z_u = 1 \mid \mathsf{E}_{H^*}\right] = 1 - \Pr\left[\bigcup_{u \in S_2 \cap H^*} Z_u = 0 \mid \mathsf{E}_{H^*}\right]$$

$$\geq 1 - \sum_{u \in S_2 \cap H^*} \Pr\left[Z_u = 0 \mid \mathsf{E}_{H^*}\right] > 1 - |S_2 \cap H^*| \cdot e^{-4\epsilon \cdot m/45}$$

$$> 1 - n \cdot e^{-4\epsilon \cdot m/45}, \text{ since } |S_2 \cap H^*| < n\,.$$

For the case $|S| \leq 4\epsilon \cdot n/45$, since Z_u ($u \in S_2 \cap H^*$) are independent random variables conditioned on E_{H^*} we can use the lower tail Chernoff bound for $Z = \sum_{u \in S_2 \cap H^*} Z_u$, i.e.,

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

Now $\mu = \mathbb{E}[Z \mid \mathsf{E}_{H^*}] = \sum_{u \in S_2 \cap H^*} \Pr[Z_u = 1 \mid \mathsf{E}_{H^*}] \geq 2\epsilon \cdot n \cdot (1-(1-m/n)^{|S|})/3$, since $|S_2 \cap H^*| \geq 2\epsilon \cdot n/3$. Also, since $|S| \geq 1$ we have

$$\mu \geq 2\epsilon \cdot n \cdot (1-(1-m/n)^{|S|})/3 \geq 2\epsilon \cdot n \cdot (1-(1-m/n))/3 = 2\epsilon \cdot m/3\,.$$

For $\delta = 1/2$ this yields $\Pr[Z < \mu/2 \mid \mathsf{E}_{H^*}] < \left(\frac{e^{-0.5}}{(0.5)^{0.5}}\right)^{2\epsilon \cdot m/3} = \left(\frac{e}{2}\right)^{-\epsilon \cdot m/3}$.
Recall however that we must bound the probability $\Pr[Z < 2|H - S_2| \mid \mathsf{E}_{H^*}]$. Therefore it is enough to also show that $\mu \geq 4 \cdot |H - S_2|$. Recall that $|S| \geq 2|H - S_2|/3$ and since $|S| \leq 4\epsilon n/45$, then $|H - S_2| \leq 2\epsilon n/15$. From $\mu \geq 2\epsilon \cdot n \cdot \left(1-(1-m/n)^{|S|}\right)/3$ and *by* $1 + x \leq e^x$ *and* $m \leq n$ we have that

$$\begin{aligned}
\mu \quad &\geq \frac{2}{3}\epsilon \cdot n \cdot (1 - e^{-\frac{m}{n}|S|}) \geq 5 \cdot |H - S_2| \cdot \left(1 - e^{\frac{-2m \cdot \epsilon |S|}{15|H-S_2|}}\right), \\
&\textit{since } (n \geq 15|H - S_2|/2\epsilon \textit{ and for } \alpha > 0,\ f(n) = n\left(1 - e^{-\frac{\alpha}{n}}\right) \nearrow \textit{ in } n) \\
&\geq 5 \cdot |H - S_2| \cdot \left(1 - e^{\frac{-4m \cdot \epsilon}{45}}\right), \ \textit{(since } |S|/|H - S_2| \geq 2/3\textit{)} \\
&> 4 \cdot |H - S_2|, \quad \textit{(since } m \geq 19/\epsilon > \frac{45 \ln 5}{4\epsilon}\textit{)}\,. \quad \square
\end{aligned}$$

4.2 The ideal functionality $\mathcal{F}_{\mathsf{prop}}$

To facilitate the exposition of our $\mathcal{M}$-ConvergeRandom protocol, we use an ideal functionality $\mathcal{F}_{\mathsf{prop}}$—see Figure 4. In summary, $\mathcal{F}_{\mathsf{prop}}$ 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 β and returns a set O_i, in the end of round β, which contains messages sent from other parties to i in the beginning of round β. 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 that 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.

Implementing the ideal functionality $\mathcal{F}_{\mathsf{prop}}$ with Propagate(). In Figure 5 we define the process that instantiates $\mathcal{F}_{\mathsf{prop}}$ 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

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 party i's input messages' set.

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 probability 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 $\mathcal{F}_{\mathsf{prop}}$.

$j \in [n]$, and adds x to $\mathcal{L}_j$ with probability m/n. All lists $\mathcal{L}_j$ are padded to the size of the maximum sized list for the current call of PROPAGATE by the party (say party p), which we call Λ_p and are encrypted using the fresh public keys (Recall that $\mathcal{M}$ is the fixed set of the $\mathcal{M}$-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 pattern of each sender does not reveal anything (irrespectively of who the recipient of x is, the adversary just sees equally-sized encrypted lists sent to all parties from each sender); and (ii) even if the adversary adaptively corrupts a party, no information about who that party sent to is revealed (because of erasure of plaintext lists and secret key)—the only information that is revealed is from whom that party received, which is harmless. We show that the lists $\mathcal{L}_j$ constructed by $\text{PROPAGATE}_p()$ is of the same order as the message list M_p of party p with overwhelming probability.

