Optimally Efficient Multi-Valued Byzantine Agreement

Matthias Fitzi*
University of Aarhus, Denmark
fitzi@daimi.au.dk

Martin Hirt
ETH Zurich, Switzerland
hirt@inf.ethz.ch

ABSTRACT

All known protocols for Byzantine agreement (BA) among n players require the message to be communicated at least $\Omega(n^2)$ times, which results in an overall communication complexity of at least $\Omega(\ell n^2)$ bits for an ℓ-bit message. We present the first BA protocol in which the message is communicated only $\mathcal{O}(n)$ times (the hidden factor is less than 2). More concretely, for a given synchronous broadcast protocol which communicates $\mathcal{B}(b)$ bits for reaching agreement on a b-bit message with security parameter κ, our construction yields a synchronous BA protocol with communication complexity $\mathcal{O}(\ell n + n\mathcal{B}(n + \kappa))$ bits. Our reduction is information theoretically secure and tolerates up to $t < n/2$ corrupted players, which is optimal for the consensus variant of BA. Although this resilience is not optimal for the broadcast (Byzantine generals) variant, it is sufficient for most distributed applications that involve BA protocols since they typically require $t < n/2$.

ACM Classification: C.2.4 [Computer-Communication Networks]: Distributed Systems; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems;
General Terms: Algorithms, Performance, Reliability, Security, Theory.
Key words: Byzantine agreement, communication complexity, cryptographic security, information-theoretic security.

1. INTRODUCTION

1.1 Byzantine Agreement, Consensus, and Broadcast

The problem of Byzantine agreement (BA), as originally proposed by Pease, Shostak, and Lamport [PSL80], is the following: n players $P_1, \dots, P_n$ want to reach agreement on some message m, but up to t of the players are malicious and try to prevent the others from reaching agreement. The misbehavior of players is modeled with a central adversary who *corrupts* up to t players and takes full control over them. There are two flavors of the BA problem: In the broadcast (aka Byzantine generals) problem, a designated player (the sender) holds an input message m, and all players should learn m and agree on it. In the consensus problem, every player P_i holds a (supposedly the same) message m_i, and the players want to agree on this message. More formally, an n-player protocol is a *broadcast protocol*, when the following properties are satisfied:

- every honest P_i eventually outputs a message m_i (*termination*),
- the outputs of all honest players are equal, i.e., $m_i = m'$ for some m' (*consistency*),
- if the sender is honest then $m' = m$ (*validity*).

Analogously, an n-player protocol is a *consensus protocol*, when the following properties are satisfied:

- every honest P_i eventually outputs a message m_i (*termination*),
- the outputs of all honest players are equal, i.e., $m_i = m'$ for some m' (*consistency*),
- if every honest P_i holds the same message $m_i = m$ for some m, then $m' = m$ (*validity*).

We say that a specific protocol tolerates up to t corruptions when the above requirements are satisfied even in presence of an adversary that corrupts up to t of the players. Note that in principal, broadcast can be achieved for any threshold $t < n$, whereas consensus can be achieved only for $t < n/2$. However, within this limitation, broadcast and consensus protocols are mutually reducible: Given a consensus protocol, a broadcast protocol can be constructed as follows: First, the sender sends the message m to every player. Then, the players use the consensus protocol to reach agreement on m. On the other hand, given a broadcast protocol, a consensus protocol can be constructed as follows: First, every player broadcasts his message m_i among all players. Then every player defines as output the message that he has received most often. Note that this reduction requires n invocations to the broadcast protocol in order to realize one single consensus.

Broadcast and consensus are probably the most important primitives in distributed cryptography. They are used in almost any task that involves multiple players, like, e.g., voting, bidding, secure function evaluation, threshold key

*Supported by the European project SECOQC.

PODC'06, July 22–26, 2006, Denver, Colorado, USA.

generation etc — just to mention a few. The restriction of consensus to at most $t < n/2$ corrupted players is usually no drawback since the protocol that invokes the consensus subprotocol often cannot deal with more corruptions anyway.

1.2 Models and Bounds

We assume that the players are connected with a complete synchronous network of pairwise authenticated channels. Complete means that each pair of players shares a channel. Synchronous means that all players share a common clock and that the message delay in the network is bounded by a constant. We consider both classical models distinguished by whether or not a public-key infrastructure (PKI) is set up among the players.

