Multi-Valued Byzantine Broadcast: the $t < n$ Case

Martin Hirt, Pavel Raykov
ETH Zurich, Switzerland
{hirt,raykovp}@inf.ethz.ch

Abstract

All known protocols implementing broadcast from synchronous point-to-point channels tolerating any $t < n$ Byzantine corruptions have communication complexity at least $\Omega(\ell n^2)$. We give cryptographically secure and information-theoretically secure protocols for $t < n$ that communicate $\mathcal{O}(\ell n)$ bits in order to broadcast sufficiently long ℓ bit messages. This matches the optimal communication complexity bound for any protocol allowing to broadcast ℓ bit messages. While broadcast protocols with the optimal communication complexity exist in cases where $t < n/3$ (by Liang and Vaidya in PODC '11) or $t < n/2$ (by Fitzi and Hirt in PODC '06), this paper is the first to present such protocols for $t < n$.

Type of submission: Regular
Best student paper award: Eligible
Contact tel. +41 44 632 72 94

1 Introduction

1.1 Byzantine Broadcast

The Byzantine broadcast problem (aka Byzantine generals) is stated as follows [PSL80]: A specific party (the sender) wants to distribute a message among n parties in such a way that all correct parties obtain the same message, even when some of the parties are malicious. The malicious misbehavior is modeled by a central adversary who corrupts up to t parties and takes full control of their actions. Corrupted parties are called *Byzantine* and the remaining parties are called *correct*. Broadcast requires that all correct parties agree on the same value v, and if the sender is correct, then v is the value proposed by the sender. Broadcast is one of the most fundamental primitives in distributed computing. It is used to implement various protocols like voting, bidding, collective contract signing, etc. Basically, this list can be continued with all protocols for secure multi-party computation as defined by Yao [Yao82, GMW87].

There exist various implementations of Byzantine broadcast from synchronous point-to-point communication channels with different security guarantees. In the model without trusted setup, perfectly-secure Byzantine broadcast is achievable when $t < n/3$ [PSL80, BGP92, CW92]. In the model with trusted setup, cryptographically or information-theoretically secure Byzantine broadcast is achievable for any $t < n$ [DS83, PW96].

Closely related to the broadcast problem is the consensus problem. In consensus each party holds a value as input, and then parties agree on a common value as output of consensus. In this paper we consider the case where any number of parties may be Byzantine. In this case the consensus problem is not well-defined, and hence we do not treat it here.

1.2 Efficiency of Byzantine Broadcast

In this paper we focus on the efficiency of broadcast protocols. In particular, we are interested in optimizing their *communication complexity*. The communication complexity of a protocol is defined by Yao [Yao79] to be the number of bits sent/received by correct parties during the protocol run.[1]

Historically, the broadcast problem was introduced for binary values [PSL80]. However, in various applications *long* values are broadcast rather than bits. Examples of such applications are general purpose multi-party computation protocols and specific tasks like voting. Such a broadcast of long values is called *multi-valued* broadcast. In this paper we study the communication complexity of multi-valued broadcast protocols.

Many known protocols for multi-valued broadcast [TC84, FH06, LV11a, Pat11] are actually constructions from a broadcast of short messages and point-to-point channels. Communication complexity of such constructions is computed in terms of the point-to-point channels and the broadcast for short messages usage. The security of the protocol is based on the security of the construction and the security of the broadcast for short messages.

Let us denote the communication complexity of a short s bit message broadcast with $\mathcal{B}(s)$. The most trivial construction is to broadcast the message bit by bit, which is perfectly secure for $t < n$ and has communication complexity $\ell\mathcal{B}(1)$. The construction by Turpin and Coan [TC84] is perfectly secure and tolerates $t < n/3$ while communicating $\mathcal{O}(\ell n^2 + n\mathcal{B}(1))$ bits. The construction by Fitzi and Hirt [FH06] is information-theoretically secure and tolerates $t < n/2$ while communicating $\mathcal{O}(\ell n + n^3\kappa + n\mathcal{B}(n+\kappa))$ bits, where κ denotes a security parameter. The construction by Liang and Vaidya [LV11a] is perfectly secure and tolerates $t < n/3$ while communicating $\mathcal{O}(\ell n + \sqrt{\ell} n^2\mathcal{B}(1) + n^4\mathcal{B}(1))$ bits (for the extensions of their approach for $t < n/2$

[1] When counting the number of bits received by correct players, we take into account only messages which were *actively* received by them, i.e., messages which should be received according to the protocol specification.

