Communication Depth in Multivalued Broadcast

Gabriel Dettling[3], Martin Hirt[1], and Chen-Da Liu-Zhang[2]

[1] ETH Zurich
[2] Lucerne University of Applied Sciences and Arts
[3] Lucerne University of Applied Sciences and Arts**

Available at https://eprint.iacr.org/2025/931

Abstract. Synchronous protocols are a fundamental and widely studied model in distributed computing and cryptography. In this model, protocols proceed in rounds: at the beginning of each round, parties send messages, and all messages are guaranteed to be delivered before the round ends.
The standard communication complexity of a synchronous protocol measures the total number of bits transmitted throughout the execution. However, this metric does not accurately reflect the protocol's actual running time, since it ignores the inherent parallelism of synchronous communication. In particular, multiple parties may transmit simultaneously within the same round, and the duration of a round is determined not by the total communication volume, but by the longest transmission that must occur sequentially. To capture this aspect of synchronous computation, we introduce *communication-depth*, a new complexity measure for synchronous protocols. Informally, communication-depth quantifies the amount of communication that lies on the protocol's critical path, namely, the number of bits that must be transmitted sequentially and therefore determine the protocol's execution time. The communication-depth of a protocol is defined as the sum, over all rounds, of the cost of each round.
We consider two variants of round-cost capturing different types of parallel communication the network allows. In the *simultaneous-parties* model, the cost of a round is the maximum number of bits any party sends or receives during that round. In the *simultaneous-channels* model, the cost of a round is the maximum number of bits transmitted over any single channel during the round.
We demonstrate the usefulness of communication-depth by fully characterizing the asymptotic complexity of multivalued synchronous broadcast. Perhaps surprisingly, while protocols for ℓ-bit inputs with asymptotic optimal communication $O(\ell n)$ have been known for a long time, current protocols only achieve communication-depth $\Theta(\ell n)$ in the simultaneous-parties model and $\Theta(\ell)$ in the simultaneous-channels model.
Our results provide a complete characterization of multivalued broadcast under communication-depth for both variants. On the negative side, we show that when $n - t = \mathcal{O}(1)$, any protocol tolerating up to t corrupt parties requires communication-depth $\Omega(\ell n)$ in the simultaneous-parties model and $\Omega(\ell)$ in the simultaneous-channels model. On the positive side, we show that when $n - t = \Theta(n)$, multivalued broadcast can be achieved with communication-depth $\Theta(\ell)$ in the simultaneous-parties model and $\Theta(\ell/n)$ in the simultaneous-channels model.

1 Introduction

1.1 Practical Efficiency of Synchronous Protocols

The efficiency of a synchronous protocol is usually expressed in terms of its *round complexity* and its *communication complexity*. The round complexity denotes the

** Part of the work was done while the author was at ETH Zurich.

number of communication rounds the protocol requires, and the communication complexity denotes the number of bits that need to be sent in an execution of the protocol. However, what matters in practice is the execution time of a protocol, i.e. the sum of the durations of all rounds. The duration of every round must be long enough to ensure all messages which an honest party may send within it will arrive before the start of the next round. This duration depends on the round-trip time of the network (hence the significance of the round complexity) and (assuming some constant bandwidth) it depends on the sizes of the messages which honest parties might send in this round.

Beyond Communication Complexity. To illustrate why the communication complexity fails to account for this, consider for example the following two protocols: The first protocol, π_1, consists of n rounds, where in the i-th round party P_i sends some huge message to his right neighbor P_{i+1}[4], and all other parties send to each other a single bit. The second protocol π_2, also consists of n rounds, where in the first round, every party P_i sends some huge message to his right neighbor P_{i+1}, and in all other rounds, each party sends to each other party a single bit. Both protocols π_1 and π_2 have the same round complexity, the same communication complexity, and the same overall communication complexity per party. Still, in many real-life networks, protocol π_2 performs much better than protocol π_1. The reason for this is that in protocol π_2, all the huge messages can be transmitted *simultaneously* in Round 1, so only this round needs to be long. All other rounds can be very short (only single bits are transmitted). Moreover, even if in π_1 some of the large messages could be conditionally omitted, the protocol's performance wouldn't necessarily improve. This is due to the fact that parties must remain synchronized, and they must wait long enough to guarantee any transmitted messages arrive in time. Thus, unless it is common knowledge among all parties that the large message in some round can be skipped, they must wait long enough for it to arrive.

1.2 Contributions

Efficiency Metrics. We introduce a new a class of protocol parameters, called the *communication-depth* of a protocol, which — together with the round complexity — better represent the execution time of a protocol. Depending on what kind of parallel communication the network allows, a different one of these new parameters is relevant. Intuitively, the communication-depth of a protocol corresponds to the number of bits that must be sent in sequence and cannot be parallelized in the given network type. All parameters have in common that the communication-depth of a protocol is the sum of the communication-depths of each round, as in the synchronous model, a new protocol round can be started only once the current round is finished for all parties.

For networks that allow different parties to transmit messages simultaneously (e.g., parties are connected via a router to the Internet), we define the *simultaneous-parties communication-depth* of a protocol π, denoted as $D_{\mathrm{sp}}(\pi)$,

[4] more precisely: To $P_{(i \bmod n)+1}$.

to be the sum over all rounds of the maximum number of bits any party needs to send or receive in that round. For networks that allow simultaneous transmissions through all bilateral channels (e.g., parties are bilaterally connected with dedicated network cables), we define the *simultaneous-channels communication-depth* of a protocol π, denoted as $D_{\text{sc}}(\pi)$, to be the sum over all rounds of the maximal number of bits that need to be transmitted between any two parties in that round. In all these definitions, we assume that the bandwidth of the network is fixed (but not necessarily known).

Application: Multi-Valued Broadcast with Optimal Communication-Depth. In a broadcast protocol, a sender P_s wants to distribute a message m among a set of recipients $\mathcal{P} = \{P_1, \ldots, P_n\}$ in such a way that all recipients obtain the same message, even if the sender and/or up to t of the recipients are corrupted. In the multi-valued variant of broadcast, the message m is assumed to be long; we denote its bit-length by ℓ, and assume that $\ell \gg n$. Such protocols for long messages are of special interest, since they are used as core building blocks in the design of numerous distributed protocols, including multi-party computation [Yao82; GMW87; BGW88; CCD88; RB89] and blockchains [Nak08; But+13] (for example, to distribute large blocks).

As each of the n recipients needs to learn the ℓ-bit message, any multi-valued broadcast protocol must communicate at least $\Omega(\ell n)$ bits overall. This lower bound, while not straightforward to match has been (asymptotically) achieved in all relevant models through a significant line of work called *extension protocols*, which construct multivalued broadcast with a small usage of broadcast for short messages [FH06; HR14; CO18; Nay+20; GP21].

However, when looking at communication-depth, all known multi-valued broadcast protocols which tolerate a corrupt majority require at least $D_{\text{sp}} = \Omega(\ell n)$ and $D_{\text{sc}} = \Omega(\ell)$, which leads to the question of whether this is optimal.

What is the optimal communication-depth for multivalued broadcast?

We give a complete characterization of multi-valued broadcast in terms of each of the introduced communication-depth metrics.

Feasibility. We construct multi-valued broadcast protocols that not only offer an optimal communication complexity $\mathcal{O}(\ell n)$, but also achieve optimal communication-depth $D_{\text{sp}} = \mathcal{O}(\ell)$ (when parties can communicate simultaneously), respectively $D_{\text{sc}} = \mathcal{O}(\ell/n)$ (when bilateral channels can be used simultaneously). That means that the actual execution time of our protocols is $\Omega(n)$ times lower than the execution time of the state of the art, over the same network.

Our protocols tolerate $t < (1 - \varepsilon)n$ corruptions (for an arbitrary constant ε), which we prove to be optimal for $D_{\text{sp}} = \mathcal{O}(\ell)$. We provide one protocol with cryptographic security, and one protocol with information-theoretic security.

Theorem 1. *Let κ be a security parameter and $\epsilon > 0$ some constant. For up to $t < (1 - \epsilon)n$ corruptions, broadcast of an ℓ-bit message can be achieved:*

– *assuming collision-resistant hash functions and a PKI for signatures, with a simultaneous-channels communication-depth of* $\mathcal{O}(\frac{\ell}{n} + n^3\kappa)$*, a simultaneous-parties communication-depth of* $\mathcal{O}(\ell + n^4\kappa)$ *and communication complexity* $\mathcal{O}(n\ell + n^5\kappa)$*.*
– *assuming a precomputation phase with a broadcast channel, with information theoretic security and a simultaneous-channels communication-depth of* $\mathcal{O}(\frac{\ell}{n} + n^6\kappa)$*, a simultaneous-parties communication-depth of* $\mathcal{O}(\ell + n^7\kappa)$ *and communication complexity* $\mathcal{O}(n\ell + n^8\kappa)$*.*

Note that while our protocols do not use signatures directly, the PKI / precomputation assumptions are used when instantiating the short-message broadcast protocols [DS83; PW96] and have been shown to be necessary [PSL80].