- In the model where no PKI is set up among the players, information-theoretically secure consensus (as well as broadcast) is achievable if at most $t < n/3$ players are corrupted. This bound cannot be improved even when only cryptographic security is required and/or the pairwise channels are secret.
- In the model where an information-theoretic PKI (i.e., a PKI with respect to an information-theoretically secure pseudo-signature scheme [CR90, PW96]) is set up among the players, information-theoretically secure consensus is achievable if $t < n/2$ players are corrupted. Broadcast is even achievable secure against any number of corrupted players.
- In the model where a cryptographic PKI (i.e., a PKI with respect to a cryptographically-secure signature scheme) is set up among the players, cryptographically secure consensus is achievable if at most $t < n/2$ players are corrupted. Broadcast is even achievable for any number of corrupted players.

Note that, in the latter two models, the pairwise channels need not be authenticated since authentication can be achieved by exploiting the PKI.

1.3 Efficiency of Byzantine Agreement

We are interested in the communication complexity of BA protocols. The *bit complexity* of a protocol is defined as the overall number of bits transmitted by all honest players during the whole protocol or, alternatively, the overall number of bits received and *processed* by all honest players. The messages transmitted by corrupted players are only taken into account as far as they are actually read by an honest player as, inherently, the adversary can make the corrupted players send arbitrarily long messages.

In the model where no PKI is set up, both broadcast and consensus among n players with t corruptions are achievable for $t < n/3$ with communicating $\mathcal{O}(\ell n^2)$ bits, where ℓ denotes the length of the message [BGP92, CW92]. These protocols are optimally-efficient for $\ell = \mathcal{O}(1)$ [DR85]. In the model with an information-theoretic PKI, consensus on ℓ bits is possible for $t < n/2$ with $\mathcal{O}(\ell n^3 + n^7\kappa)$ bits of communication, and broadcasting ℓ bits is possible for $t < n$ with $\mathcal{O}(\ell n^2 + n^6\kappa)$ bits of communication [PW96], where κ denotes the security parameter (i.e., the error probability $\varepsilon < 2^{-\kappa}\ell$). In the model with a cryptographic PKI, consensus is possible for $t < n/2$ with $\Omega(\ell n^3 + n^4\kappa)$ bits of communication, and broadcast is possible for $t < n$ with $\mathcal{O}(\ell n^2 + n^3\kappa)$ bits of communication [DS83] — here, κ denotes the length of a signature, which is typically larger than a security parameter in an information-theoretic protocol. All models have in common that the most efficient known BA protocols require a communication of at least $\Omega(\ell n^2)$ bits for an ℓ bit message. For large ℓ, this term dominates the overall communication complexity.

1.4 Multi-Valued Byzantine Agreement

Broadcast and consensus are the probably most vital subprotocols in many distributed applications. However, in real-life applications typically agreement must be reached on *long messages* rather than on single bits. For example, in a voting protocol, the authorities must agree on the set of all ballots to be tallied (which can be gigabytes of data). Multi-valued Byzantine agreement is also relevant in secure multi-party computation, where many broadcast invocations can be parallelized and thereby optimized to a single invocation with a long message.

The first generic reduction from long-message broadcast to short-message broadcast was given by Turpin and Coan [TC84]. The idea of their reduction is as follows: First, the sender sends his input message m to every player. Second, every player echoes the received message to every other player. Third, every player broadcasts one bit, depending on whether or not he has received at least $n - t$ identical messages in the second step. If at least $n - t$ players have seen at least $n-t$ identical messages, then all players output the message they have received most often. Otherwise, all players output some default value, denoted as $\perp$.[1] One can easily verify that this protocol is correct for $t < n/3$ (but not for $t < n/2$). The communication complexity of the protocol is $\mathcal{O}(\ell n^2 + n\mathcal{B}(1))$ bits for ℓ-bit broadcast.

An obvious approach towards a more efficient reduction (where the message is communicated only $\mathcal{O}(n)$ times) is as follows: The sender sends the message m bilaterally to every player, and broadcasts the hash value of m (using some short-message broadcast protocol). Then, every player broadcasts one bit, indicating whether or not the message he received fits the broadcasted hash value. If at least $n - t$ players accept the sender's message, then m is accepted as the broadcasted message; otherwise, the broadcast failed and all players set $m =\perp$. This approach suffers from two limitations: First, it requires a cryptographic hash function, as broadcasting for example a universal hash value does not work (the sender could derive two different messages m and m' fitting the chosen universal hash function). Second, and more importantly, even when the broadcast was accepted, there are up to t honest players who did not receive m, and the accepting players must deliver m to the rejecting players. In the case of $t < n/2$, there might be only a single honest accepting player, so *every* accepting honest player must send m to *every* rejecting player, which sums up to $\Omega(\ell n^2)$ bits of communication.