Lemma 6. *If M_p is the input set of party p, the number of elements added to list $\mathcal{L}_j$ at Line 7 of* PROPAGATE() *is $\leq |M_p| \cdot 2m/n \leq \Lambda_p$ with probability $1 - \mathsf{negl}(\kappa)$.*

Lemma 7. *Let s be the length in bits of a message in $\mathcal{M}$. The communication complexity of* PROPAGATE() *for party p with input M_p, is $O(\max\{n, |M_p|\} \cdot m \cdot s)$.*

Security of PROPAGATE(). The lemma below proves that PROPAGATE() securely instantiates $\mathcal{F}_{\mathsf{prop}}$. The key property that we leverage is that each sender sends 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. *Assuming a CPA-secure PKE scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *and secure erasure,* PROPAGATE() *t-securely computes $\mathcal{F}_{\mathsf{prop}}$ according to Definition 3.*

1: **procedure** PROPAGATE$_p$(SendRandom, M_p)
Input: A set of of messages M_p.
Output: A set of messages O_p.

2: Set $(sk_p, pk_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$; ▷ at time 0
3: SEND$(pk_p, [n])$;
4: RECEIVE(); ▷ at time Δ
5: **for** all $x \in M_p$ **do**
6: **for** $j = 1$ to n **do**
7: Add x to list $\mathcal{L}_j$ with probability m/n;
8: Let $\Lambda_p = 2m \left\lceil \frac{|M_p|}{n} \right\rceil$;
9: **for** $j = 1$ to n **do**
10: Pad list $\mathcal{L}_j$ to maximum size Λ_p;
11: $ct_j \leftarrow \mathsf{Enc}(pk_j, \mathcal{L}_j)$;
12: Erase $\mathcal{L}_j$ from memory;
13: **for** $j = 1$ to n **do**
14: SEND(ct_j, j);
15: $\mathcal{C} \leftarrow$ RECEIVE(); ▷ at time 2Δ
16: **for** all $ct \in \mathcal{C}$ **do**
17: Decrypt ct using sk_p and output a list $\mathcal{L}$;
18: Add $\mathcal{L}$ to O_p;
19: Erase sk_p from memory;
20: **return** O_p;

Fig. 5. PROPAGATE(), a secure instantiation 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.

4.3 Our $\mathcal{M}$-ConvergeRandom Protocol

We now analyze the $\mathcal{M}$-CONVERGERANDOM protocol, depicted in Figure 6, solving the $\mathcal{M}$-Converge problem with improved communication. Our protocol proceeds in $\lceil \log(\epsilon \cdot n) \rceil$ rounds. In each round, every honest party p uses $\mathcal{F}_{\mathsf{prop}}$ to send messages in their local set that are not in their constraint set, to a few randomly selected parties (Line 3). To decide what to send in the next round, p takes the union of the received messages (Line 4), and from the resulting set of messages, keeps only the ones that belong in set $\mathcal{M}$ (Line 6) and are not in p's constraint set. For example, messages $m \notin \mathcal{M}$, sent by the adversary can be safely discarded and so can any message $m \in \mathcal{C}_p$.

The proof of Lemma 9 requires defining and proving some properties of the standalone probabilistic experiment ADDRANDOMEDGESADAPTIVE (Fig. 3), which encapsulates how $\mathcal{M}$-CONVERGERANDOM works.

Lemma 9 (Propagation in presence of an adaptive adversary). *Fix an initially-honest party p and a message $\mathsf{m}^* \in M_p - \bigcup_{h \in \mathcal{H}} \mathcal{C}_h$. Then, with probability $1 - \mathsf{negl}(\kappa)$, all remaining honest parties after $\mathcal{M}$-CONVERGERANDOM terminates have received m^*.*

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

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

Fig. 6. Our $\mathcal{M}$-ConvergeRandom$_p$ protocol.