Impossibility. We complement our feasibility result by showing lower bounds on the introduced communication-depth metrics. The lower bounds show that our multi-valued broadcast protocols achieve the optimal simultaneous-channels and simultaneous-parties communication-depth, in the regime where $t = \theta(n)$ and $t \geq n/2$.

Theorem 2. *Let* t *and* n *be such that* $t \geq n/2$*. Any* ℓ*-bit broadcast protocol* π *for* n *parties secure against* t *corruptions requires* $D_{\text{sp}}(\pi) \geq \ell\lfloor\frac{n}{n-t}\rfloor$*, and* $D_{\text{sc}}(\pi) \geq \lceil\frac{\ell}{n-t}\rceil$*.*

1.3 Related Work

Broadcast Protocols There is an extensive literature on broadcast protocols. The first extension protocol that leveraged broadcast for short messages into broadcast for long messages was due to Turpin and Coan [TC84], which achieved $O(\ell n^2)$ communication for ℓ-bit inputs, and resilience up to $t < n/3$ corruptions. After that, many other such extension protocols appeared in the literature [FH06; LV11b; PR11; HR14; CO18; Nay+20; GP21; Bha+22], achieving the optimal communication complexity of $O(\ell n)$ bits, for different corruption regimes, computational assumptions and setups. In addition, previous research focused on minimizing other complexity metrics such as e.g. the round complexity or the seed-round complexity and seed-communication complexity [GP21]. However, while optimal simultaneous-parties and simultaneous-channels communication-depth has been achieved in the honest majority case (as we discuss below), to the best of our knowledge, no existing protocol achieves optimal simultaneous-parties communication-depth or optimal simultaneous-channels communication-depth in the corrupt majority regime.

Honest Majority. The broadcast protocol by Chen [Che21] achieves both optimal simultaneous-parties communication-depth and optimal simultaneous-channels communication-depth while tolerating up to $t < n/3$ corruptions. For $t < n/2$, the same can be achieved e.g. using the consensus protocol by Nayak et al. [Nay+20] or a modified version of the consensus protocol by Fitzi and Hirt [FH06] or Ganesh and Patra [GP21]. To this end, one would need to use protocols based on Reed-Solomon codes (such as eg. the protocols `Encode`, `Distribute` and `Reconstruct`

from [Nay+20]) for the initial message dissemination (and, in the case of [FH06] and [GP21], also for transmission of some messages within the protocol) as sending the entire message over one channel already incurs suboptimal simultaneous-channels communication-depth ℓ.

Corrupt Majority. We briefly discuss why the mentioned protocols for the corrupt majority setting do not achieve optimal communication-depth. Many of these, e.g. [HR14; GP21; CO18] work by dividing the message into blocks and using some form of conflict tracking to determine which blocks should be sent between which parties. In the case where there are no corruptions, this usually boils down to requiring P_s to send the entire message to every other party. This already incurs suboptimal simultaneous-parties communication-depth $\Omega(\ell n)$ and suboptimal simultaneous-channels communication-depth $\Omega(\ell)$. Notably, the protocol by Nayak et al. [Nay+20] does not follow this approach as it uses Reed-Solomon codes instead of transmitting the message directly. However, this is not sufficient for achieving optimal communication-depth in and of itself, and neither is simply modifying the protocols from [HR14; GP21; CO18] in the manner we outlined for the honest majority case. This is due to the fact that achieving optimal communication-depth requires balancing communication across all channels in every round. Previous protocols allow for adversary strategies that prevent this, either by forcing an honest party to send large amounts of data within a single round (e.g. [GP21]), or by forcing honest parties to wait and do nothing within many rounds (e.g. [HR14; CO18; Nay+20]).

Table 1. The table includes the most relevant synchronous broadcast protocols along with their corruption resilience t and each of the communication-depth metrics. All protocols in this table have communication complexity $\mathcal{O}(\ell n)$. An asterisk indicates that modifications are needed to achieve the stated result.

	Threshold	Security	D_{sp}	D_{sc}
Chen21 [Che21]	$t < n/3$	perfect	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell/n)$
Nay$^+$20 [Nay+20]	$t < n/2$	cryptogr.	$\mathcal{O}(\ell)^*$	$\mathcal{O}(\ell/n)^*$
FH06 [FH06]	$t < n/2$	inf.-theor.	$\mathcal{O}(\ell)^*$	$\mathcal{O}(\ell/n)^*$
CO18 [CO18]	$t < n$	inf.-theor.	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell)$
GP21 [GP21]	$t < n$	cryptogr.	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell)$
Nay$^+$20 [Nay+20]	$t < (1-\varepsilon)n$	cryptogr.	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell)$
This Paper	$t < (1-\varepsilon)n$	cryptogr.	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell/n)$
This Paper	$t < (1-\varepsilon)n$	inf.-theor.	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell/n)$

Balanced Communication The *balanced communication cost* of a protocol refers to the maximal communication cost any party incurs over the entire protocol [KS11; BCG21; GK24; FGK24; Blu+23]. This is related to, but distinct

from our notion of simultaneous-parties communication-depth, which amounts to taking the sum of the balanced communication costs *within each round*. Thus, any protocol with $D_{\text{sp}} \leq C$ also has a balanced communication cost of at most C, but the converse does not hold in general. For example, the protocols π_1, π_2 outlined in section 1.1 have equal balanced communication cost, but different simultaneous-parties communication-depth. For another example, consider the broadcast protocol for $t < (1 - \varepsilon)n$ corruptions from [Nay+20] which has balanced communication of $\mathcal{O}(\ell)$ but simultaneous-parties communication-depth of $\mathcal{O}(\ell n)$.

Congested Clique Model A line of work [Lot+03; DKO14; Cen+15; KS18] investigates the so-called *congested clique model*, where the size of any message is restricted to some constant b (typically $b = \mathcal{O}(\log n)$), and only b bits may be sent between any two parties in any given round. This is equivalent to requiring that every round has $D_{\text{sc}} \leq b$. We generalize this model by allowing the communication depth to vary across rounds and introducing the communication depth as an efficiency metric.

Network-Aware Broadcast In [LV11a; LV12], broadcast protocols for $t < n/3$ corruptions are introduced which are designed to have high throughput when used repeatedly with a common set of corrupt parties.

Pipes Model A concurrent work [LNS25] introduced the *pipes model*, a model for analyzing latency and throughput in state machine replication (SMR) protocols. The paper follows the same motivation as ours and focuses on a metric similar to our definition of simultaneous-parties communication-depth, where the latency of a round is a function of the parties' bandwidth. Their protocols are designed for SMR in the $t < n/3$ regime, whereas we consider synchronous Byzantine broadcast in the dishonest-majority regime.

2 The Communication Depth of a Protocol

We define the *communication-depth* of a protocol to be the total number of bits that may need to be transmitted sequentially. This depends on the type of the network, namely which channels can be used simultaneously. We distinguish two types of networks:

- An independent-parties network, where all parties can communicate simultaneously, but every party can only send or receive one message at a time. A typical example of such a network is the Internet, where each party is connected via a (bandwidth-limited) uplink to a network.
- A fully-bilateral network, where any two parties can communicate with each other simultaneously. An example of such a network is when parties are bilaterally connected with dedicated network cables.

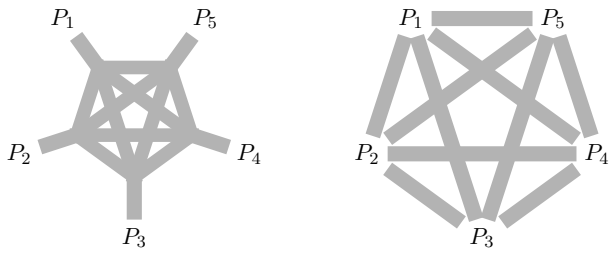

Fig. 1. An independent-parties and a fully-bilateral network.

For each network type, we define the corresponding communication depth. Note that in Def. 1, we are first considering the maximal communication depth any round may have (depending on inputs, randomness, adversary) and only then taking the sum over all rounds. This leads to a rather strong notion of communication depth, similar to the conservative message complexity defined in [Dam+15]. One might also consider weaker versions by first taking the sum over all rounds, and only then considering the maximal, or even the expected value of the overall sum. However, as mentioned in Sec. 1.1, such a relaxation would not necessarily be more relevant. Indeed, since parties must remain synchronized, conditionally transmitting less data in some rounds is only useful if all honest parties are able to agree on it in advance or have access to some synchronization primitive.

Definition 1. *Consider a synchronous n-party protocol π. For an execution of π with inputs $\mathbf{x}$, randomness $\mathbf{r}$ and adversary $\mathcal{A}$, write $\ell_{i,j}^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A})$ to denote the number of bits which π requires to be sent from P_i to P_j in round r. We define ...*

– *... the* simultaneous-parties communication-depth *of π as*

$$D_{\mathrm{sp}}(\pi) := \sum_r \max_{\mathbf{x},\mathbf{r},\mathcal{A}} \max_{i \in [n]} \{\ell_i^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A})\}$$

where

$$\ell_i^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A}) = \sum_j \ell_{i,j}^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A}) + \ell_{j,i}^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A})$$

– *... the* simultaneous-channels communication-depth *of π as*

$$D_{\mathrm{sc}}(\pi) := \sum_r \max_{\mathbf{x},\mathbf{r},\mathcal{A}} \max_{i,j \in [n]} \left\{\ell_{i,j}^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A}) + \ell_{j,i}^{(r)}(\pi, \mathbf{x}, \mathbf{r}, \mathcal{A})\right\}$$

where the maximum is taken over all possible inputs $\mathbf{x}$, random tapes $\mathbf{r}$ and all relevant adversaries $\mathcal{A}$, i.e. PPT adversaries in the computational setting, and possibly unbounded in the information-theoretic setting.