1.5 Our Results

We present a reduction for BA (broadcast or consensus) with long messages to BA (broadcast or consensus) with

[1]Typically, we assume $\perp$ to be outside of the domain of BA in order to make this value distinguishable from a regular outcome. In order to rigorously comply with the definition of BA, this symbol can be mapped to a default value inside of the respective domain.

short messages. Given a BA protocol which communicates $\mathcal{B}(b)$ bits for reaching agreement on a b-bit message, we construct a protocol that reaches agreement on an ℓ-bit message with complexity $\mathcal{O}(\ell n + n\mathcal{B}(n+\kappa))$, where κ denotes a security parameter. This is the first BA protocol in which the message is communicated only $\mathcal{O}(n)$ times, in contrast to previous protocols which communicate the whole message at least $\Omega(n^2)$ times. Communicating the message $\mathcal{O}(n)$ times is optimal both for broadcast and for consensus; no protocol can achieve broadcast or consensus of an ℓ-bit message with complexity $o(\ell n)$.

The round complexity of the resulting protocol is (up to a constant factor) the same as the round complexity of the underlying BA protocol. In particular, constant-round BA protocols for short messages are transformed into very efficient constant-round BA protocols for long messages.[2]

The reduction is information-theoretically secure with an arbitrarily small error probability (negligible in the security parameter). The reduction is robust against an active adversary corrupting up to $t < n/2$ of the players, which is optimal for consensus, but not optimal for broadcast when an appropriate key setup is established. We give a security analysis only for the case of a static adversary that corrupts players at the beginning of the protocol; however, the protocol is also secure with respect to an adaptive adversary that can corrupt players during the protocol execution, given that the employed broadcast protocol for short messages is adaptively secure.

Applying our reduction to the most efficient BA protocols in the literature, this yields the following complexities:

- In the model without key setup, information-theoretically secure broadcast and consensus are possible for $t < n/3$ by communicating $\Omega(\ell n^2)$ bits [BGP92, CW92]. Our construction reduces the required communication to $\mathcal{O}(\ell n + n^3(n+\kappa))$.

- In the model with an information-theoretic key setup, information-theoretically secure broadcast is possible for $t < n$ by communicating $\Omega(\ell n^2 + n^6\kappa)$ bits, and information-theoretically secure consensus for $t < n/2$ requires $\Omega(\ell n^3 + n^7\kappa)$ bits [PW96]. Our construction reduces the required communication to $\mathcal{O}(\ell n + n^7\kappa)$.

- In the model with a cryptographic public-key setup, cryptographically secure broadcast is possible for $t < n$ by communicating $\Omega(\ell n^2 + n^3\kappa)$ bits, and consensus is possible for $t < n/2$ by communicating $\Omega(\ell n^3 + n^4\kappa)$ bits [DS83]; Our construction reduces the required communication to $\mathcal{O}(\ell n + n^4(n+\kappa))$.

2. THE PROTOCOL IN A NUTSHELL

In this section we present the main ideas and the structure of our protocol. Technically, we construct an efficient *consensus* protocol for long messages based on access to a given (slow) *broadcast* primitive for short messages. Note that, with respect to the complexity of the final protocol, this special case optimally represents all four possible cases where the long-message protocol as well as the short-message protocol are either broadcast or consensus. This is since broadcast can be achieved with one single invocation of consensus.

The construction assumes authenticated channels among the players and black-box access to the given broadcast primitive; yielding a consensus protocol for $t < n/2$ that is as secure as the given broadcast protocol — up to a negligible error probability.[3]

The protocol proceeds in three stages, where each stage brings the players "closer" to agreement. The protocol may be aborted in any stage when (provably) inconsistencies among honest players' inputs are detected. In this case, every player picks a default output message, denoted as $\perp$.