Proof. Let $h_\beta \geq \epsilon \cdot n$ be the set of parties that remain honest after the adversary has performed corruptions for round β of $\mathcal{M}$-ConvergeRandom. Thus, for all $\beta \geq 0$, $h_{\beta+1} \subseteq h_\beta$. Also, let R_β denote the set of parties that *i.* remain honest after the adversary has performed corruptions for round β; and *ii.* receive m^* for the first time during round β. Let

$$\omega_\beta = \left(\bigcup_{i=1}^{\beta} R_i\right) \cap h_\beta = (R_\beta \cap h_\beta) \cup \left(\left(\bigcup_{i=1}^{\beta-1} R_i\right) \cap h_\beta\right)$$
$$= R_\beta \cup \left(\left(\bigcup_{i=1}^{\beta-1} R_i\right) \cap h_{\beta-1} \cap h_\beta\right) = R_\beta \cup (\omega_{\beta-1} \cap h_\beta)$$

We shall use the above notation for sets interchangeably with the notation for their cardinality, which will be clear depending on the context of the operations. We first prove that with probability $1 - \mathsf{negl}(\kappa)$, for all rounds $\beta \leq \lceil \log(\epsilon \cdot n) \rceil$:

1. If $\omega_{\beta-1} \leq \epsilon \cdot n/3$, then $R_\beta \geq (2/3) \cdot \omega_\beta$ and $\omega_\beta \geq 2^\beta$
2. If $\omega_{\beta-1} > \epsilon \cdot n/3$, then $\omega_\beta = h_\beta \geq \epsilon \cdot n$

(*1.*) In $\mathcal{M}$-ConvergeRandom, a message m^* at round β is propagated in a randomized fashion using $\mathcal{F}_{\mathsf{prop}}$. So, 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 RevealedEdges. Thus, since $h_\beta \geq \epsilon \cdot n$, we can compute the number of "new" honest receivers R_β of message m in the end of round β by (almost) directly applying Lemma 5, with S_2 being $h_\beta - \omega_{\beta-1}$ (the set of honest parties that have never received m^* by that round of the protocol) and $R_{\beta-1}$ being S, i.e. the set of honest senders of m^* for round β. Also $R_0 = \omega_0 \geq 1$, since there is at least one honest sender for m^* (namely p) for round 1. We use induction. For $\beta = 1$, if $R_0 \leq \epsilon n/3$, by applying Lemma 5:

$$\Pr[R_1 \geq 2\omega_0] = \Pr[R_1 \geq 2R_0] \geq 1 - p = 1 - \mathsf{negl}(\kappa)\,,$$

where $p = \max\{n \cdot e^{-4\epsilon \cdot m/45}, (\frac{e}{2})^{-\epsilon \cdot m/2}\}$ and since $m = \Theta(\kappa)$ and $\kappa = \mathsf{poly}(n)$. Then, we have that $\omega_1 = R_1 + \omega_0 \geq 3\omega_0 \geq 2$, with probability $1 - \mathsf{negl}(\kappa)$. Also,

$$\frac{\omega_1}{R_1} = \frac{R_1 + \omega_0 \cap h_0}{R_1} = \frac{R_1 + \omega_0}{R_1} \leq \frac{R_1 + R_1/2}{R_1} = 3/2,$$

thus $R_1 \geq (2/3)\omega_1$ with probability $1 - \mathsf{negl}(\kappa)$. Therefore, the base case holds. Let us assume the claim holds for round $\beta - 1 \leq \lceil \log(\epsilon \cdot n) \rceil - 1$. Then, if $\omega_{\beta-1} \leq \epsilon n/3$, we prove that $R_\beta \geq (2/3) \cdot \omega_\beta$ and $\omega_\beta \geq 2^\beta$. By $\mathcal{M}$-ConvergeRandom, all honest receivers in round $\beta - 1$ become senders in round β. So, for applying Lemma 5,we match the sets as follows: $h_{\beta-1} - \omega_{\beta-1} := S_2, h_{\beta-1} := H, h_\beta := H', R_{\beta-1} := S$. Notice that the $u \in S_2$ s.t. $Z_u^{(\beta)} = 1$ are exactly the $u \in R_\beta$ and thus $\sum_{u \in S_2} Z_u^{(\beta)} = R_\beta$. Also, notice that

$$H - S_2 = h_{\beta-1} - (h_{\beta-1} - \omega_{\beta-1}) = h_{\beta-1} \cap \omega_{\beta-1} = \omega_{\beta-1},$$

meaning: $|H - S_2| = \omega_{\beta-1} \leq \epsilon n/3$ and $S = R_{\beta-1} \geq 2\omega_{\beta-1}/3 \geq 2|H - S_2|/3$.

Finally, we have $m = \Theta(\kappa) \geq 19/\epsilon = \Theta(1)$. Therefore, we can use Lemma 5 for round β: Thus, $\Pr[R_\beta \geq 2|H - S_2|] = \Pr[R_\beta \geq 2\omega_{\beta-1}] \geq 1 - p = 1 - \mathsf{negl}(\kappa)$. Therefore, $\omega_\beta \geq R_\beta \geq 2\omega_{\beta-1} \geq 2 \cdot 2^{\beta-1} = 2^\beta$, with probability $1 - \mathsf{negl}(\kappa)$. Also,

$$\frac{\omega_\beta}{R_\beta} = \frac{R_\beta + \omega_{\beta-1} \cap h_\beta}{R_\beta} \leq \frac{R_\beta + \omega_{\beta-1}}{R_\beta} \leq \frac{R_\beta + R_\beta/2}{R_\beta} = 3/2,$$

thus $R_\beta \geq (2/3)\omega_\beta$, with probability $1 - \mathsf{negl}(\kappa)$, which completes the proof.