Trivially, we have that $D_{\mathrm{sp}} \leq nD_{\mathrm{sc}}$, and that the communication complexity is at most nD_{sp}.

3 High-Level Overview of Our Multi-Valued Broadcast

On a high level, our protocols follow the following message propagation mechanism: First, the sender P_s sends the message m to all parties. Some parties claim to have received the message, others claim not to have received the message. Then, those parties who have received the message forward it to those parties who have not received it, and so on. This defines a sequence of sets $S_0, S_1, S_2, \ldots$, where $S_0 = \{P_s\}$ and S_k are those parties who have received the message in the k-th step.

The propagation is such that in the k-th step, essentially only parties in S_{k-1} and S_k communicate. Also, using error-correction codes we achieve that each party in $S_{k-1} \cup S_k$ only sends $n\lceil \frac{|m|}{n-t} \rceil$ bits overall, equally distributed over all outgoing channels. During the propagation, corrupted parties can provoke errors, which dynamically changes the sequence of sets $S_0, S_1, S_2, \ldots$.

However, we do not directly propagate the message itself (otherwise, most of the time, most of the channels would be unused). Instead, the original message m is cut into blocks $m_1, m_2, \ldots$ of equal length, and each block is propagated separately (but not at all independently). The propagation of block m_1 starts in round 1, the propagation of block m_2 starts in round 2, etc. Furthermore, we keep the sequence of sets $S_0, S_1, S_2, \ldots$ identical for all blocks, which implies that each block is at a different position in this sequence. For example, in round 5, the block m_1 is propagated from S_4 to S_5, the block m_2 is propagated from S_3 to S_4, the block m_3 is propagated from S_2 to S_3, etc. This ensures that in any round, every party is involved in the propagation of at most 2 blocks, which nicely balances all available channels.

The following figure illustrates the simultaneous propagation of the blocks:

Table 2. The message m is divided into blocks $m_1, m_2 \ldots$. The figure includes the mechanics on how the blocks are propagated across the rounds.

	Rd 1	Rd 2	Rd 3	Rd 4	Rd 5	...
Block m_1	$\{P_s\}$	S_1	S_2	S_3	S_4	S_5
Block m_2		$\{P_s\}$	S_1	S_2	S_3	S_4
Block m_3			$\{P_s\}$	S_1	S_2	S_3
Block m_4				$\{P_s\}$	S_1	S_2
.....						

Using some extended analysis inspired by the Dispute Control framework [BH06], we can bound the number of sets S_k to $\frac{2n}{n-t}$, and hence set the number of blocks to $\frac{2n}{n-t}$. In the cryptographic protocol, we use collision-resistant hashes (broadcasted by the sender right at the beginning of the protocol) to decide whether a

received message is correct or not. In the protocol with statistical security, we use universal hash functions instead, with the additional challenge that the sequence $S_0, S_1, S_2, \ldots$ changes over time.

4 Preliminaries

Model We consider a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ which are able to communicate over a network of pairwise authenticated channels. We assume this network is *synchronous*, i.e. we assume that all parties have access to a common clock and that for each round there is a known upper bound on the delay incurred by computation and message transmission. Note that this upper bound may vary between rounds. We consider *static byzantine corruption*, i.e. we assume that before a protocol starts, an adversary chooses a set of at most t corrupted parties and exerts full control over them during protocol execution. Parties that are not corrupted are called *honest*. We provide protocols with *information-theoretic* security, i.e. guarantees always hold with high probability as well as *cryptographic* security, i.e. guarantees hold with high probability when the adversary is computationally bounded.

Primitives We introduce the primitives that are used in our protocols.

Broadcast. We provide a formal definition for broadcast, which allows a sender to distribute consistently a message to a set of parties.

Definition 2 (Broadcast). *An interactive protocol π with a single designated input-holding party P_s is a t-secure* ***broadcast protocol*** *if the following properties hold whenever the adversary corrupts up to t parties:*

1. **(Validity)** *If P_s is honest with input m, then all honest parties output m.*
2. **(Consistency)** *All honest parties output the same value.*
3. **(Termination)** *All honest parties terminate.*

Hash Functions In the context of cryptographic security, we write $\texttt{CRHash} : \{0,1\}^* \to \{0,1\}^\kappa$ to denote a collision resistant hash function, i.e. we assume the probability that the adversary is able to find $m \neq m'$ with $\texttt{CRHash}(m) = \texttt{CRHash}(m')$ to be negligible in κ. In the context of information theoretic security, we write $\{\texttt{ITHash}_\alpha\}_{\alpha \in \mathcal{K}}$ for an ϵ-universal family of hash functions for some ϵ which is negligible in κ. This means that if $m \neq m'$, then the probability that for $\alpha \in \mathcal{K}$ chosen uniformly at random we have $\texttt{ITHash}_\alpha(m) = \texttt{ITHash}_\alpha(m')$ is at most ϵ. Such a hash function can for instance be constructed as follows (taken from [HR14]): Take $\mathcal{X} = \{0,1\}^\ell$ as the domain, and set $\mathcal{Y} = \mathcal{K} = \mathrm{GF}(2^\kappa)$. We can then define $\texttt{ITHash}_\alpha : \mathcal{X} \to \mathcal{Y}$ as $\texttt{ITHash}_\alpha(m) = f_m(\alpha)$ where f_m is obtained by viewing m as a polynomial over $\mathrm{GF}(2^\kappa)$ of degree $\leq \lceil \frac{\ell}{\kappa} \rceil - 1$, adding padding as needed. Any two distinct polynomials of degree $\lceil \frac{\ell}{\kappa} \rceil - 1$ can coincide in at most $\lceil \frac{\ell}{\kappa} \rceil - 1$ points, which implies that for distinct $m, m' \in \mathcal{X}$ and for $\alpha \in \mathcal{K}$ chosen uniformly at random, we have

$$\Pr[\texttt{ITHash}_\alpha(m) = \texttt{ITHash}_\alpha(m')] \leq \lceil \frac{\ell}{\kappa} \rceil 2^{-\kappa}$$

Erasure Codes We use Reed-Solomon Codes [RS60] as erasure codes. Given any value $m \in \{0,1\}^\ell$ and $d \geq 1$, we view m as a sequence of d elements of $\mathrm{GF}(2^{\lceil \ell/d \rceil})$ (adding padding as needed). Write f_m for the polynomial whose coefficients are given by this sequence, and $(m_1, \ldots, m_n) = \texttt{RS-Enc}_d(m)$ for the values obtained by evaluating f_m on n different predefined points in $\mathrm{GF}(2^{\lceil \ell/d \rceil})$. Since f_m has degree at most $d-1$, any d of the m_i are sufficient to reconstruct m.

5 Dispute Control Framework

Overview In this section, we present the *Dispute Control Framework* used in our protocols, which combines the approach from [BH06] with an improved way of inferring a set $\Delta^* \subseteq \binom{\mathcal{P}}{2}$ of effective disputes from the set $\Delta \subseteq \binom{\mathcal{P}}{2}$ of observed disputes (using the technique from [Wan+20]). We then introduce a distance function d_{Δ^*} on $\mathcal{P}$, which depends on the current set of effective disputes Δ^*, and we show that d_{Δ^*} only takes values in $[0, \frac{2n}{n-t}] \cup \{+\infty\}$.

Dispute Control The Dispute Control Framework works by having all parties keep track of a set $\Delta \subseteq \binom{\mathcal{P}}{2}$. Initially, we set $\Delta = \emptyset$, and whenever a party P_i claims that another party P_j is corrupted, a conflict is registered by adding $\{P_i, P_j\}$ to Δ. Since no honest party would ever accuse another honest party of being corrupted, Δ is guaranteed to always be valid.

Definition 3. *A set of conflicts $\Delta \subseteq \binom{\mathcal{P}}{2}$ is called **valid** if there exist no two honest parties P_i and P_j with $\{P_i, P_j\} \in \Delta$.*

In [BH06], the validity of Δ is used to show that any party which is in conflict with more than t parties must be corrupted. Such parties are then inferred to be in conflict with all other parties. We use a stronger way of inferring conflicts: If two parties taken together are in conflict with more than t parties, one of the two must be corrupt[5] and we infer a conflict between them. These inferred conflicts are then recursively used to infer even more conflicts.

Definition 4. *Given a set of conflicts $\Delta \subseteq \binom{\mathcal{P}}{2}$ and a party $P_i \in \mathcal{P}$, write*

$$\Delta_{P_i} = \{P_j \in \mathcal{P} \mid \{P_i, P_j\} \in \Delta\}$$

for the set of parties which are in conflict with P_i.