In the sequel, we describe the three stages. For the sake of conciseness, we write $\mathcal{P}$ for the set of players and $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}$ for the subset of honest players. Note that $\mathcal{P}_{\text{honest}}$ is typically not known and cannot be computed; we just use it for the analysis of the protocol. Furthermore, we denote the inputs to the consensus protocol by m_i and the outputs by m_i'.

Checking Stage

In the *Checking Stage*, the players in $\mathcal{P}$ compare their respective messages and jointly determine an "accepting subset" $\mathcal{P}_{\text{acc}} \subseteq \mathcal{P}$ of size at least $n-t$, such that all accepting players hold the same message, and all (honest) players holding this message are accepting. This stage can be aborted when inconsistencies among honest players are detected. More formally, the checking stage must satisfy the following properties:

(1) If all honest players $P_i \in \mathcal{P}_{\text{honest}}$ hold the same input message $m_i = m$, then the checking stage succeeds, i.e., does not abort (*completeness*).

(2) All accepting honest players $P_i \in \mathcal{P}_{\text{acc}} \cap \mathcal{P}_{\text{honest}}$ hold the same input message $m_i = m$ for some m (*consistency*).

(3) If all honest players $P_i \in \mathcal{P}_{\text{honest}}$ hold the same input message $m_i = m$, then they all accept, i.e., $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{acc}}$ (*unambiguity*).

Consolidation Stage

In the *Consolidation Stage*, the accepting players help the other players to receive the right message. This will result in a (commonly known) "happy subset" $\mathcal{P}_{\text{ok}} \subseteq \mathcal{P}$, such that all happy players hold the same message, and the majority of happy players is honest. Also this stage may be aborted in case of inconsistencies among the honest players' inputs.[4] More formally, the consolidation stage must satisfy the following properties:

(4) If all honest players accept the checking stage, i.e., $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{acc}}$, then the consolidation stage succeeds (*completeness*).

[2] Note that the round complexity of *expected* constant-round BA protocols is not preserved automatically, because our construction invokes the underlying BA protocol $\mathcal{O}(n)$ times in parallel. However, known BA protocols with expected constant rounds [FM89] can easily be modified such that arbitrarily many *parallel* invocations require only expected constant rounds (by using the same common coins for all parallel instances), which suffices to make our construction expected constant round as well.

[3] Implying that, given broadcast with resilience $t < n/3$, the resulting protocol will achieve only this resilience as well.

[4] We stress that we do not require that $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{ok}}$, even if all honest players hold the same input message m_i. The *only* requirements are that all players in $\mathcal{P}_{\text{ok}}$ (claim to) hold the same (valid output) message m, and that a strict majority of players in $\mathcal{P}_{\text{ok}}$ is honest.

(5) All happy players $P_i \in \mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{honest}}$ hold the same output message $m'_i = m'$ for some m' (*consistency*).

(6) The output message m' is equal to the input message m_i of an accepting honest player $P_i \in \mathcal{P}_{\text{acc}} \cap \mathcal{P}_{\text{honest}}$ (*validity*).

(7) The majority of happy players is honest i.e., $|\mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{honest}}| > |\mathcal{P}_{\text{ok}}|/2$ (*unambiguity*).

Claiming Stage

In the *Claiming Stage*, the happy players distribute the message to the unhappy players. In order to keep the communication low, every happy player is distributing only part of the message rather than the full-length message. This stage will never be aborted. More formally, the claiming stage must satisfy the following properties:

(8) All players $P_i \in \mathcal{P}_{\text{honest}} \setminus \mathcal{P}_{\text{ok}}$ pick the same output message $m'_i = m'$ (*consistency*).

(9) The output message satisfies $m' = m'_i$ for some honest happy player $P_i \in \mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{honest}}$ (*validity*).

3. CHECKING STAGE

3.1 The Protocol

The goal of the checking stage is to identify a set $\mathcal{P}_{\text{acc}} \subseteq \mathcal{P}$ of size at least $n-t$, such that all honest players in $\mathcal{P}_{\text{acc}}$ hold the same input message m. Furthermore, $\mathcal{P}_{\text{acc}}$ must contain all honest players if they all hold the same message.