see Appendix A). The construction by Patra [Pat11] is perfectly secure and tolerates $t < n/3$ while communicating $\mathcal{O}(\ell n + n^2\mathcal{B}(1))$ bits.
In this paper we consider the case where $t < n$. In this model existing protocols [DS83, PW96] were designed to broadcast bits, but since they are based on signatures (cryptographic or information-theoretic) they can be easily adopted to broadcast long messages. The protocol by Dolev and Strong [DS83] is cryptographically secure and has communication complexity $\Omega(\ell n^2 + n^3\kappa)$. The protocol by Pfitzmann and Waidner [PW96] is information-theoretically secure and has communication complexity $\Omega(\ell n^2 + n^6\kappa)$ [Fit03].

1.3 Contributions

Consider any protocol for multi-valued broadcast. Since each correct player must learn the value proposed by the sender, the communication costs of the broadcast protocol must be at least $\Omega(\ell n)$.
In case where $t < n$ known protocols [DS83, PW96] communicate at least $\Omega(\ell n^2)$ bits. In this paper we give two generic constructions for a multi-valued broadcast which allow to achieve optimal communication complexity of $\mathcal{O}(\ell n)$ bits for $t < n$. The first construction is cryptographically secure and communicates $\mathcal{O}(\ell n + n(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$ bits. The second construction is information-theoretically secure and communicates $\mathcal{O}(\ell n + n^3(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$ bits. The following table summarizes the communication costs of multi-valued broadcast protocols:

Threshold	**Security**	**Bits Communicated**	**Literature**
$t < n/3$	perfect	$\Omega(\ell n^2)$	[BGP92]
		$\mathcal{O}(\ell n + \sqrt{\ell}n^4 + n^6)$	[LV11a] with [BGP92]
		$\mathcal{O}(\ell n + n^4)$	[Pat11] with [BGP92]
$t < n/2$	inf.-theor.	$\mathcal{O}(\ell n + n^7\kappa)$	[FH06] with [PW96]
	cryptographical	$\mathcal{O}(\ell n + n^4(n + \kappa))$	[FH06] with [DS83]
$t < n$	inf.-theor.	$\Omega(\ell n^2 + n^6\kappa)$	[PW96]
		$\mathcal{O}(\ell n + n^{10}\kappa)$	This with [PW96]
	cryptographical	$\Omega(\ell n^2 + n^3\kappa)$	[DS83]
		$\mathcal{O}(\ell n + n^5\kappa)$	This with [DS83]

2 Model and Definitions

Parties. We consider a setting consisting of n parties (players) $\mathcal{P} = \{P_1, \ldots, P_n\}$ with some designated party called the sender, which we denote with P_s for some $s \in \{1, \ldots, n\}$. We assume that the parties are connected with a synchronous authentic point-to-point network. Synchronous means that all parties share a common clock and that the message delay in the network is bounded by a constant.
For a set of parties $A \subseteq \mathcal{P}$ we define $\overline{A}$ to be $\mathcal{P} \setminus A$.

Broadcast definition. A broadcast protocol allows the sender P_s to distribute a value v_s among parties $\mathcal{P}$ such that:

- (Termination) Every correct party $P_i \in \mathcal{P}$ terminates.
- (Consistency) All correct parties in $\mathcal{P}$ decide on the same value.
- (Validity) If the sender P_s is correct, then every correct party $P_i \in \mathcal{P}$ decides on the value proposed by the sender $v_i = v_s$.

Adversary. The faultiness of parties is modeled in terms of a central adversary corrupting up to $t < n$ parties, making them deviate from the protocol in any desired manner. We distinguish two types of security in this paper: *cryptographic* and *information-theoretic*. Cryptographic security guarantees that the protocol is secure based on some computational assumptions (e.g., signatures and/or collision resistant hash functions), while information-theoretical (also called statistical) security captures the fact that even a computationally unbounded adversary cannot violate the security of the protocol with a non-negligible probability.