Definition 5. *Given a set of conflicts $\Delta \subseteq \binom{\mathcal{P}}{2}$, define the set of **directly inferred conflicts** Δ' as*

$$\Delta' = \Delta \cup \{\{P_i, P_j\} : |\Delta_{P_i} \cup \Delta_{P_j}| > t\}$$

[5] This could in principle be replaced by an even stronger rule: If Δ is valid, any pair of parties either contains a corrupted party or it is contained in the set $\mathcal{H}$ of honest parties, which fulfills $|\mathcal{H}| \geq n - t$ and $\Delta \cap \binom{\mathcal{H}}{2} = \emptyset$. One might therefore infer a conflict between any pair of parties that is not contained in a set with these two properties. Unfortunately, the problem of computing the inferred conflicts from inputs n, t, Δ then becomes NP-Hard (the k-Clique decision problem can be reduced to this). In contrast, computing Δ^* as defined in Def. 5 can be easily done in polynomial time.

and the set of ***effective conflicts*** Δ^* *as*

$$\Delta^* = \lim_{k\to\infty} \Delta_k, \text{ where } \Delta_0 = \Delta \text{ and } \Delta_{k+1} = (\Delta_k)' \text{ for all } k \in \mathbb{N}.$$

Proposition 1. *If a set of conflicts* Δ *is valid, then* Δ^* *is valid.*

Proof. Assume that Δ is valid. For any two honest parties P_i and P_j we have $\Delta_{P_i} \cup \Delta_{P_j} \subseteq \mathcal{C}$, where $\mathcal{C}$ denotes the set of corrupted parties. This implies $|\Delta_{P_i} \cup \Delta_{P_j}| \leq |\mathcal{C}| \leq t$ and thus $\{P_i, P_j\} \notin \Delta'$. Therefore, Δ' is valid and, by induction, so are all the Δ_k. Since $\binom{\mathcal{P}}{2}$ is finite, we have $\Delta^* = \Delta_k$ for some $k \in \mathbb{N}$. □

Distance If two parties P and P' are in conflict, we know one of them is corrupt and therefore there is no point in directly sending messages between them. Any communication from P to P' will need to be redirected through other parties. To quantify the degree of indirection, we define the Δ^**-distance*:

Definition 6. *The* Δ^*-distance *between* P *and* P'*, denoted by* $d_{\Delta^*}(P, P')$*, is the distance between* P *and* P' *in the graph* $G := (\mathcal{P}, \binom{\mathcal{P}}{2} \setminus \Delta^*)$.

We show that this distance cannot grow very much before P and P' become disconnected and any communication between them is impossible:

Lemma 1. *Let* $\Delta \subseteq \binom{\mathcal{P}}{2}$*, and* $P, P' \in \mathcal{P}$*. We either have* $d_{\Delta^*}(P, P') \leq \frac{2n}{n-t}$ *or* $d_{\Delta^*}(P, P') = +\infty$

Proof. Write $d := d_{\Delta^*}(P, P')$ and assume $0 < d < +\infty$. By renumbering the parties we can assume that $P = P_0, P_1, \ldots, P_d = P'$ is a shortest path from P to P' in the graph $G := (\mathcal{P}, \binom{\mathcal{P}}{2} \setminus \Delta^*)$. Consider the sets $\mathcal{U}_j := N_G(P_{j-1}) \cap N_G(P_j)$ for $j \in \{1, \ldots, d\}$. Note that for nonconsecutive j_1, j_2 we have $\mathcal{U}_{j_1} \cap \mathcal{U}_{j_2} = \emptyset$, as otherwise our shortest path could be made even shorter. Moreover, by the definition of G we have $\mathcal{U}_j = \mathcal{P} \setminus ((\Delta^*)_{P_{j-1}} \cup (\Delta^*)_{P_j})$. As was argued in the proof of Prop. 1, we have $\Delta^* = \Delta_k$ for some $k \in N$ and therefore $(\Delta^*)' = \Delta^*$. It follows that $|(\Delta^*)_{P_{j-1}} \cup (\Delta^*)_{P_j}| \leq t$ and thus $|\mathcal{U}_j| \geq n - t$ for all $j \in \{1, \ldots, d\}$.

We get

$$n = |\mathcal{P}| \geq |\bigsqcup_{j=1}^{\lceil \frac{d}{2} \rceil} \mathcal{U}_{2j-1}| \geq \frac{d}{2}(n-t)$$

which implies $d_{\Delta^*}(P, P') = d \leq \frac{2n}{n-t}$. □

6 Broadcast with Cryptographic Security

We first present the protocol `CryptoPropagate`. Given a message which has already been broadcasted towards a subset of all parties, `CryptoPropagate` attempts to broadcast it towards more parties. The Dispute Control Framework presented in Sec. 5 is used to ensure progress (i.e. parties which fail to receive the message will incur additional disputes). We then present our protocol `CryptoBroadcast` which uses repeated application of `CryptoPropagate` to achieve broadcast.

6.1 Cryptographic Propagation

The protocol `CryptoPropagate` works on the assumptions that the message m has been broadcasted towards all parties in the set $H_k(\Delta^*) := \{P \in \mathcal{P} \mid d_{\Delta^*}(P_s, P) \leq k\}$[6] for some k, and that all honest parties agree on hash values $h_1, \ldots, h_n$ with $\texttt{CRHash}(m_i) = h_i$ for $i = 1, \ldots, n$. Here, $(m_1, \ldots, m_n) = \texttt{RS-Enc}_{n-t}(m)$ denotes a Reed-Solomon encoding of m into n shares of size $\lceil \ell/(n-t) \rceil$ each, and `CRHash` denotes some collision resistant hash function.

The goal of the protocol then is for the parties in $S_k(\Delta^*) := \{P \in \mathcal{P} \mid d_{\Delta^*}(P_s, P) = k\}$ to transmit m towards all parties in $S_{k+1}(\Delta^*)$. This is done in two steps, first a distribution step ensuring that each $P_i \in S_{k+1}(\Delta^*)$ knows m_i, and then a reconstruction step ensuring that each $P_i \in S_{k+1}(\Delta^*)$ knows enough shares to reconstruct m. Note that there is no guarantee that any party actually learns m. All we show is that parties which are in $S_{k+1}(\Delta^*)$ at the beginning of the protocol either learn m or will be placed outside of $H_{k+1}(\Delta^*)$ at the end of the protocol due to newly arisen disputes.

In the following, we'll call candidate values for m or for m_i *good* or *bad* depending on whether they are consistent with all of the hash values $h_1, \ldots, h_n$, i.e. m'_i is good iff $\texttt{CRHash}(m'_i) = h_i$ and m' is good iff all of the $(m'_1, \ldots m'_n) = \texttt{RS-Enc}_{n-t}(m')$ are good.

Protocol 1 `CryptoPropagate`

Inputs: All parties: $h_1, \ldots, h_n, \Delta, k$. Parties in $S_k(\Delta^*)$: m.

1. Each $P_i \in S_k(\Delta^*)$ computes $(m_1, \ldots, m_n) = \texttt{RS-Enc}_{n-t}(m)$, and sends m_j to each $P_j \in S_{k+1}(\Delta^*)$ with $\{P_i, P_j\} \notin \Delta^*$.
2. Each $P_j \in S_{k+1}(\Delta^*)$ computes and broadcasts an accusation vector A_j, where $A_j[i] = 1$ if and only if the share received from P_i in round 1 is bad (does not match h_j).
   For each i, j with $A_j[i] = 1$, all parties add the conflict $\{P_i, P_j\}$ to Δ.
3. Each $P_i \in S_k(\Delta^*) \cup S_{k+1}(\Delta^*)$ sends m_i to each $P_j \in S_{k+1}(\Delta^*)$ with $\{P_i, P_j\} \notin \Delta^*$.
4. Each $P_j \in S_{k+1}(\Delta^*)$ checks which of the shares m_i received in round 3 are good.
   - If less than $n - t$ shares are good, then P_j broadcasts an accusation vector $\tilde{A}_j$, where $\tilde{A}_j[i] = 1$ if and only if the share received from P_i is bad (does not match h_i).
   - If at least $n - t$ shares are good, then P_j uses those to reconstruct m'. Then P_j checks whether m' is good (by computing the shares $m'_1, \ldots, m'_n$ of m' and checking them against the hashes $h_1, \ldots, h_n$). If m' is good, then P_j sets $m = m'$. Otherwise, clearly all parties in $S_k(\Delta^*)$ are corrupted, and P_j broadcasts the according accusation vector $\tilde{A}_j$ with $\tilde{A}_j[i] = 1$ if and only if $P_i \in S_k(\Delta^*)$.

   For each i, j with $\tilde{A}_j[i] = 1$, all parties add the conflict $\{P_i, P_j\}$ to Δ.