The checking protocol is based on multilateral equality checks. These equality checks will be realized with a family of ε-almost two-universal hash functions [CW79]. This is a family $\mathcal{U} = \{U_k : k = 0, \ldots, 2^\kappa - 1\}$, where each hash function U_k maps arbitrary strings $\{0,1\}^*$ to κ-bit strings. The family is almost-universal when for any distinct messages m_1 and m_2, the probability that the hash values $U_k(m_1)$ and $U_k(m_2)$ are equal for a random key $k \in \{0, \ldots, 2^\kappa - 1\}$, is bounded by $2^{-\kappa}\ell$, where ℓ denotes the bit-length of the longer message.

A universal hash function can for example be constructed as follows: The message m is interpreted as a polynomial f_m over GF(2^κ) with degree $\lceil \ell/\kappa \rceil$.[5] The hash function is defined as $U_k(m) = f_m(k)$. The probability that two given messages collide on a randomly chosen key is $2^{-\kappa} \cdot \lceil \ell/\kappa \rceil$.

The protocol for the checking stage is given in the box "Protocol Checking".

Protocol Checking

1. Every player $P_i \in \mathcal{P}$ selects at random a key k for a universal hash function U_k, computes $h_i = (k, U_k(m_i))$, and broadcasts h_i using the given broadcast protocol.
2. Every player $P_j \in \mathcal{P}$ checks the received hash values against his own message m_j and prepares a vector $\vec{v}_j \in \{\mathsf{accept}, \mathsf{reject}\}^n$, where the i-th entry in the vector denotes whether or not P_i's hash value is consistent with P_j's message. Note that the j-th position of $\vec{v}_j$ is always accept. Finally, P_j broadcasts the vector $\vec{v}_j$ among all players in $\mathcal{P}$.
3. If at least $n - t$ of the broadcasted vectors are equal (and $\vec{v}_i$ has accept in the i-th position), we denote this vector by $\vec{v}$ and denote the set of players broadcasting $\vec{v}$ by $\mathcal{P}_{\text{acc}}$. Otherwise, we abort the checking stage.

3.2 Security Analysis

We prove that the protocol satisfies the requirements (1)–(3) of Section 2:

(1) Completeness: Assume that all honest players $P_i \in \mathcal{P}_{\text{honest}}$ hold input $m_i = m$ for some m. Then every broadcasted hash value h_i will either match all m_i or no m_i.[6] Hence, every $P_i \in \mathcal{P}_{\text{honest}}$ will broadcast the same vector $\vec{v}_i$, and there are at least $|\mathcal{P}_{\text{honest}}| \geq n-t$ such vectors, and the stage succeeds.

(2) Consistency: If two honest players $P_i, P_j \in \mathcal{P}_{\text{honest}}$ hold different input messages $m_i \neq m_j$, then with high probability, P_i will not accept the hash value broadcasted by P_j, and vice versa. However, both P_i and P_j will accept their own hash value. Hence, the vectors $\vec{v}_i$ and $\vec{v}_j$ will be different, and either P_i or P_j (or both) will not accept the checking stage, i.e., $\{P_i, P_j\} \not\subseteq \mathcal{P}_{\text{acc}}$.

(3) Unambiguity: According to the proof of (1), we know that if all honest players $P_i \in \mathcal{P}_{\text{honest}}$ hold the same input message m_i, then they will broadcast the same vector $\vec{v}_i$, and all honest players will accept, i.e., $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{acc}}$.

3.3 Communication Complexity

The checking protocol communicates $2n\mathcal{B}(\kappa) + n\mathcal{B}(n)$ bits, where $\mathcal{B}(b)$ denotes the communication complexity for broadcasting a b-bit message with the short-message broadcast protocol.

4. CONSOLIDATION STAGE

4.1 The Protocol

In the consolidation stage, the (at least) $n - t$ accepting players in $\mathcal{P}_{\text{acc}}$ (who all hold the same message m) help the other players to learn the message. However, in order to keep the message complexity small, we will allow every accepting player to help only *one* non-accepting player. Therefore, we assign an accepting player to every non-accepting player. This is formalized with an injective mapping function $\Phi : (\mathcal{P} \setminus \mathcal{P}_{\text{acc}} \rightarrow \mathcal{P}_{\text{acc}})$. For example, the player with the smallest index in $\mathcal{P} \setminus \mathcal{P}_{\text{acc}}$ is associated with the player with the smallest index in $\mathcal{P}_{\text{acc}}$, etc. The protocol for the consolidation stage is given in the box "Protocol Consolidation".