3 Protocols Overview

We present cryptographically and information-theoretically secure constructions for multi-valued broadcast. Both constructions are built over point-to-point channels and an oracle for broadcasting short messages. When describing protocols we often say that players broadcast messages, while meaning that they actually use the given broadcast oracle.
On the highest level both constructions broadcast the long message block by block, where each block is broadcast using a special protocol for block broadcast. This block broadcast protocol achieves optimal communication complexity only in *good* executions, while in *bad* executions more bits need to be communicated. We select the number of blocks in such a way that good executions outnumber bad ones and the total communication complexity is optimal. Whether an execution is good or bad is determined using the *Dispute Control Framework* [BH06]. Dispute control is a technique which keeps track of disputes (also called conflicts) between players and ensures that occurred disputes cannot show up again. Intuitively, an execution is good if it is dispute-free, and bad otherwise.
We employ the dispute control framework as follows. We consider a set of unordered pairs of parties Δ, where $\{P_i, P_j\} \in \Delta$ represents the fact that parties P_i and P_j accuse each other of being Byzantine. Parties start a protocol by setting Δ to be the empty set. Then during the protocol run they add new disputes to Δ when they learn about new accusations. We ensure that Δ always remains *valid*, meaning that if $\{P_i, P_j\} \in \Delta$ then at least one of the players P_i, P_j is Byzantine.

4 Cryptographically Secure Construction

First, we present a protocol `CryptoBlockBC` for broadcasting blocks. `CryptoBlockBC` makes use of an external procedure for broadcasting short values and a set of disputes Δ. Then we plug `CryptoBlockBC` in the protocol `CryptoBC`, which broadcasts an ℓ bit message block by block q times. In each invocation of `CryptoBlockBC` we will use the same global variable Δ with the disputes among the players. This means that if parties P_i and P_j conflict during some block broadcast, then they conflict in all later invocations of `CryptoBlockBC`. Then, we count the communication complexity of the resulting construction and select q which makes its optimal.

4.1 Block Broadcast Protocol `CryptoBlockBC`

The protocol `CryptoBlockBC` employs a collision-resistant hash function CRHash, i.e., no efficient algorithm can find two different inputs v, v' with $\mathsf{CRHash}(v) = \mathsf{CRHash}(v')$.[2] In the beginning of the protocol the sender broadcasts a hash $h = \mathsf{CRHash}(v_s)$ of the value it holds. The goal of

[2] This is rather informal definition of collision resistance for unkeyed hash functions, for a more formal treatment see [Rog06].

the protocol is to ensure that all correct players learn v_s. All parties during the protocol run are divided into two sets: H and $\overline{H}$. The set H consists of happy players who have already learned v_s, and $\overline{H}$ who have not. At each iteration of `CryptoBlockBC` we try to move a player from $\overline{H}$ to H. We select a pair of players P_x, P_y such that $P_x \in H$ and $P_y \in \overline{H}$. Then P_x sends the value it holds to P_y. This procedure is meaningless if parties P_x, P_y are in the dispute, so the pair is chosen such that $\{P_x, P_y\} \notin \Delta$. Once P_y receives a value from P_x it verifies that its hash is h; in the positive case P_y is included in H and in the negative case a conflict between P_x and P_y is found. Hence at each iteration we either include one player into H or we discover a new conflict between a pair of players.

Protocol `CryptoBlockBC`(v_s):

1. Parties initialize happy set H to be $\{P_s\}$.
2. Sender P_s: Broadcast $h := \mathsf{CRHash}(v_s)$.
3. While $\exists\, P_x, P_y \in \mathcal{P}$ s.t. $P_x \in H$ and $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$ do
   - $r.1$ P_x: Send v_x to player P_y. Denote received value by v_y.
   - $r.2$ P_y: If $h = \mathsf{CRHash}(v_y)$ broadcast 1, else broadcast 0.
   - $r.3$ If P_y broadcasted 1 then parties add P_y to H, otherwise they add $\{P_x, P_y\}$ to Δ.
4. $\forall P_i \in \mathcal{P}$: If $P_i \in H$ decide on v_i, otherwise decide on $\perp$.

Lemma 1. Given that the initial dispute set Δ_s is valid and CRHash is a collision-resistant hash function, protocol `CryptoBlockBC` achieves broadcast (of v_s) and terminates with a valid dispute set Δ_e. Furthermore, the protocol communicates at most $\mathcal{B}(|h|) + (n+d)(|v_s| + \mathcal{B}(1))$ bits, where $d = |\Delta_e| - |\Delta_s|$, $|h|$ is the output length of CRHash, and $|v_s|$ is the block length.