Output: All parties: Δ. Parties in $S_{k+1}(\Delta^*)$: m

[6] see Sec. 5 for definitions of Δ^*, d_{Δ^*}

Proposition 2. *`CryptoPropagate` achieves message propagation with cryptographic security, i.e. assuming that no hash collisions are found, that all parties agree on $h_1, \dots h_n$, that Δ is valid and that all honest parties in $H_k(\Delta^*)$ know a good value m, the protocol will return a valid $\tilde{\Delta} \supseteq \Delta$ such that all honest parties in $H_{k+1}(\tilde{\Delta}^*)$ know a good value m.*

Proof. Assume all the conditions given above are satisfied. We first show that there are no conflicts between honest parties, i.e. if Δ_r denotes the set of detected conflicts after round r, we show that $\Delta_1, \dots, \Delta_4$ are all valid. For Δ_1, this is clear as we assume that the protocol starts with a valid set of conflicts. For $\Delta_2 = \Delta_3$, we need to consider the accusations broadcasted in round 2: Since all honest $P_i \in S_k(\Delta_1^*)$ know good values for all m_j, no honest $P_j \in S_{k+1}(\Delta_1^*)$ will accuse another honest party in round 2. For Δ_4, we show that no honest P_i is accused by another honest party in round 4. There are two cases: (1) If $P_i \in S_{k+1}(\Delta_3^*)$, then by $S_{k+1}(\Delta_3^*) \subseteq S_{k+1}(\Delta_1^*)$,[7] P_i must have accepted the share m_i from at least one party during the first two rounds. Therefore P_i knows a good value for m_i which all honest recipients will accept. (2) If $P_i \in S_k(\Delta_3^*)$ then it follows from $S_k(\Delta_3^*) \subseteq H_k(\Delta_1^*)$ that P_i knows a good value for m (and thus also for all m_j). Assuming that no hash collisions were found, it is impossible for an honest P_j to reconstruct a bad value for m in round 4. Therefore, no honest P_j will accuse P_i in round 4. This proves that Δ_4 is valid as well.

Finally, we show that after the protocol, every honest $P_i \in H_{k+1}(\Delta_4^*)$ knows a good value for m. If $P_i \in H_k(\Delta_4^*)$, we have $P_i \subseteq H_k(\Delta_1^*)$ and there is nothing to prove. If $P_i \in S_{k+1}(\Delta_4^*)$, there exists some $P_j \in S_k(\Delta_4^*)$ with $\{P_j, P_i\} \notin \Delta_4^*$. By the argument used in the proof of Lemma 1, the set $\mathcal{U} = \mathcal{P} \setminus ((\Delta_4^*)_{P_i} \cup (\Delta_4^*)_{P_j})$ contains at least $n - t$ parties. All of these were at distance k or $k+1$ from P_s during the entire protocol, and P_i never rejected a message from any of them. This means that in round 3, P_i received enough good messages to reconstruct m, and since P_i did not reject the message received from P_j, the value for m which P_i reconstructed was good. □

6.2 Cryptographic Broadcast

Our broadcast protocol `CryptoBroadcast` works by having P_s broadcast the hash values required for `CryptoPropagate`. Then, propagation steps are applied to every block until all parties either know all blocks or are disqualified from the broadcast protocol. This is done in such a way that the propagation steps can be parallelized efficiently.

[7] $S_{k+1}(\Delta_3^*) \subseteq S_{k+1}(\Delta_1^*)$ follows from the facts that the Δ^*-distance between parties can only grow as new conflicts are added and that in round two, only conflicts between parties in $S_k(\Delta_1^*)$ and in $S_{k+1}(\Delta_1^*)$ are added.

Protocol 2 `CryptoBroadcast`

Initialize $\Delta = \emptyset$.

0. P_s divides their input message $m = m^{(1)}|\ldots|m^{(q)}$ into q blocks of length ℓ/q each, computes the shares $(m_1^{(k)}, \ldots, m_n^{(k)}) = \texttt{RS-Enc}_{n-t}(m^{(k)})$ of each block k, and broadcasts $h_i^{(k)} = \texttt{CRHash}(m_i^{(k)})$ for each $k = 1, \ldots, q$ and $i = 1, \ldots, n$.

Repeat the following for $r = 1, \ldots$

r: All Parties run $\Delta^{(k)} \leftarrow \texttt{CryptoPropagate}(h_1^{(k)}, \ldots, h_n^{(k)}, \Delta, r-k, m^{(k)})$ in parallel for each $k \in \{1, \ldots, q\}$ with $k \leq r$ and for which there exists $P \in \mathcal{P}$ with $r - k < d_{\Delta^*}(P_s, P) < +\infty$.
If there is no such k, the protocol is terminated.
Otherwise, Δ is replaced by the union of all the $\Delta^{(k)}$ and the protocol proceeds to round $r+1$.

Output: If $d_{\Delta^*}(P_s, P_i) < +\infty$, P_i reconstructs and outputs m, otherwise P_i outputs $\perp$.

Lemma 2. *Assume that no hash collisions are found. Then for all $r > 0$, we have that after round r of `CryptoBroadcast`, Δ is valid and for all $k = 1, \ldots, q$, every honest party $P \in H_{r-k+1}(\Delta^*)$ knows a good value for $m^{(k)}$.*

Proof. For $r = 0$, this is clear. Let $r > 0$ and assume that the lemma is true for $r-1$. Then all assumptions outlined in Prop. 2 are fulfilled for every instance of `CryptoPropagate` that is executed during round r. By Prop. 2, all the $\Delta^{(k)}$ are valid, which implies that Δ is also valid after round r. Moreover, $\Delta^{(k)} \subseteq \Delta$ implies $d_{\Delta^{(k)*}}(P_s, P) \leq d_{\Delta^*}(P_s, P)$, and thus for all k corresponding to an instance of `CryptoPropagate` that was executed in round r, we have that every honest party $P \in H_{r-k+1}(\Delta^*)$ knows a good value for $m^{(k)}$. For all other values of k, we either have $k \geq r+1$, in which case only $P = P_s$ can possibly fulfill $d_{\Delta^*}(P_s, P) \leq r - k + 1 \leq 0$, or we have that after round $r-1$ there was no P with $r - k < d_{\Delta^*}(P_s, P) < +\infty$. In this case, every honest party P either already knows a good value for $m^{(k)}$, or P fulfilled $d_{\Delta^*}(P_s, P) = +\infty$ after round $r-1$. Since no conflict is ever removed from Δ, this holds after round r too. By induction, the lemma holds for every r. □

Proposition 3. *`CryptoBroadcast` achieves broadcast for up to $t < n$ corruptions with cryptographic security, and if $q = \frac{2n}{n-t}$ its simultaneous-channels communication-depth is at most*

$$\mathcal{O}\left(\frac{\ell}{n-t} + \frac{n^2}{n-t}\left(\mathcal{B}(\kappa) + \mathcal{B}(n)\right)\right)$$

where $\mathcal{B}(k)$ denotes the simultaneous-channels communication-depth of the underlying broadcast protocol when used with k-bit input.

Proof. By Lemma 1, the protocol terminates after at most $q + \frac{2n}{n-t}$ rounds. Consistency follows from Lemma 2 applied to the last round: Since Δ^* is valid, either every honest party P or no honest party P fulfills $d_{\Delta^*}(P_s, P) < +\infty$. If the latter

is the case, all honest parties know good values for every block and — assuming no hash collisions are found — output the same value m. Validity follows from the same argument, using the fact that an honest P_s broadcasts the correct hash values in round 0.

The simultaneous-channels communication-depth of round 0 is $nq\mathcal{B}(\kappa)$. In every other round, the propagation steps involve two rounds with a message of size $\mathcal{O}(\frac{\ell}{(n-t)q})$ being transmitted between (some) pairs of parties, as well as two rounds in which (some) parties are required to broadcast up to n bits. In total, we get a simultaneous-channels communication-depth of at most

$$\mathcal{O}\left(nq\mathcal{B}(\kappa) + \left(q + \frac{2n}{n-t}\right)\left(\frac{\ell}{(n-t)q} + n\mathcal{B}(n)\right)\right)$$

With $q = \frac{2n}{n-t}$, this simplifies to

$$\mathcal{O}\left(\frac{\ell}{n-t} + \frac{n^2}{n-t}\left(\mathcal{B}(\kappa) + \mathcal{B}(n)\right)\right)$$

□

By using Dolev-Strong [DS83], which has a simultaneous-channels communication-depth of $\mathcal{O}(\ell n + \kappa n^2)$ as our broadcast oracle, we get the following:

Theorem 3. *Let $\epsilon > 0$. For up to $t < (1-\epsilon)n$ corruptions, cryptographically secure broadcast of an ℓ-bit message can be achieved with a simultaneous-channels communication-depth of at most $\mathcal{O}(\frac{\ell}{n} + n^3\kappa)$, a simultaneous-parties communication-depth of at most $\mathcal{O}(\ell + n^4\kappa)$ and a communication complexity of at most $\mathcal{O}(n\ell + n^5\kappa)$, assuming a PKI for signatures.*

Proof. Follows from Prop. 3 and from $D_{\text{seq}} \leq nD_{\text{sp}} \leq n^2 D_{\text{sc}}$. □

7 Broadcast with Information-Theoretic Security

Our information-theoretic broadcast protocol is structured in a similar way to the cryptographic version. However, we use a universal family $\{\texttt{ITHash}_\alpha\}_{\alpha\in\mathcal{K}}$ of hash functions rather than the collision resistant hash function `CRHash`. Such a family of hash functions can be constructed as follows (taken from [HR14]): Take $\mathcal{X} = \{0,1\}^\ell$ as the domain, and set $\mathcal{Y} = \mathcal{K} = \text{GF}(2^\kappa)$. We can then define $\texttt{ITHash}_\alpha : \mathcal{X} \to \mathcal{Y}$ as $\texttt{ITHash}_\alpha(m) = f_m(\alpha)$ where f_m is obtained by viewing m as a polynomial over $\text{GF}(2^\kappa)$ of degree $\leq \lceil\frac{\ell}{\kappa}\rceil - 1$, adding padding as needed. Any two distinct polynomials of degree $\lceil\frac{\ell}{\kappa}\rceil - 1$ can coincide in at most $\lceil\frac{\ell}{\kappa}\rceil - 1$ points, which implies that for distinct $m, m' \in \mathcal{X}$ and for $\alpha \in \mathcal{K}$ chosen uniformly at random, we have $\Pr[\texttt{ITHash}_\alpha(m) = \texttt{ITHash}_\alpha(m')] \leq \lceil\frac{\ell}{\kappa}\rceil 2^{-\kappa}$.