4.2 Security Analysis

We prove that the protocol satisfies the requirements (4)–(7) of Section 2. We again stress that we do not require $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{ok}}$ (it might be that $\mathcal{P}_{\text{ok}}$ contains just a single player, who then is honest and knows the valid message m).

(4) Completeness: Recall that all honest accepting players $P_i \in \mathcal{P}_{\text{acc}} \cap \mathcal{P}_{\text{honest}}$ hold the same message $m_i = m$.

[5] In order to guarantee that the polynomials f_m and $f_{m'}$ for two different messages $m \neq m'$ differ, we append a 1-bit to the message.

[6] Note that all honest players receive the same hash values h_i, as a broadcast protocol (for short messages) is used for distributing h_i.

Protocol Consolidation

1. For every non-accepting player $P_j \in (\mathcal{P} \setminus \mathcal{P}_{\text{acc}})$, the accepting player $P_i = \Phi(P_j)$ sends $m = m_i$ to P_j, who denotes the received message with $m_j^?$.
2. Every player $P_j \in (\mathcal{P} \setminus \mathcal{P}_{\text{acc}})$ (uniformly) selects a random key k defining the hash function U_k, and computes and broadcasts $h_j = (k, U_k(m_j^?))$.
3. Every player $P_i \in \mathcal{P}_{\text{acc}}$ checks every h_j against the message $m = m_i$, and broadcasts a vector $\vec{v}_i \in \{\mathsf{accept}, \mathsf{reject}\}^{|\mathcal{P} \setminus \mathcal{P}_{\text{acc}}|}$, where the j-th entry indicates whether or not the j-th hash value matches m.
4. If at least $n-t$ of the broadcasted vectors are equal, we denote this vector by $\vec{v}$. Let $\mathcal{P}_{\text{rej}} \subseteq (\mathcal{P} \setminus \mathcal{P}_{\text{acc}})$ denote those players whose corresponding entry in $\vec{v}$ is reject. We set $\mathcal{P}_{\text{ok}} = (\mathcal{P} \setminus \mathcal{P}_{\text{rej}} \setminus \{\Phi(P_j) : P_j \in \mathcal{P}_{\text{rej}}\})$. Every player $P_i \in \mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{acc}}$ sets his output message $m_i' = m_i$, and every player $P_i \in (\mathcal{P}_{\text{ok}} \setminus \mathcal{P}_{\text{acc}})$ sets $m_i' = m_i^?$. If no $n-t$ of the broadcasted vectors are equal, the consolidation stage is aborted.

Hence, whatever hash values h_i are broadcasted in Step 2 of the consolidation protocol, all honest players in $\mathcal{P}_{\text{acc}}$ will broadcast the same vectors $\vec{v}_j$. If $\mathcal{P}_{\text{honest}} \subseteq \mathcal{P}_{\text{acc}}$, there are at least $|\mathcal{P}_{\text{honest}}| \geq n-t$ identical vectors $\vec{v}_j$, and the consolidation stage succeeds.

(5) Consistency: Let m denote the message of the honest players in $\mathcal{P}_{\text{acc}}$. Consider an arbitrary honest happy player $P_j \in \mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{honest}}$. If $P_j \in \mathcal{P}_{\text{acc}}$, then $m_j' = m_j = m$. If $P_j \notin \mathcal{P}_{\text{acc}}$, then P_j has received m_j from $\Phi(P_j)$ in Step 2 of the Consolidation protocol. As $P_j \in \mathcal{P}_{\text{ok}}$, the hash value of m_j has been confirmed by at least $n-t$ ($> t$) players in $\mathcal{P}_{\text{acc}}$, hence at least one honest player P_i (holding $m_i = m$) has confirmed m_j, what means that with overwhelming probability, $m_j = m$.

(6) Validity: As shown in the proof of (5), all honest happy players $P_i \in \mathcal{P}_{\text{ok}} \cap \mathcal{P}_{\text{honest}}$ set $m_i' = m$, where m is the message held by every honest player in $\mathcal{P}_{\text{acc}}$.

(7) Unambiguity: Consider the set $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$. This set consists of pairs $\{P_i, \mathcal{P}_j\}$ with $P_i = \Phi(P_j)$, where P_j's hash was rejected by at least $n-t$ players in $\mathcal{P}_{\text{acc}}$, hence by at least one honest player ($n-t > t$). Hence either P_i is corrupted (and did not send the message $m_i = m$ to P_j), or P_j is corrupted (and did not broadcast the correct hash value of the received message). As at least half of the players in $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ are corrupted, but the majority of players in $\mathcal{P}$ is honest, it follows that the majority of players in $\mathcal{P}_{\text{ok}}$ is honest.