PROOF First, we prove that at each iteration of the while loop all correct players in H always hold the same value v such that $\mathsf{CRHash}(v) = h$. A player is included into H under condition that it broadcasts 1 at Step $r.2$, which he does only if it holds a value v with $\mathsf{CRHash}(v) = h$. Hence for any two correct players $P_i, P_j \in H$ it must hold that $\mathsf{CRHash}(v_i) = h$ and $\mathsf{CRHash}(v_j) = h$. Since CRHash is collision-resistant it implies that $v_i = v_j$.[3]
(Validity of Δ_e) We show that whenever P_x and P_y are correct then $\{P_x, P_y\}$ is not added to Δ at Step $r.3$. A correct $P_x \in H$ holds v_x with $\mathsf{CRHash}(v_x) = h$ and sends $v_x = v_y$ to P_y at Step $r.1$, who successfully verifies that $\mathsf{CRHash}(v_y) = h$ and broadcasts 1 at Step $r.2$, hence $\{P_x, P_y\}$ is not added to Δ at Step $r.3$.
(Termination) At each iteration of the while loop either the happy set H or the dispute set Δ grows. $|H|$ is limited by n and $|\Delta|$ is limited by n^2, hence the number of iterations is limited.
(Consistency) We prove that in the end of the protocol all correct players belong either to H (and decide on the same value v) or to $\overline{H}$ (and decide on $\perp$). As shown above Δ remains valid in all iterations, hence for correct players P_x and P_y the pair $\{P_x, P_y\} \notin \Delta$. Hence, if $P_x \in H$ and $P_y \in \overline{H}$ then the while loop does not terminate.
(Validity) The sender P_s is always in H. If P_s is correct then it decides on v_s and due to the consistency criterion all other correct players decide on v_s as well.
(Communication complexity analysis) At each iteration of the while loop either H or Δ grows. Hence, the total number of iterations of the while loop is upper bounded by $n + d$ where d is $|\Delta_e| - |\Delta_s|$. So, the total communication costs of the protocol are upper bounded by $\mathcal{B}(|h|) + (n+d)(|v_s| + \mathcal{B}(1))$. ∎

[3] More formally, when an adversary can provoke two correct players to hold colliding values for CRHash with non-negligible probability, then this adversary can be used to construct an efficient collision-finding algorithm for CRHash.

4.2 Constructing Broadcast for Long Messages

Now we plug in `CryptoBlockBC` in the protocol `CryptoBC` which broadcasts a message block by block.

> **Protocol** `CryptoBC`(v_s, q):
> 1. Parties initialize dispute set Δ with the empty set.
> 2. Sender P_s: Cut v_s in q pieces $v^1, \ldots, v^q$ (add padding if required).
> 3. For $r = 1, \ldots, q$ invoke `CryptoBlockBC`(v^r), denote the output of party P_i by v_i^r.
> 4. $\forall P_i \in \mathcal{P}$: If one of $v_i^r = \perp$ then output $\perp$, otherwise output $v_i^1 || \cdots || v_i^q$.

Since block broadcast is invoked q times, due to Lemma 1 the total communication complexity is at most

$$\sum_{i=1}^{q} \Big[\mathcal{B}(|h|) + (n + d_i)(\ell/q + \mathcal{B}(1))\Big] = q\mathcal{B}(|h|) + (qn + \sum_{i=1}^{q} d_i)(\ell/q + \mathcal{B}(1))$$

bits. We know that the sum of d_i is upper bounded by the total number of possible disputes n^2. Hence we have that communication complexity is upper bounded by $q\mathcal{B}(|h|) + (qn + n^2)(\ell/q + \mathcal{B}(1))$. By setting $q = n$ we get that the total communication is at most $2\ell n + 2n^2\mathcal{B}(1) + n\mathcal{B}(|h|)$ which is $\mathcal{O}(\ell n + n(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$. The following theorem summarizes the cryptographically secure construction presented in this section:

Theorem 1. In the setting with $t < n$, the protocol `CryptoBC` with $q = n$ achieves cryptographically secure broadcast of ℓ bit messages with communication complexity $\mathcal{O}(\ell n + n(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$ (where κ is a security parameter).

In order to obtain a concrete multi-valued broadcast protocol we instantiate `CryptoBC` with the protocol [DS83]:

Theorem 2. The protocol `CryptoBC` with $q = n$ and implementation of broadcast for short messages by [DS83] is a cryptographically secure multi-valued broadcast protocol for $t < n$ and has communication complexity $\mathcal{O}(\ell n + n^5\kappa)$ bits (where κ is a security parameter).