(*2.*) For every party $p_i : i = 1, \ldots, n$ we define Bernoulli R.V.s $Z_i^{(\beta)} = 1$ if and only if party p_i received m^* at round β. If $\beta = 1$ and $\omega_0 > \epsilon n/3$, then from the first round there are $\geq \epsilon n/3$ honest senders for m^*. We bound the probability

$$\Pr[\bigcup_{i=1}^{n} \{Z_i^{(1)} = 0\}] \leq \sum_{i=1}^{n} \Pr\left[Z_i^{(1)} = 0\right] = n \cdot \left(1 - \frac{m}{n}\right)^{R_0} \leq n \cdot \left(1 - \frac{m}{n}\right)^{\epsilon n/3}$$
$$= n \cdot \left[\left(1 - \frac{m}{n}\right)^{n/m}\right]^{\epsilon m/3} \leq n \cdot e^{-\epsilon m/3} = \mathsf{negl}(\kappa).$$

Now, if $\beta > 1$, then there was some round $\beta^* \leq \beta$ s.t. $\beta^* \stackrel{\text{def}}{=} \arg\min_{i \geq 0}\{\omega_i > \epsilon n/3\}$. If $\beta^* = 0$, we already showed that all parties receive m^* in the first round. Thus, for all subsequent rounds i, $\omega_i = h_i$. Else, if $\beta^* > 0$, by definition $\omega_{\beta^*-1} \leq \epsilon n/3$. So, we can apply (*1.*), meaning that $R_{\beta^*} \geq 2\omega_{\beta^*}/3 \geq 2\epsilon n/9$. Accordingly, for round β^*, we can again bound the probability

$$\Pr[\bigcup_{i=1}^{n} \{Z_i^{(\beta^*)} = 0\}] \leq n \cdot \left[\left(1 - \frac{m}{n}\right)^{n/m}\right]^{2\epsilon m/9} \leq n \cdot e^{-2\epsilon m/9} = \mathsf{negl}(\kappa),$$

So, for all subsequent rounds $i \geq \beta^*$, $\omega_i = h_i$.

The results from (*1.*),(*2.*) combined mean that, if the protocol runs for more than $\lceil \log \epsilon n/3 \rceil$ rounds without reaching case (*2.*), it is definite that, after applying (*1.*) for round $\beta = \lceil \log(\epsilon n/3) \rceil$, case (*2.*) will hold for the next round. Therefore, with probability $1 - \mathsf{negl}(\kappa)$, all remaining honest parties after $\mathcal{M}$-ConvergeRandom terminates have received the message m^*. □

Theorem 2. *Protocol* $\mathcal{M}$-ConvergeRandom *from Figure 6 is an adaptively t-secure* $\mathcal{M}$*-Converge protocol for all* $t < (1 - \epsilon) \cdot n$ *and fixed* $\epsilon \in (0, 1)$*. The number of bits sent by one party is*

$$O(n \log n \cdot m \cdot s + \sum_{i=1}^{\lceil \log \epsilon n \rceil} |M_p^{(i)} - \mathcal{C}_p^{(i)}| \cdot m \cdot s),$$

where s is the length in bits of a message in $\mathcal{M}$ and $M_p^{(i)} - \mathcal{C}_p^{(i)}$ denotes the set given as input to the i-th call of PROPAGATE.

Proof. Security follows by applying Lemma 9 for all initially-honest parties and for all of their initial messages. Every message m^* is delivered to all honest parties that remain after the termination of the protocol, as required by Definition 5. The communication complexity (CC) is dominated by the calls to $\mathcal{F}_{\mathsf{prop}}$. Each call to $\mathcal{F}_{\mathsf{prop}}$ is securely instantiated by executing PROPAGATE with the same input set (Lemma 8). By Lemma 7, the i-th such call to PROPAGATE invokes a total of $O(\max\{n, |M_p^i - \mathcal{C}_p^i|\} \cdot m \cdot s) = O(n \cdot m \cdot s + |M_p^{(i)} - \mathcal{C}_p^{(i)}| \cdot m \cdot s)$ communication, where $M_p^i - \mathcal{C}_p^i$ is the corresponding set of messages input to PROPAGATE at that call. Thus, the number of bits sent by one party is