4.3 Communication Complexity

The consolidation protocol communicates $\ell n + 2n\mathcal{B}(\kappa) + n\mathcal{B}(n)$ bits.

5. CLAIMING STAGE

In the claiming stage, the players in $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$ must have the possibility to receive the message. However, we cannot allow these players to ask *any* player in $\mathcal{P}_{\text{ok}}$ for the message, because corrupted players could misuse this freedom to ask every player for the message, which would imply the communication of $\Omega(\ell n^2)$ bits. We apply a trick to reduce the communication complexity.

The message m is transformed into a polynomial f_m over $\text{GF}(2^c)$ with degree $d-1$ for appropriate values c and d.[7] In order to compute the polynomial and thus the message, one needs d points on this polynomial, so we set $d = \lceil (|\mathcal{P}_{\text{ok}}| + 1)/2 \rceil$, and $c = \lceil (\ell + 1)/d \rceil$. Every player $P_i \in \mathcal{P}_{\text{ok}}$ sends $f_m(i)$ to every player in $\mathcal{P} \setminus \mathcal{P}_{\text{ok}}$. The receiving player can (with help of the other players in $\mathcal{P}_{\text{ok}}$) distinguish good points from bad points, and ends up with $\lceil (|\mathcal{P}_{\text{ok}}| + 1)/2 \rceil$ good points, which help him interpolate f_m and derive m. The protocol for the claiming stage is given in the box "Protocol Claiming".

Protocol Claiming

1. Every $P_i \in \mathcal{P}_{\text{ok}}$ computes c, d, and f_m as described above, and sends the c-bit piece $y_i = f_m(i)$ to every $P_j \in (\mathcal{P} \setminus \mathcal{P}_{\text{ok}})$.
2. Every $P_i \in \mathcal{P}_{\text{ok}}$ selects at random a key k for a universal hash function U_k, computes the hash value $h_i = (k, U_k(f_m(1)), \ldots, U_k(f_m(n)))$, and sends it to every $P_j \in (\mathcal{P} \setminus \mathcal{P}_{\text{ok}})$.
3. Every $P_j \in (\mathcal{P} \setminus \mathcal{P}_{\text{ok}})$ checks each piece y_i (for each i with $P_i \in \mathcal{P}_{\text{ok}}$) against the i-th entry of every hash value h_m (for each m with $P_m \in \mathcal{P}_{\text{ok}}$). If more than $|\mathcal{P}_{\text{ok}}|/2$ of the hash values match a piece y_i, then the piece is accepted, otherwise rejected. Then P_j interpolates the polynomial f_m from the (at least) d accepted pieces y_i and derives his output message m_j'.

5.1 Security Analysis

The majority of players in $\mathcal{P}_{\text{ok}}$ is honest, so a good piece y_i of an honest player $P_i \in \mathcal{P}_{\text{ok}}$ will be confirmed by the majority of players in $\mathcal{P}_{\text{ok}}$. Hence, P_j will accept all pieces y_i of honest players P_i. As all honest players P_i hold the same message $m_i = m$, all these pieces indeed lie on a polynomial of degree $d-1$, and the d pieces allow a unique interpolation of this polynomial, and a correct reconstruction of the message m.

5.2 Communication Complexity

The claiming protocol communicates $(\ell + n)n + n^3\kappa$ bits.

6. COMPLEXITY ANALYSIS

The communication complexity of each stage is given in the following table:

Checking	$2n\mathcal{B}(\kappa) + n\mathcal{B}(n)$ bits
Consolidation	$\ell n + 2n\mathcal{B}(\kappa) + n\mathcal{B}(n)$ bits
Claiming	$(\ell + n)n + n^3\kappa$ bits
Total	$4n\mathcal{B}(\kappa) + 2n\mathcal{B}(n) + 2\ell n + n^3\kappa + n^2$ bits

[7] Again, we append a 1-bit to the message.

This is less than $2\ell n + 6n\mathcal{B}(n+\kappa) + n^3\kappa + n^2$. As all broadcast protocols in the literature have complexity at least n^2 per broadcasted bit, we can write the complexity of our consensus protocol as $\mathcal{O}(\ell n + n\mathcal{B}(\kappa + n))$. One can easily transform this consensus protocol into a broadcast protocol with the same complexity.