5 Information-Theoretically Secure Construction

This section is organized similar to the cryptographic case. First, we present a protocol `ITBlockBC` for broadcasting blocks which is analogous to `CryptoBlockBC`, with the difference that it relies on a universal hash function instead of a collision-resistant one. As in the cryptographic case we then plug `ITBlockBC` in the `ITBC` protocol, which broadcasts a message block by block q times. Then, we count the communication complexity of the resulting protocol `ITBC`, and select the number of blocks q which makes it optimal.

5.1 Universal Hash Functions

Consider a family of functions $\mathcal{U} = \{U_k\}_{k \in \mathcal{K}}$ indexed with a key set $\mathcal{K}$, where each function U_k maps elements of some set $\mathcal{X}$ to a fixed set of bins $\mathcal{Y}$. The family $\mathcal{U}$ is called ε-universal if for any two distinct messages v_1 and v_2,

$$\frac{|\{k \in \mathcal{K} \mid U_k(v_1) = U_k(v_2)\}|}{|\mathcal{K}|} \leq \varepsilon.^4$$

[4] This is a combinatorial definition of a universal hash function, usually the last condition is written probabilistically as $\Pr[k \xleftarrow{\$} \mathcal{K} : U_k(v_1) = U_k(v_2)] \leq \varepsilon$.

A ε-universal hash function can for example be constructed as follows: Let $\mathcal{X} = \{0,1\}^\ell$, $\mathcal{K} = \mathcal{Y} = \mathrm{GF}(2^\nu)$, and any value $v \in \{0,1\}^\ell$ be interpreted as a polynomial f_v over $\mathrm{GF}(2^\nu)$ of degree $\lceil \ell/\nu \rceil - 1$. The hash function is defined as $U_k(v) = f_v(k)$. We know that two distinct polynomials of degree $\lceil \ell/\nu \rceil - 1$ can match in at most $\lceil \ell/\nu \rceil - 1$ points. Hence, for any two distinct $v_1, v_2 \in \{0,1\}^\ell$,

$$\frac{|\{k \in \{0,1\}^\nu \mid U_k(v_1) = U_k(v_2)\}|}{2^\nu} \leq \frac{\lceil \ell/\nu \rceil - 1}{2^\nu} \leq 2^{-\nu}\ell.$$

So, $\{U_k\}_{k \in \{0,1\}^\nu}$ is a family of $(2^{-\nu}\ell)$-universal hash functions.
We will denote a ε-universal hash function with ITHash.

5.2 Block Broadcast Protocol `ITBlockBC`

Similarly to the cryptographic case all parties during the protocol `ITBlockBC` run are divided into two sets: H and $\overline{H}$. The set H consists of happy players who have already learned v_s, and $\overline{H}$ who have not. The difference to the cryptographic case is that the set H is not monotonically growing—it may happen that the same player may be added/removed from H several times. At each iteration of `ITBlockBC` we try to move a player from $\overline{H}$ to H. We select a pair of players P_x, P_y such that $P_x \in H$ and $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$. Then P_x sends the value it holds to P_y. Now player P_y needs to verify that the value received from P_x is the value that correct parties in H hold. In order to do so, P_y broadcasts a key k for ε-universal hash function ITHash, and then P_s broadcasts a hash h for this key. As long as P_y honestly chooses k uniformly at random, with overwhelming probability correct players will obtain different hashes if they hold different values. If a party in $H \cup \{P_y\} \setminus \{P_s\}$ holds a value with a hash h, then he broadcasts 1, and 0 otherwise (the sender P_s does not broadcast because if he is correct he can broadcast only 1). If every party broadcasts 1, then the iteration was successful and P_y is added to H. Otherwise, some of the parties in $H \cup \{P_y\}$ do not hold the right value and we search for new disputes.
An important difference from the cryptographic case is that disputes may occur not only between P_x and P_y, but between any two parties in H. In order to find such disputes, one must be able to reason about the history of how H was formed. We will keep a history set T which will contain pairs of players (P_x, P_y) such that P_y learned the value it holds from P_x.