$$\sum_{i=1}^{\lceil \log \epsilon n \rceil} O(n \cdot m \cdot s + |M_p^{(i)} - \mathcal{C}_p^{(i)}| \cdot m \cdot s) = O(n \log n \cdot m \cdot s + \sum_{i=1}^{\lceil \log \epsilon n \rceil} |M_p^{(i)} - \mathcal{C}_p^{(i)}| \cdot m \cdot s)$$

with probability $1 - \mathsf{negl}(\kappa)$. □

4.4 An Extension: The $\mathcal{M}$-DistinctConverge Protocol

Recall that the $\mathcal{M}$-Converge problem was defined with respect to a message set $\mathcal{M}$. For achieving PBC with low communication complexity in the trusted PKI setting (see Section 6), we need a slightly different version of the $\mathcal{M}$-Converge problem, namely the $\mathcal{M}$-DistinctConverge problem, defined with respect to a parameter k. Now, 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 $\mathcal{M}$-DistinctConverge.

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 $\mathcal{M}$-DistinctConverge problem.

Definition 8 (t-secure $\mathcal{M}$-DistinctConverge protocol). *Let $\mathcal{M} \subseteq \{0,1\}^*$ be an efficiently recognizable set. A protocol Π executed by n parties, where every honest party p initially holds input set $M_p \subseteq \mathcal{M}$ and constraint set $\mathcal{C}_p \subseteq \mathcal{M}$, is a* t-secure $\mathcal{M}$-DistinctConverge protocol *if all remaining honest parties upon termination, with probability $1 - \mathsf{negl}(\kappa)$, output a set*

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

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

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

Fig. 7. Our $\mathcal{M}\text{-DistinctCV}_p$ protocol.

when at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of the protocol.

Figure 7 shows $\mathcal{M}$-DistinctCV, a modification of $\mathcal{M}$-ConvergeRandom.

Theorem 3. *Let $k > 0$. Protocol $\mathcal{M}$-DistinctCV from Figure 7 is an adaptively t-secure $\mathcal{M}$-DistinctConverge protocol for all $t < (1 - \epsilon) \cdot n$ and fixed $\epsilon \in (0, 1)$. The total number of bits sent by all parties is*

$$\tilde{O}(n \cdot \max\{n, |\mathtt{distinct}_k(\mathcal{M})|\} \cdot m \cdot s)\,.$$

5 Parallel Broadcast in the Bulletin PKI Model

We present a protocol to achieve PBC in the Bulletin PKI model. This lack of trusted setup induces additional difficulty to the problem. Still, the proposed protocol uses communication cubic in n. We describe the high-level idea of the protocol. Each party internally maintains state for every signature. Each triplet (b, s, j) (for $b \in \{0, 1\}, s, j \in [n]$) defines a specific signature, so there could be $2 \cdot n^2$ signatures overall, each of size κ (where κ denotes the security parameter). The propagation of a specific signature $\sigma_j := \mathsf{sig}_j(b, s)$ is independent of whether b was added to $\mathsf{Extracted}_i^s$ at that given round. Instead, whenever a party i observes a new signature, i.e. $\mathsf{sig}_j(b, s)$ for some (b, s, j) that i never observed before, they proceed to propagate the signature exactly twice. If the signature was observed during the subround of some $\mathcal{M}$-ConvergeRandom, then they continue propagating it only during the next subround (due to how $\mathcal{M}$-ConvergeRandom updates C_p). They also initiate the next round's $\mathcal{M}$-ConvergeRandom with the signature being on their input set but not on their constraint set. Notice that $\mathcal{M}$-ConvergeRandom is defined with respect to the set $\mathcal{M}$. Here, we fix $\mathcal{M}$ to consist of all of the parties' valid signatures for messages $[b, s]$, where b is a bit and s a slot in $[n]$. Next, we prove the security of our protocol.

Lemma 10 (t-consistency: BulletinPBC). BulletinPBC *satisfies consistency (Def. 2) in the $(\mathcal{F}_{\mathsf{prop}})$-hybrid world, with probability $1 - \mathsf{negl}(\kappa)$.*

```
1: procedure {b_1, ..., b_n} ← BulletinPBC_p(b_p)
       Input: Local bit b_p.
       Output: Decision bits b_1, ..., b_n.
   ▷ Fix set M = {sig_v([b, s]) such that b ∈ {0, 1}, s, v ∈ [n]}

2:     Extracted_p^i = ∅, for i = 1, ..., n;                    ▷ global variable
3:     Local_p = ∅;                                             ▷ global variable
4:     Send(sig_p([b_p, p]), [n]);
5:     for round r = 1 to t + 1 do                  ▷ t: bound on dishonest parties
6:         AddMySignature_p(r);
7:         ForwardSignatures_p(r);
8:     for slot i = 1, ..., n do
9:         return b_i ∈ Extracted_p^i if |Extracted_p^i| = 1, else return canonical bit 0;

1: procedure AddMySignature_p(r)

2:     Local_p ← Local_p ∪ Receive();
3:     for all [b, s] such that b ∉ Extracted_p^s do
4:         if Local_p contains ≥ r valid signatures on [b, s] (including sig_s([b, s])) then
5:             Add b to Extracted_p^s;
6:             Add sig_p([b, s]) to Local_p;

1: procedure ForwardSignatures_p(r)

2:     if r ≤ t then
3:         Let C_p contain all signatures that p has propagated exactly twice via
       ConvergeRandom;
4:         Local_p ← M-ConvergeRandom(Local_p, C_p);
```