7.1 Information Theoretic Propagation

The information-theoretic propagation protocol `ITPropagate` is analogous to `CryptoPropagate`, with the main difference being that we use `ITHash` instead

of `CRHash`. Consequently, new keys have to be chosen and the corresponding hash values broadcasted whenever some message needs to be verified. We'll call candidate values for the block m or for one of the m_i *α-good* if they are consistent with all hash values broadcasted by P_s in round 2 of `ITPropagate`, and *β-good* if they are consistent with all hash values broadcasted by P_s in round 5 of `ITPropagate`. We'll also refer to these hash values as *α-hashes* or *β-hashes* depending on the round in which they were broadcasted.

Protocol 3 `ITPropagate`
Inputs: All parties: Δ, k. Parties in $H_k(\Delta^*)$: m

1. Each $P_i \in S_k(\Delta^*)$ computes $(m_1, \dots m_n) = \texttt{RS-Enc}_{n-t}(m)$ and sends m_j to each $P_j \in S_{k+1}(\Delta^*)$ with $\{P_i, P_j\} \notin \Delta^*$
2. Each $P_j \in S_{k+1}(\Delta^*)$ chooses and broadcasts a random $\alpha_j \in \mathcal{K}$.
   P_s then broadcasts $\texttt{ITHash}_{\alpha_j}(m_i)$ for each $i = 1, \dots, n$ and for each α_j.
3. Each $P_i \in H_k(\Delta^*) \setminus \{P_s\}$ checks if their value of m is α-good. If not, they broadcast a bit indicating that they reject the α-hashes.
   Each $P_i \in S_{k+1}(\Delta^*)$ computes and broadcasts a rejection vector R_i, where $R_i[j] = 1$ if and only if P_i received an α-bad share from P_j in round 1.
   For every i, j for which (A1) or (B1) holds, the conflict $\{P_i, P_j\}$ is added to Δ.
   (A1) $P_i, P_j \in H_k(\Delta^*)$ and exactly one of P_i, P_j rejected the α-hashes
   (B1) We have $R_j[i] = 1$ but P_i did not reject the α-hashes.
4. Each $P_i \in S_k(\Delta^*) \cup S_{k+1}(\Delta^*)$ sends m_i to each $P_j \in S_{k+1}(\Delta^*)$ with $\{P_i, P_j\} \notin \Delta^*$
5. Each $P_j \in S_{k+1}(\Delta^*)$ chooses and broadcasts a random $\beta_j \in \mathcal{K}$.
   P_s then broadcasts $\texttt{ITHash}_{\beta_j}(m_i)$ for each $i = 1, \dots, n$ and for each β_j.
6. Each $P_i \in H_k(\Delta^*) \setminus \{P_s\}$ checks if their value of m is β-good. If not, they broadcast a bit indicating that they reject the β-hashes.
   Each $P_i \in S_{k+1}(\Delta^*)$ computes and broadcasts a rejection vector $\tilde{R}_i$, where $\tilde{R}_i[j] = 1$ if and only if P_i received a β-bad share from P_j in round 4. Additionally, if P_i received $\geq n - t$ β-good shares, P_i uses these to reconstruct m' and checks if it is β-good. If it isn't, or if P_i's value of m_i is β-bad, P_i broadcasts a bit indicating that they reject the β-hashes.
   For every i, j for which (A2), (B2) or (C2) holds, the conflict $\{P_i, P_j\}$ is added to Δ:
   (A2) $P_i, P_j \in H_k(\Delta^*)$ and exactly one of P_i, P_j rejected the β-hashes.
   (B2) We have $\tilde{R}_j[i] = 1$ but P_i did not reject the β-hashes.
   (C2) $P_i \in S_{k+1}(\Delta^*)$ rejected the β-hashes, but $P_j \in H_k(\Delta^*)$ did not.

Output: All parties: Δ. Parties in $S_{k+1}(\Delta^*)$: m

Proposition 4. *`ITPropagate` achieves information-theoretic message propagation, i.e. if Δ is valid and all honest parties $P_i \in H_k(\Delta^*)$ agree on m, the protocol will return a valid set $\tilde{\Delta} \supseteq \Delta$ such that all honest $P_i \in H_{k+1}(\tilde{\Delta}^*)$ agree on m with high probability.*

Proof. We write Δ_r to denote the set of detected conflicts after round r of `ITPropagate`, and we begin by showing the following:

Claim. If $P_i \in H_k(\Delta_2^*)$ rejects the α-hashes in round 3, we have $P_i \notin H_{k+1}(\Delta_3^*)$. Moreover, if $P_i \in H_{k+1}(\Delta_5^*)$ rejects the β-hashes in round 6, we have $P_i \notin H_{k+1}(\Delta_6^*)$.

Proof of Claim. We only show the first statement, the second is analogous. Let $P_i \in H_k(\Delta_2^*)$ be some party that rejected the α-hashes in round 3. If $d_{\Delta_3^*}(P_s, P_i) = +\infty$, we are done. Otherwise, consider some shortest path from P_s to P_i in the graph $G = (\mathcal{P}, \binom{\mathcal{P}}{2} \setminus \Delta_3^*)$. We show that this path must contain a party $P_X \notin H_k(\Delta_2^*)$: Assume that this is not the case. Then all parties other than P_s would have had to check and potentially reject the α-hashes. Since there was no conflict registered between P_s and the next P_j on the path, P_j did not reject the hashes (see condition (A1)). Applying this argument inductively to every link in the path would show that P_i did not reject the hashes, contrary to our assumption. Therefore, the path contains some $P_X \notin H_k(\Delta_2^*)$. Since distances can only increase as conflicts are detected, we get $d_{\Delta_3^*}(P_s, P_i) = d_{\Delta_3^*}(P_s, P_X) + d_{\Delta_3^*}(P_X, P_i) \geq (k+1) + 1 \geq k+2$ which proves the claim.

Next, we'll show — assuming that the protocol starts with a valid set of conflicts Δ_0 and that we have agreement on the value of m among all honest $P_i \in H_k(\Delta_0^*)$ — that all the Δ_r are valid. For $\Delta_2 = \Delta_1 = \Delta_0$, there is nothing to show. For Δ_3, agreement on m among all honest parties in $H_k(\Delta_2^*)$ implies that either all or none of the honest parties in $H_k(\Delta_2^*)$ will reject the α-hashes. Moreover, for any value transmitted between two honest parties, it is clear that the sender and recipient will agree on wether or not said value was α-good. Therefore conditions (A1) and (B1) never lead to a conflict between two honest parties, and $\Delta_5 = \Delta_4 = \Delta_3$ are valid. For Δ_6, the argument that conditions (A2) and (B2) do not lead to conflicts between honest parties is analogous to the previous argument, using the fact that $H_k(\Delta_5^*) \subseteq H_k(\Delta_2^*)$. As for condition (C2) in round 6, there are two cases: (1) If an honest $P_i \in S_{k+1}(\Delta_5^*)$ rejects the β-hashes due to m_i being β-bad, then as all honest parties in $H_k(\Delta_5^*) \cup \{P_i\}$ agree on the value of m_i,[8] any honest $P_j \in H_k(\Delta_5^*)$ will reject as well. (2), if an honest $P_i \in S_{k+1}(\Delta_5^*)$ rejects the β-hashes due to having reconstructed some β-bad m from β-good shares, then consider some honest $P_j \in H_k(\Delta_5^*)$. Either P_j holds the same β-bad value for m that P_i reconstructed, or P_j holds a value m' s.t. at least one share of m' has a different value different from the corresponding β-good share used by P_i to reconstruct m. Since β_i was chosen uniformly at random, this different value is β-bad with high probability. Therefore, with high probability, any honest $P_j \in H_k(\Delta_5^*)$ will reject the β-hashes as well. This proves that $\Delta_1, \ldots, \Delta_6$ are all valid.