Protocol `ITBlockBC`(v_s):

1. Parties initialize happy set H to be $\{P_s\}$ and history set T to be $\emptyset$.
2. While $\exists\, P_x, P_y \in \mathcal{P}$ s.t. $P_x \in H$ and $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$ do
   - $r.1$ P_x: Send v_x to player P_y. Denote received value by v_y. Add (P_x, P_y) to T.
   - $r.2$ P_y: Generate random $k \in \mathcal{K}$ and broadcast it.
     Sender P_s: Broadcast $h := \mathsf{ITHash}_k(v_s)$.
   - $r.3$ $\forall P_i \in H \cup \{P_y\} \setminus \{P_s\}$: If $h = \mathsf{ITHash}_k(v_i)$ then broadcast 1, otherwise 0.
   - $r.4$ If all parties broadcasted 1
     - Add P_y to H.

     else
     - For all $(P_i, P_j) \in T$ s.t. P_i broadcasted 1 (resp. $P_i = P_s$) and P_j broadcasted 0, add $\{P_i, P_j\}$ to Δ.
     - Set H to $\{P_s\}$, T to $\emptyset$.
3. $\forall P_i \in \mathcal{P}$: If $P_i \in H$ decide on v_i, otherwise decide on $\perp$.

Lemma 2. Given that the initial dispute set Δ_s is valid and ITHash is a universal hash function, protocol `ITBlockBC` achieves broadcast (of v_s) and terminates with a valid dispute set Δ_e (except with negligible probability). Furthermore, the protocol communicates at most $(n + nd)(|v_s| + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1))$ bits, where $d = |\Delta_e| - |\Delta_s|$, $|h|$ is the output length of ITHash, $|k|$ is the key length of ITHash, and $|v_s|$ is the block length.

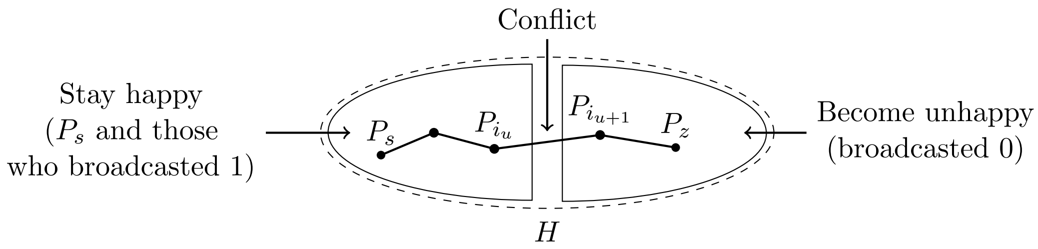

Figure 1: Conflict finding in the iteration of `ITBlockBC`.

PROOF First, we prove that at each iteration of the while loop all correct players in H always hold the same value v. More precisely, we need to show that if a correct player P_y is added to H, then, given that all correct players in H hold the same value v, it holds that $v_y = v$. We have that all parties in $H \cup \{P_y\} \setminus \{P_s\}$ broadcast 1 at Step r.3. This implies that P_y successfully verifies that $\mathsf{ITHash}_k(v_y) = h$, and all correct parties in H verify that $\mathsf{ITHash}_k(v) = h$. Due to the fact that P_y is correct, the key k is chosen uniformly at random, so given that $\mathsf{ITHash}_k(v_y) = \mathsf{ITHash}_k(v)$, it must hold with overwhelming probability $1 - \varepsilon$ that $v_y = v$.
Second, we show that if the condition at Step r.4 is false then at least one new conflict is found. We have that not all players in $H \cup \{P_y\} \setminus \{P_s\}$ broadcasted 1. Consider two possible cases:

(Exists $P_z \in H \setminus \{P_s\}$ which broadcasts 0 at step r.3) For P_z to be included in H there must exist a sequence of players $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ in H such that $P_{i_1} = P_s, P_{i_k} = P_z$ and $(P_{i_j}, P_{i_{j+1}}) \in T$ for all $j = 1, \ldots, k-1$ (see illustration in Figure 1). In the r^{th} iteration some of the players in H stayed happy (P_s and those who broadcasted 1) and some become unhappy (broadcasted 0). We know that P_s stayed happy and P_z became unhappy. Hence in a row $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ there are players of both types. Then we have that exist two players $P_{i_u}, P_{i_{u+1}}$ such that P_{i_u} stays happy and $P_{i_{u+1}}$ becomes unhappy. By construction of T, $(P_{i_u}, P_{i_{u+1}}) \in T$ implies that $\{P_{i_u}, P_{i_{u+1}}\}$ is not yet in Δ. Consequently, the pair $\{P_{i_u}, P_{i_{u+1}}\}$ will be identified as having a conflict and will be added to Δ.

(Each $P_i \in H \setminus \{P_s\}$ broadcasts 1 at step r.3) It means that P_x broadcasts 1 (or $P_x = P_s$) and P_y broadcasts 0. Hence the new dispute $\{P_x, P_y\}$ will be added to Δ.