Fig. 8. The protocols for PBC in the Bulletin-PKI. The main PBC protocol BulletinPBC invokes the procedures AddMySignature and ForwardSignatures. The later calls a $\mathcal{M}$-Converge protocol, i.e. $\mathcal{M}$-ConvergeRandom, in order for each honest party to convey its internal view to all other parties.

Proof. Suppose for some slot s, an honest party p adds bit b to $\mathsf{Extracted}_p^s$ at some round r. We prove that by the end of the protocol, all honest parties j add b to their $\mathsf{Extracted}_j^s$ sets with probability at least $1 - \mathsf{negl}(\kappa)$. We distinguish the following cases according to the round 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.)

If $r \leq t$, then p has at least r distinct, valid signatures for (b, s) and also p creates its own signature. So, p observed each of these $\geq r+1$ signatures for the first time at some round $r-k$ where $k \in \{0, \ldots, r-1\}$. For every signature σ with $k \geq 1$, p called a $\mathcal{M}$-ConvergeRandom with σ in its initial input set M_p and not in its constraint set C_p. (Suppose not. Then σ is observed during some $\mathcal{M}$-ConvergeRandom by p, sent once with Propagate and never sent again by p, a contradiction, since only signatures sent twice go in $\mathcal{C}_p$.) For the signatures with $k = 0$, p initiates a $\mathcal{M}$-ConvergeRandom during round r. From Theorem 2, by round $r+1$ every honest party j has $\geq r+1$ valid signatures

for (b, s) and adds b to their $\mathsf{Extracted}_j^s$ set during AddMySignature$_j(r+1)$.

Else, if $r > t$ then, p observes $t+1$ valid signatures for (b, s). At least one of the signatures comes from an honest party h. So, h has added b to $\mathsf{Extracted}_h^s$ by some round $r' \leq t$. Thus, for honest party h the case $r' \leq t$ can be applied, and so all honest parties j add b to their $\mathsf{Extracted}_j^s$ sets by round $r'+1$. □

Lemma 11 (t-validity: BulletinPBC). BulletinPBC *satisfies validity (Definition 2), in the* $(\mathcal{F}_{\mathsf{prop}})$*-hybrid world with probability* $1 - \mathsf{negl}(\kappa)$.

Proof. Follows directly from the proof of consistency (Lemma 10) and the idealized signature scheme. Every honest sender s with input bit b_s, executes Line 4 and thus sends to all parties their signature for (b_s, s) at the start of the protocol. So, all honest parties h add b_s to their $\mathsf{Extracted}_h^s$ sets. By the idealized signature scheme, no other bit b' could bare a valid signature from honest sender s, thus no other bit b' could be in an h's $\mathsf{Extracted}_h^s$ set. □

Theorem 4 (Communication Complexity of BulletinPBC). *The total number of bits exchanged by all parties in* BulletinPBC *is* $\tilde{O}(n^3 \cdot \kappa^2)$.

Proof. The communication complexity (CC) of BulletinPBC is dominated by the CC of $\mathcal{M}$-ConvergeRandom. By Theorem 2 each such call invokes $O(n \log n \cdot ms + \sum_{i=1}^{\lceil \log \epsilon n \rceil} |M_p^{(i)} - C_p^{(i)}| \cdot ms)$ communication for each party, where $M_p^{(i)} - C_p^{(i)}$ is the corresponding set of messages input to Propagate. The sets input to Propagate by the protocol are sets containing signatures of size κ each. During each super-round j of the protocol, $\mathcal{M}$-ConvergeRandom is called once, with input set $M_p^{(j)} - C_p^{(j)}$. Due to $\mathcal{M}$-ConvergeRandom, when a message is input by party p to Propagate at some subround, it is subsequently added to C_p and thus won't be input to Propagate again during that super-round. Let for a fixed party p, $I_p^{(j,i)}$ denote the input set $M_p - C_p$ for its call of $\mathcal{M}$-ConvergeRandom at subround i of round j. In BulletinPBC, parties only propagate signatures they have propagated less than twice. Thus, each message can be propagated by the same party at most twice meaning that, for each party p, during all sub-rounds of all super-rounds of the protocol, it holds that $\sum_{j=1}^{(t+1)} \sum_{i=1}^{\lceil \log \epsilon n \rceil} |I_p^{(j,i)}| \leq 2|\mathcal{M}| = O(n^2)$.
We can count the overall CC of each party during the protocol, so we have