Finally, we need to show that all honest parties in $H_{k+1}(\Delta_6^*)$ agree on m. If, at any point in the protocol, there were no honest parties in $S_{k+1}(\Delta^*)$, we can

[8] This agreement is due to the fact that either $P_i \in S_{k+1}(\Delta_2^*)$, and therefore P_i picked a uniformly random α_i which — by the claim — every $P_j \in H_k(\Delta_5^*)$ accepted or $P_i \in H_k(\Delta_2^*) \cap S_{k+1}(\Delta_3^*)$, in which case it holds by assumption.

conclude that all honest parties in $H_{k+1}(\Delta_6^*)$ were also in $H_k(\Delta_0^*)$ at the start of the protocol, and we get agreement on m among them by assumption. From now on, we can therefore assume that both the α_i and the β_i have some randomly chosen keys among then. We'll show that every honest party in $H_{k+1}(\Delta_6^*)$ has a β-good value for m, proving agreement on m with high probability. By the Claim we already know that all parties in $H_{k+1}(\Delta_6^*)$ accepted the β-hashes. All that's left to show is that any $P_i \in S_{k+1}(\Delta_6^*)$ accepted enough shares to reconstruct m: Given such a P_i, let $P_j \in S_k(\Delta_6^*)$ with $\{P_i, P_j\} \notin \Delta_6^*$. By the proof of Lemma 1, the set $\mathcal{U} = \mathcal{P} \setminus ((\Delta_6^*)_{P_i} \cup (\Delta_6^*)_{P_j})$ contains at least $n - t$ parties, all of which accepted the β-hashes. Since P_i is not in conflict with any of the parties in $\mathcal{U}$ we get that P_i accepted all shares received from parties in $\mathcal{U}$ during round 6, and therefore P_i was able to reconstruct m.

□

7.2 Information Theoretic Broadcast

`ITBroadcast` is constructed from `ITPropagate` in an analogous way to the construction of `CryptoBroadcast` from `CryptoPropagate`.

Protocol 4 `ITBroadcast`
Inputs: P_s holds some message m divided into q blocks $m = m^{(1)}|\ldots|m^{(q)}$

Initialize $\Delta = \emptyset$
Repeat the following for $r = 1, \ldots$
r: All Parties run $\Delta^{(k)} \leftarrow$ `ITPropagate`$(\Delta, r-k, m^{(k)})$ in parallel for each $k \in \{1, \ldots, q\}$ with $k \leq r$ and for which there exists $P \in \mathcal{P}$ with $r - k < d_{\Delta^*}(P_s, P) < +\infty$.
If there is no such k, the protocol is terminated.
Otherwise, Δ is replaced by the union of all the $\Delta^{(k)}$ and the protocol proceeds to round $r + 1$
Output: If $d_{\Delta^*}(P_s, P_i) < +\infty$, P_i reconstructs and outputs m, otherwise P_i outputs $\perp$.

Lemma 3. *After round r of `ITBroadcast` Δ is valid and for all $k = 1, \ldots, q$, there is agreement on the value of $m^{(k)}$ among all honest parties in $H_{r-k+1}(\Delta^*)$ with high probability.*

Proof. Analogous to Lemma 2, using Prop. 4 instead of Prop. 2. □

Proposition 5. *`ITBroadcast` achieves broadcast for up to $t < n$ corruptions with information theoretic security. If $q = \frac{2n}{n-t}$, its simultaneous-channels communication-depth is at most*

$$\mathcal{O}\left(\frac{\ell}{n-t} + \frac{n^3}{(n-t)^2}\mathcal{B}(\kappa) + \frac{n^2}{(n-t)}\mathcal{B}(n)\right)$$

where $\mathcal{B}(k)$ denotes the simultaneous-channels communication-depth of the underlying broadcast protocol when used with k-bit input.

Proof. Correctness and Termination after at most $q + \frac{2n}{n-t}$ rounds can be proved in an analogous way to the cryptographic version (Prop. 3).

The simultaneous-channels communication-depth of one round is $\mathcal{O}(\frac{\ell}{q(n-t)})$ for rounds 1 and 4 of `ITPropagate`, $\mathcal{O}(qn\mathcal{B}(\kappa))$ for broadcasting the keys and hash values, and $\mathcal{O}(n\mathcal{B}(n) + qn\mathcal{B}(1))$ for broadcasting the acceptance bits. All in all, using $q = \frac{2n}{n-t}$ and dropping some lower order terms, this leads to a simultaneous-channels communication-depth

$$\mathcal{O}\left(\frac{\ell}{n-t} + \frac{n^3}{(n-t)^2}\mathcal{B}(\kappa) + \frac{n^2}{(n-t)}\mathcal{B}(n)\right)$$

□

With the broadcast protocol from [PW96] — which has a communication complexity of $\mathcal{O}(\ell n + \kappa n^5)$ — we get the following:

Theorem 4. *Let $\epsilon > 0$. For up to $t < (1-\epsilon)n$ corruptions, broadcast of an ℓ-bit message can be achieved with information theoretic security and a simultaneous-channels communication-depth of at most $\mathcal{O}(\frac{\ell}{n} + n^6\kappa)$, a simultaneous-parties communication-depth of at most $\mathcal{O}(\ell + n^7\kappa)$ and a communication complexity of at most $\mathcal{O}(n\ell + n^8\kappa)$, assuming a precomputation phase with a broadcast channel.*

Proof. Follows from Prop. 5, and the facts that $D_{\text{sp}} \leq nD_{\text{sc}}$ and that the communication complexity is upper bounded by nD_{sp}. □

8 Lower Bounds

In this section we present our lower bounds. Consider a protocol π that allows a sender P_s to broadcast an ℓ-bit message to a set of n recipients, where up to t of the recipients might be corrupted.

For completeness, we first state the well-known fact that the communication-complexity must be $\Omega(\ell n)$.

Lemma 4. *([FH06]) Any protocol for broadcasting an ℓ-bit message to a set of n recipients has an expected communication complexity of at least $\Omega(\ell n)$.*

Proof. This follows from the fact that every recipient needs to receive the ℓ-bit message by validity of the broadcast protocol.

Next, we show that the simultaneous-parties communication-depth $D_{\text{sp}}(\pi) = \Omega(\ell\lceil\frac{n}{n-t}\rceil)$.

Lemma 5. *Any broadcast protocol which tolerates up to $t \geq \frac{n}{2}$ corruptions has a simultaneous-parties communication-depth D_{sp} of at least $\ell\lfloor\frac{n}{n-t}\rfloor$.*

Proof. Let π be a protocol for broadcasting an ℓ-bit message to n recipients, where t of them may be corrupt. We will show that if π has a simultaneous-parties communication-depth $D_{\text{sp}}(\pi) < \ell\lfloor\frac{n}{n-t}\rfloor$, then there exists an adversary

for which π has constant failure probability (i.e. not negligible w.r.t. the security parameter).

Consider the following adversary strategy: The adversary partitions the n recipients into $k = \lfloor \frac{n}{n-t} \rfloor$ sets $C_1, \ldots, C_k$ of $n-t$ parties each (the last set might be bigger). The adversary chooses $j \in \{1, \ldots, k\}$ at random and corrupts all parties from sets C_i with $i \neq j$. Note that this is possible since C_j contains at least $n-t$ parties.

Then, the adversary lets each corrupted party follow the protocol instructions, except that any communication to or from parties in other partition sets is blocked. More precisely, every party $P \in C_i$ executes the prescribed protocol with the following modifications: 1) whenever a party $P \in C_i$ gets a message from $P' \notin C_i \cup P_s$, then P ignores the message and proceeds as if it did not receive any message; and 2) whenever $P \in C_i$ is instructed to send a message to $P' \notin C_i \cup P_s$, the message is dropped.

Hence, no matter which of the partition sets is uncorrupted, the overall communication pattern is always the same from the perspective of P_s: The sender can talk to all recipients, but recipients can only talk to other recipients in the same partition set (and the sender).

As P_s is unable to know which of the partition sets is the uncorrupted one, he must ensure validity by sending at least ℓ bits of information into each partition set. More precisely, if there is a set C_i that receives at most $\ell - 1$ bits, then with probability $1/k$, C_i is the set of honest parties, and with probability at least $1/2k$, they output a message different from the senders input, thus violating validity.

This implies that any secure broadcast protocol π has $D_{\text{sp}}(\pi) \geq \ell \lfloor \frac{n}{n-t} \rfloor$. □

Finally, we consider the simultaneous-channels communication-depth:

Lemma 6. *Any n-party protocol for broadcasting an ℓ-bit message to a set of n recipients has a simultaneous-channels communication-depth D_{sc} of at least $\lceil \frac{\ell}{n-t} \rceil$, where t denotes the corruption threshold.*

Proof. Assume π is a secure broadcast protocol tolerating up to t corruptions. Consider an adversary which corrupts t parties (not including P_s) and instructs them to withhold all messages. By validity, any honest $P_i \neq P_s$ must receive at least ℓ bits over the course of the protocol. Since P_i only receives messages from $n-t$ parties, there exists some party from which P_i receives at least $\lceil \frac{\ell}{n-t} \rceil$ bits. No matter how these bits are distributed over the rounds, they will incur a simultaneous-channels communication-depth $D_{\text{sc}}(\pi) \geq \lceil \frac{\ell}{n-t} \rceil$. □

9 Conclusions

We introduced a new complexity measure for synchronous protocols, called communication depth, which addresses some of the shortcomings of communication complexity. Unlike the communication complexity, communication-depth accounts for the difference between parallel and sequential communication, and it accounts for parties' need to remain synchronized even when the amount of data

which needs to be communicated in a round is not common knowledge before the round starts.