Now we proceed with the proof of the current lemma.
(Validity of Δ_e) We show that whenever P_i and P_j are correct then $\{P_i, P_j\}$ is never added to Δ. The pair $\{P_i, P_j\}$ is added to Δ only when P_i sent some v to P_j (i.e., $(P_i, P_j) \in T$), and they disagree for some key k whether $\mathsf{ITHash}_k(v)$ equals h. Hence, P_i or P_j is corrupted.
(Termination) There can be at most n successive iterations where the set H grows (condition at Step r.4 is true). As shown above whenever condition at Step r.4 is false a new conflict is found. The number of conflicts is limited and so must be the number of the while loop iterations.
(Consistency) We prove that in the end of the protocol all correct players belong either to H (and decide on the same value v) or to $\overline{H}$ (and decide on $\perp$). As shown above Δ remains valid in all iterations, hence for any two correct players P_x, P_y, the pair $\{P_x, P_y\} \notin \Delta$. Hence, if $P_x \in H$ and $P_y \in \overline{H}$ then the while loop does not terminate.
(Validity) The correct sender P_s is always in H. The sender P_s decides on v_s and due to the consistency criterion all other correct players decide on v_s as well.
(Communication complexity analysis) There can be at most n consecutive iterations, where no conflict is found, hence the total number of iterations is at most $n+nd$, where $d = |\Delta_e| - |\Delta_s|$. The communication costs of each iteration are at most $|v_s| + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1)$. So, the total communication costs of the protocol are upper bounded by $(n+nd)(|v_s| + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1))$.■

5.3 Constructing Broadcast for Long Messages

Similarly to the cryptographic case, we plug `ITBlockBC` in the protocol `ITBC` which simply broadcasts a message block by block. The protocol `ITBC` is a copy of the protocol `CryptoBC` with the only difference that `CryptoBlockBC` is substituted with `ITBlockBC`.

Protocol `ITBC`(v_s, q):

1. Parties initialize dispute set Δ to be an empty set.
2. Sender P_s: Cut v_s in q pieces $v^1, \dots, v^q$ (add padding if required).
3. For $r = 1, \dots, q$ invoke `ITBlockBC`(v^r), denote the output of party P_i by v_i^r.
4. $\forall P_i \in \mathcal{P}$: If one of $v_i^r = \perp$ then output $\perp$, otherwise output $v_i^1 || \cdots || v_i^q$.

Due to Lemma 2 the total communication complexity is at most

$$\sum_{i=1}^{q} \left[(n + d_i n)(\ell/q + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1)) \right] = n(q + \sum_{i=1}^{q} d_i)(\ell/q + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1)).$$

This expression is bound by $n(q + n^2)(\ell/q + \mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1))$. By setting $q = n^2$ we have that communication costs are at most $2\ell n + 2n^3(\mathcal{B}(|h|) + \mathcal{B}(|k|) + n\mathcal{B}(1)))$ which is $\mathcal{O}(\ell n + n^3(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$. The following theorem summarizes the information-theoretically secure construction presented in this section:

Theorem 3. In the setting with $t < n$, the protocol `ITBC` with $q = n^2$ achieves information-theoretically secure broadcast of ℓ bit messages with communication complexity $\mathcal{O}(\ell n + n^3(\mathcal{B}(\kappa) + n\mathcal{B}(1)))$ (where κ is a security parameter).

In order to obtain a concrete multi-valued broadcast protocol we instantiate `ITBC` with the protocol [PW96]:

Theorem 4. The protocol `ITBC` with $q = n^2$ and implementation of broadcast for short messages by [PW96] is an information-theoretically secure multi-valued broadcast protocol for $t < n$ and has communication complexity $\mathcal{O}(\ell n + n^{10}\kappa)$ bits (where κ is a security parameter).

6 Conclusions

Existing multi-valued broadcast protocols achieve optimal communication complexity only for $t < n/3$ [LV11a] or $t < n/2$ [FH06]. In this paper we proposed the first broadcast protocols that tolerate any $t < n$ Byzantine corruptions and achieve optimal communication complexity $\mathcal{O}(\ell n)$ for sufficiently long messages of ℓ bits. One of the proposed protocols is cryptographically secure and the other one is information-theoretically secure. The cryptographically secure protocol is based on the security of the signature scheme and a collision-resistance of the hash function employed. It communicates $\mathcal{O}(\ell n + n^5\kappa)$ bits. The information-theoretically secure protocol may fail with a negligible probability and needs to communicate $\mathcal{O}(\ell n + n^{10}\kappa)$ bits.
The presented constructions `CryptoBC` and `ITBC` leave room for different optimizations. In particular, it is an interesting task to optimize the number of rounds used. Our constructions require $\mathcal{O}(n^2)$ rounds (cryptographic one), respectively $\mathcal{O}(n^3)$ rounds (information-theoretic one). It seems that the obvious approach with simultaneously broadcasting many blocks does not improve the round complexity in the worst case. We leave the optimization of the round complexity as an open question for future research.