$$\mathsf{CC}(p) = \sum_{j=1}^{(t+1)} O(n \log n \cdot m\kappa + \sum_{i=1}^{\lceil \log \epsilon n \rceil} |I_p^{(j,i)}| \cdot m\kappa) = O(\sum_{j=1}^{(t+1)} \sum_{i=1}^{\lceil \log \epsilon n \rceil} |I_p^{(j,i)}| \cdot m \cdot \kappa)$$
$$+O(n^2 \log n \cdot m \cdot \kappa) = \tilde{O}(n^2 \cdot \kappa^2),$$

where, we used the fact that $\sum_{j=1}^{(t+1)} \sum_{i=1}^{\lceil \log \epsilon n \rceil} |I_p^{(j,i)}| = O(n^2)$.Thus, the CC of $\mathcal{M}$-ConvergeRandom for all parties is $\tilde{O}(n^3 \cdot \kappa^2)$. □

6 Parallel Broadcast in the Trusted PKI Model

In this section we use trusted setup to reduce the communication complexity of PBC from cubic to quadratic. Our new protocol TrustedPBC uses protocol

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. 9. Functionality $\mathcal{F}_{\mathsf{mine}}$.

$\mathcal{M}$-DistinctCV from Section 4.4. To facilitate the exposition and our proof, TrustedPBC in Figure 10 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 9), which was also presented in Chan et al. [8] and was shown to be instantiable from standard assumptions (with setup) by [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 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. Next, we provide some definitions and necessary results.

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

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 13. *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 $s \in [n]$. □

procedure $\{b_1, \ldots, b_n\} \leftarrow \text{TRUSTEDPBC}_p(b_p)$
Input: Local bit b_p.
Output: Decision bits $b_1, \ldots, b_n$.

$\mathsf{Extracted}_p^i = \emptyset$, for $i = 1, \ldots, n$; ▷ global variable
$\mathsf{Voted}_p^i = \emptyset$, for $i = 1, \ldots, n$; ▷ global variable
$\mathsf{Local}_p = \emptyset$; ▷ global variable
$\text{SEND}(\mathsf{sig}_p([b_p, p]), [n])$;
for round $r = 1$ to $R + 1$ **do** ▷ R is also a global variable
  $\text{DISTRIBUTE}_p(r)$;
  $\text{VOTE}_p(r)$;
for slot $i = 1, \ldots, n$ **do**
  return $b_i \in \mathsf{Extracted}_p^i$ if $|\mathsf{Extracted}_p^i| = 1$ else **return** canonical bit 0;

procedure $\text{DISTRIBUTE}_p(r)$

$\mathsf{Local}_p \leftarrow \mathsf{Local}_p \cup \text{RECEIVE}()$;
Let $\mathcal{V} = \{v_i\}$ be valid r-batches (in Local_p) on $\{[b_i, s_i]\}$ s.t. $b_i \notin \mathsf{Extracted}_p^{s_i}$;
for all $v_i \in \mathcal{V}$ **do** Add b_i to $\mathsf{Extracted}_p^{s_i}$;
if $r \leq R$ **then** $\mathsf{Local}_p \leftarrow \mathcal{M}_r\text{-DISTINCTCV}_p(\mathcal{V}, \emptyset, k^*)$;

procedure $\text{VOTE}_p(r)$

if $r \leq R$ **then**
  Let $\mathcal{V} = \{v_i\}$ be valid r-batches (in Local_p) on $\{[b_i, s_i]\}$ s.t. $b_i \notin \mathsf{Voted}_p^{s_i}$;
  for all $v_i \in \mathcal{V}$ s.t. $\mathcal{F}_{\mathsf{mine}}(\mathsf{Mine}, b_i, s_i) = 1$ **do**
    Add b_i to $\mathsf{Voted}_p^{s_i}$;
    Add b_i in $\mathsf{Extracted}_p^{s_i}$ if $b_i \notin \mathsf{Extracted}_p^{s_i}$;
    Extend v_i to a valid $(r+1)$-batch v_i' by adding p's signature on $[b_i, s_i]$;
    $\text{SEND}(v_i', [n])$;

Fig. 10. TRUSTEDPBC calls two procedures, i.e. DISTRIBUTE and VOTE. During each execution, DISTRIBUTE calls the $\mathcal{M}_r$-DISTINCTCV protocol, for each honest party to convey its internal view to all other parties. $\mathcal{M}_r$ follows Def. 11.