The following table states the overall bit complexities for agreeing on an ℓ-bit message with security parameter κ, both for the most efficient protocols in the literature and for the protocol presented in this paper (instantiated with the most efficient BA protocol). Note that "our" protocol has the same complexity for consensus and for broadcast.

Model	**Lit.**	**Broadcast Consensus**	**Ours**
no PKI	[BGP92], [CW92]	$\Omega(\ell n^2)$ $\Omega(\ell n^2)$	$\mathcal{O}(\ell n + n^3(n+\kappa))$
i.t. PKI	[PW96]	$\Omega(\ell n^2 + n^6\kappa)$ $\Omega(\ell n^3 + n^7\kappa)$	$\mathcal{O}(\ell n + n^7\kappa)$
crypt. PKI	[DS83]	$\Omega(\ell n^3\kappa)$ $\Omega(\ell n^4\kappa)$	$\mathcal{O}(\ell n + n^4(n+\kappa))$

The following table indicates a lower bound on the message size ℓ such that our protocol is strictly better than previously achieved.

Model	ℓ
no PKI	$\omega(n(n+\kappa))$
i.t. PKI	$\omega(n^5\kappa)$
crypt. PKI	$\omega(n^2(n+\kappa))$

7. LOWER BOUNDS

In this section we prove that any broadcast or consensus protocol which tolerates a constant fraction of the players to be corrupted (i.e., $t \in \Omega(n)$), requires the communication of at least $\Omega(\ell n)$ bits for agreeing on an ℓ-bit message.

7.1 The Broadcast Case

The broadcast case is trivial: Every player $P_i \in \mathcal{P}$ must learn the ℓ-bit message, hence the protocol requires the communication of $\Omega(\ell n)$ bits.

7.2 The Consensus Case

In the consensus case, consider the following setting: In a protocol with n players, all but t players start with the same message m. If these t players are corrupted, then the protocol would not need to communicate at all, and every player could output m. If these t players are honest, then the protocol would not need to communicate at all, and every player could output, say, $\perp$. However, these two cases are indistinguishable. Hence the protocol must ensure that the t players not knowing the message learn the message. Therefore, each of these t players must receive the message, which (for $t \in \Omega(n)$) requires the communication of $\Omega(\ell n)$ bits.

8. CONCLUSIONS & OPEN PROBLEMS

We have proposed a reduction that converts any broadcast protocol into a highly efficient consensus protocol for long messages. The reduction is constant-round and information-theoretically secure, for a negligibly small error probability.

The reduction immediately yields broadcast and consensus protocols for n players, in which the message is communicated only $\mathcal{O}(n)$ times. This improves on all previous known protocols that require the message to be transmitted at least $\Omega(n^2)$ times. Communicating the message only $\mathcal{O}(n)$ times is optimal both for broadcast and for consensus.

It seems that the reduction is also possible in the asynchronous world; however, this is left as an open problem.

9. ACKNOWLEDGMENTS

Special thanks go to Krzysztof Pietrzak for his help for making the reduction constant-round, and to Thomas Holenstein for many fruitful discussions on further optimizations and analysis of the protocol.

10. REFERENCES

[BGP92] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. *Computer Science Research*, pp. 313–322, 1992. Preliminary version appeared in STOC '89.

[CR90] D. Chaum and S. Roijakkers. Unconditionally secure digital signatures. In *CRYPTO '90*, LNCS 537, pp. 206–214. Springer, 1990.

[CW79] L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences (JCSS)*, 18(4):143–154, 1979. Preliminary version appeared in STOC '77.

[CW92] B. A. Coan and J. L. Welch. Modular construction of a Byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, Mar. 1992. Preliminary version appeared in PODC '89.

[DR85] D. Dolev and R. Reischuk. Bounds on information exchange for Byzantine agreement. *Journal of the ACM*, 32(1):191–204, Jan. 1985.

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

[FM89] P. Feldman and S. Micali. An optimal probabilistic algorithm for synchronous Byzantine agreement. *Automata, languages and programming*, 372:341–378, 1989. Preliminary version appeared in STOC '88.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.

[PW96] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t >= n/3$. Technical report, IBM Research, 1996.

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