References

[BGP92] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Bit optimal distributed consensus. In *Computer Science Research*, pages 313–322. Plenum Publishing Corporation, New York, NY, USA, 1992. Preliminary version appeared in STOC '89.

[BH06] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference — TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer-Verlag, March 2006.

[CW92] Brian A. Coan and Jennifer L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97:61–85, March 1992. Preliminary version appeared in PODC '89.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983. Preliminary version appeared in STOC '82.

[FH06] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued Byzantine agreement. In *Proceedings of the 26th annual ACM symposium on Principles of distributed computing*, PODC '06, pages 163–168, New York, NY, USA, 2006. ACM.

[Fit03] Matthias Fitzi. *Generalized Communication and Security Models in Byzantine Agreement*. PhD thesis, ETH Zurich, March 2003. Reprint as vol. 4 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-853-3, Hartung-Gorre Verlag, Konstanz, 2003.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[LV10a] Guanfeng Liang and Nitin Vaidya. Complexity of multi-value byzantine agreement. Technical report, University of Illinois at Urbana-Champaign, "`http://www.crhc.illinois.edu/wireless/papers/ba_sum_capacity_0729.pdf`", 2010.

[LV10b] Guanfeng Liang and Nitin H. Vaidya. Short note on complexity of multi-value byzantine agreement. *CoRR*, abs/1007.4857, 2010.

[LV11a] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with Byzantine failures. In *Proceedings of the 30th annual ACM symposium on Principles of distributed computing*, PODC '11, pages 11–20, New York, NY, USA, 2011. ACM.

[LV11b] Guanfeng Liang and Nitin H. Vaidya. Error-free multi-valued consensus with byzantine failures. *CoRR*, abs/1101.3520, 2011.

[Pat11] Arpita Patra. Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity. In *Proceedings of the 15th international conference on Principles of Distributed Systems*, OPODIS '11, pages 34–49. Springer, 2011.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[PW96] Birgit Pfitzmann and Michael Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. Technical report, IBM Research, 1996.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.

[TC84] Russell Turpin and Brian A. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[Yao79] Andrew C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC '79, pages 209–213, New York, NY, USA, 1979. ACM.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

A On the Constructions of Liang and Vaidya

Liang and Vaidya have four papers based on the similar techniques which relate to the construction of multi-valued broadcast and consensus protocols:

1. The first paper [LV10a] presents a perfectly secure multi-valued broadcast protocol which tolerates $t < n/3$.
2. The second paper [LV10b] proposes an information-theoretically secure modification of the broadcast protocol from the first paper which tolerates $t < n/3$.
3. The third paper [LV11a] constructs a perfectly secure multi-valued consensus protocol for $t < n/3$.
4. The fourth paper [LV11b] is an archive version of the third paper containing the same protocol.

In the third and the fourth paper the authors explain how their protocols can be modified to tolerate $t \geq n/3$ by substituting the employed procedure for 1-bit broadcast with a protocol that tolerates $t \geq n/3$ (e.g., [DS83, PW96]). This modification can be applied in all four papers since they share similar structure and are based on similar techniques. In the following we clarify why even with this modification the protocols from the cited above papers cannot tolerate $t \geq n/2$.
In the first and the second paper the presented broadcast protocols describe communication between players based on their trust to each other. While the trust concept applies for any $t < n$, the broadcast protocols put a limitation on the trust relation—it is required that any two players either trust each other or there is another player whom both players trust (see Section V.B in [LV10a] and Section 3 in [LV10b]). Clearly, such a mutually trusted player exists only for $t < n/2$ and hence the protocols' behavior is not well-defined for $t \geq n/2$.
In the third and the fourth paper a consensus protocol for $t < n/3$ is presented. As consensus is not achievable for $t \geq n/2$, the proposed modification can at most construct consensus for $t < n/2$, which in turn provides broadcast for the same bound only.