Our protocol (see Figure 10) is inspired by the single-sender protocol of Chan et al. [8] (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.) 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 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, $\mathsf{sig}_i([b_i, i])$. Afterwards, the distribution and voting 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 $\mathsf{Extracted}$ sets. Instead of every node using a $\mathsf{SEND\text{-}ALL}$ to send each v_i (as ChBC would do), all nodes run $\mathcal{M}_r$-DistinctCV from Figure 7 ($\mathcal{M}_r$ defined per Def. 11), with their initial constraint sets $\mathcal{C}_p$ empty. $\mathcal{M}_r$-DistinctCV assures that all parties eventually see the other parties' inputs but since there is overlap between input messages, it uses less communication.
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 via $\mathcal{F}_{\mathsf{mine}}$. If they are, they add their own signature to extend the valid r-batch to a valid $(r+1)$-batch.

Lemma 14 (t-consistency). *Let $R = 2\kappa/\epsilon$.* TrustedPBC *satisfies t-consistency, per Definition 2, in the $(\mathcal{F}_{\mathsf{mine}}, \mathcal{F}_{\mathsf{prop}})$-hybrid world with probability $1 - \mathsf{negl}(\kappa)$.*

Lemma 15 (t-validity). *Let $R = 2\kappa/\epsilon$.* TrustedPBC *satisfies t-validity, per Definition 2, in the $(\mathcal{F}_{\mathsf{mine}}, \mathcal{F}_{\mathsf{prop}})$-hybrid world with probability $1 - \mathsf{negl}(\kappa)$.*

Theorem 5 (Communication). *Let $R = 2\kappa/\epsilon$. The total number of bits exchanged by all parties in* TrustedPBC *is $\tilde{O}(n^2 \cdot \kappa^4)$ with probability $1 - \mathsf{negl}(\kappa)$.*

Acknowledgements

This research was supported in part by the National Science Foundation, VMware, the Ethereum Foundation and Protocol Labs.

References

1. Ittai Abraham, T.-H. Hubert Chan, Kartik Nayak Danny Dolev, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proc. of ACM Symposium on Principles of Distributed Computing (PODC)*, pages 317–326, 2019.
2. Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round mpc with identifiable abort and public verifiability. In *Proc. of CRYPTO*, volume 12171, pages 562–592, 2020.
3. Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In *Proc. of TCC*, volume 12550, pages 353–380, 2020.
4. Edward Bortnikov, Maxim Gurevich, Idit Keidar, Gabriel Kliot, and Alexander Shraer. Brahms: byzantine resilient random membership sampling. In *Proc. of ACM PODC*, pages 145–154, 2008.
5. Elette Boyle, Ran, and Aarushi Goel. Breaking the $o(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. In *Proc. of ACM PODC*, pages 319–330, 2021.
6. Vitalik Buterin. A guide to 99% fault tolerant consensus. Blog Post, 2018.
7. Ran Canetti. Security and composition of multiparty cryptographic protocols. In *Journal of Cryptology*, volume 13, pages 143–202, 2000.

8. T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. In *Proc. of IACR International Conference on Public-Key Cryptography (PKC)*, volume 12111, pages 246–265, 2020.
9. Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of ACM PODC*, pages 1–12, 1987.
10. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
11. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proc. of ACM STOC*, pages 148–161, 1988.
12. Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *Proc. of International Symposium on Distributed Computing (DISC)*, volume 5805, pages 449–463, 2009.
13. Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *Proc. of IEEE FOCS*, pages 658–668, 2007.
14. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement in $t + 1$ rounds. In *Proc. of ACM STOC*, pages 31–41, 1993.
15. Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. Scalable Byzantine Reliable Broadcast. In *Proc. of DISC*, volume 146, pages 22:1–22:16, 2019.
16. Zygmunt J Haas, Joseph Y Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on networking*, pages 479–491, 2006.
17. Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. In *Proc. of ACM PODC*, pages 420–429, 2010.
18. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proc. of ACM-SIAM SODA*, pages 990–999, 2006.
19. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *Transactions on Programming Languages and Systems*, 4:382–401, 1982.
20. Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In *Proc. of ACM PODC*, pages 303–312, 2014.
21. Dahlia Malkhi, Yishay Mansour, and Michael K Reiter. On diffusing updates in a byzantine environment. In *Proc. of the IEEE Symposium on Reliable Distributed Systems*, pages 134–143, 1999.
22. Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. *Cryptology ePrint Archive*, 2022.
23. Silvio Micali. Very simple and efficient byzantine agreement. In *Proc. of ITCS*, volume 67, pages 6:1–6:1, 2017.
24. Silvio Micali and Vinod Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. Technical report, MIT, 2017.
25. Atsuki Momose and Ling Ren. Optimal Communication Complexity of Authenticated Byzantine Agreement. In *Proc. of DISC*, pages 32:1–32:16, 2021.
26. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27:228–234, 1980.
27. Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In *Proc. of TCC, Part I*, volume 12550, pages 381–411, 2020.