Additionally, we examined the communication depth of multivalued broadcast protocols. Previous protocols only achieved optimal communication depth in the honest majority case. We introduce a protocol which achieves optimal communication depth and optimal communication complexity in the setting where $n - t = \Theta(n)$.

References

[BCG21] Elette Boyle, Ran Cohen, and Aarushi Goel. "Breaking the O($\sqrt{n}$)-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party". *Proceedings of the 40th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 2021, pp. 319–330.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". *20th ACM STOC*. ACM Press, May 1988, pp. 1–10. DOI: 10.1145/62212.62213.

[BH06] Zuzana Beerliova-Trubiniova and Martin Hirt. "Efficient Multi-Party Computation with Dispute Control". *Theory of Cryptography Conference — TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. Springer-Verlag, Mar. 2006, pp. 305–328.

[Bha+22] Amey Bhangale et al. "Efficient Adaptively-Secure Byzantine Agreement for Long Messages". *ASIACRYPT 2022, Part I*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13791. LNCS. Springer, Cham, Dec. 2022, pp. 504–525. DOI: 10.1007/978-3-031-22963-3_17.

[Blu+23] Erica Blum et al. *Communication Lower Bounds for Cryptographic Broadcast Protocols*. 2023. arXiv: 2309.01466 [cs.CR]. URL: https://arxiv.org/abs/2309.01466.

[But+13] Vitalik Buterin et al. "Ethereum white paper". *GitHub repository* 1 (2013), pp. 22–23.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. "Multiparty Unconditionally Secure Protocols (Extended Abstract)". *20th ACM STOC*. ACM Press, May 1988, pp. 11–19. DOI: 10.1145/62212.62214.

[Cen+15] Keren Censor-Hillel et al. "Algebraic Methods in the Congested Clique". *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. PODC '15. Donostia-San Sebastián, Spain: Association for Computing Machinery, 2015, pp. 143–152. ISBN: 9781450336178. DOI: 10.1145/2767386.2767414. URL: https://doi.org/10.1145/2767386.2767414.

[Che21] Jinyuan Chen. "Optimal Error-Free Multi-Valued Byzantine Agreement". *35th International Symposium on Distributed Computing (DISC 2021)*. Ed. by Seth Gilbert. Vol. 209. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl

– Leibniz-Zentrum für Informatik, 2021, 17:1–17:19. ISBN: 978-3-95977-210-5. DOI: 10.4230/LIPIcs.DISC.2021.17. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2021.17.

[CO18] Wutichai Chongchitmate and Rafail Ostrovsky. *Information-Theoretic Broadcast with Dishonest Majority for Long Messages*. Cryptology ePrint Archive, Paper 2018/829. https://eprint.iacr.org/2018/829. 2018. URL: https://eprint.iacr.org/2018/829.

[Dam+15] Ivan Damgård et al. *Unconditionally Secure Computation with Reduced Interaction*. Cryptology ePrint Archive, Paper 2015/630. 2015. URL: https://eprint.iacr.org/2015/630.

[DKO14] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. "On the power of the congested clique model". *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*. PODC '14. Paris, France: Association for Computing Machinery, 2014, pp. 367–376. ISBN: 9781450329446. DOI: 10.1145/2611462.2611493. URL: https://doi.org/10.1145/2611462.2611493.

[DS83] Danny Dolev and H. Strong. "Authenticated Algorithms for Byzantine Agreement". *SIAM J. Comput.* 12 (Nov. 1983), pp. 656–666. DOI: 10.1137/0212045.

[FGK24] Rex Fernando, Yuval Gelles, and Ilan Komargodski. "Scalable Distributed Agreement from LWE: Byzantine Agreement, Broadcast, and Leader Election". *Proceedings of the 15th Annual Innovations in Theoretical Computer Science (ITCS) conference*. Vol. 287. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 46:1–46:23.

[FH06] Matthias Fitzi and Martin Hirt. "Optimally Efficient Multi-Valued Byzantine Agreement". *Proc. 25th ACM Symposium on Principles of Distributed Computing — PODC 2006*. Ed. by Dahlia Malkhi. ACM, July 2006.

[GK24] Yuval Gelles and Ilan Komargodski. "Optimal Load-Balanced Scalable Distributed Agreement". *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*. 2024, pp. 411–422.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 218–229. DOI: 10.1145/28395.28420.

[GP21] Chaya Ganesh and Arpita Patra. "Optimal extension protocols for Byzantine broadcast and agreement". *Distributed Computing* 34 (Feb. 2021). DOI: 10.1007/s00446-020-00384-1.

[HR14] Martin Hirt and Pavel Raykov. "Multi-Valued Byzantine Broadcast: the $t < n$ Case". *Advances in Cryptology — ASIACRYPT 2014*. Vol. 8874. Lecture Notes in Computer Science. Springer, Dec. 2014, pp. 448–465.

[KS11] Valerie King and Jared Saia. "Breaking the $O(n^2)$ bit barrier: Scalable Byzantine agreement with an adaptive adversary". *Journal of*

the ACM 58.4 (2011). A preliminary version appeared at PODC'10, 18:1–18:24.

[KS18] Janne H. Korhonen and Jukka Suomela. "Towards a Complexity Theory for the Congested Clique". *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. SPAA '18. Vienna, Austria: Association for Computing Machinery, 2018, pp. 163–172. ISBN: 9781450357999. DOI: 10.1145/3210377.3210391. URL: https://doi.org/10.1145/3210377.3210391.

[LNS25] Andrew Lewis-Pye, Kartik Nayak, and Nibesh Shrestha. *The Pipes Model for Latency and Throughput Analysis*. Cryptology ePrint Archive, Paper 2025/1116. 2025. URL: https://eprint.iacr.org/2025/1116.

[Lot+03] Zvi Lotker et al. "MST construction in O(log log n) communication rounds". *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '03. San Diego, California, USA: Association for Computing Machinery, 2003, pp. 94–100. ISBN: 1581136617. DOI: 10.1145/777412.777428. URL: https://doi.org/10.1145/777412.777428.

[LV11a] Guanfeng Liang and Nitin Vaidya. "Capacity of byzantine agreement with finite link capacity". Apr. 2011, pp. 739–747. DOI: 10.1109/INFCOM.2011.5935257.

[LV11b] Guanfeng Liang and Nitin Vaidya. "Error-free multi-valued consensus with Byzantine failures". *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. 2011, pp. 11–20.

[LV12] Guanfeng Liang and Nitin H. Vaidya. "Byzantine broadcast in point-to-point networks using local linear coding". *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*. PODC '12. Madeira, Portugal: Association for Computing Machinery, 2012, pp. 319–328. ISBN: 9781450314503. DOI: 10.1145/2332432.2332492. URL: https://doi.org/10.1145/2332432.2332492.

[Nak08] Satoshi Nakamoto. "Bitcoin whitepaper". *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* 9 (2008), p. 15.

[Nay+20] Kartik Nayak et al. "Improved extension protocols for Byzantine broadcast and agreement". English (US). *34th International Symposium on Distributed Computing, DISC 2020*. Ed. by Hagit Attiya. Leibniz International Proceedings in Informatics, LIPIcs. Publisher Copyright: © Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang; licensed under Creative Commons License CC-BY 34th International Symposium on Distributed Computing (DISC 2020).; 34th International Symposium on Distributed Computing, DISC 2020 ; Conference date: 12-10-2020 Through 16-10-2020. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, Oct. 2020. DOI: 10.4230/LIPIcs.DISC.2020.28.

[PR11] Arpita Patra and C Pandu Rangan. "Communication optimal multi-valued asynchronous byzantine agreement with optimal resilience". *International Conference on Information Theoretic Security*. Springer. 2011, pp. 206–226.

[PSL80] M. Pease, R. Shostak, and L. Lamport. "Reaching Agreement in the Presence of Faults". *J. ACM* 27.2 (Apr. 1980), pp. 228–234. ISSN: 0004-5411. DOI: 10.1145/322186.322188. URL: https://doi.org/10.1145/322186.322188.

[PW96] Birgit Pfitzmann and Michael Waidner. *Information-Theoretic Pseudosignatures and Byzantine Agreement for $t \geq n/3$*. Tech. rep. IBM Research, 1996.

[RB89] Tal Rabin and Michael Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract)". *21st ACM STOC*. ACM Press, May 1989, pp. 73–85. DOI: 10.1145/73007.73014.

[RS60] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018. eprint: https://doi.org/10.1137/0108018. URL: https://doi.org/10.1137/0108018.

[TC84] Russell Turpin and Brian A Coan. "Extending binary Byzantine agreement to multivalued Byzantine agreement". *Information Processing Letters* 18.2 (1984), pp. 73–76.

[Wan+20] Jun Wan et al. "Expected Constant Round Byzantine Broadcast Under Dishonest Majority". *TCC 2020, Part I*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12550. LNCS. Springer, Cham, Nov. 2020, pp. 381–411. DOI: 10.1007/978-3-030-64375-1_14.

[Yao82] Andrew Chi-Chih Yao. "Theory and Applications of Trapdoor Functions (Extended Abstract)". *23rd FOCS*. IEEE Computer Society Press, Nov. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.