Efficient Byzantine Agreement Secure Against General Adversaries *

(Extended Abstract)

Matthias Fitzi and Ueli Maurer

Department of Computer Science
Swiss Federal Institute of Technology (ETH), Zurich
CH-8092 Zurich, Switzerland,
{fitzi,maurer}@inf.ethz.ch

Abstract. This paper presents protocols for Byzantine agreement, i.e. for reliable broadcast, among a set of n players, some of which may be controlled by an adversary. It is well-known that Byzantine agreement is possible if and only if the number of cheaters is less than $n/3$. In this paper we consider a general adversary that is specified by a set of subsets of the player set (the adversary structure), and any one of these subsets may be corrupted by the adversary. The only condition we need is that no three of these subsets cover the full player set. A result of Hirt and Maurer implies that this condition is necessary and sufficient for the existence of a Byzantine agreement protocol, but the complexity of their protocols is generally exponential in the number of players. The purpose of this paper is to present the first protocol with polynomial message and computation complexity for any (even exponentially large) specification of the adversary structure. This closes a gap in a recent result of Cramer, Damgård and Maurer on applying span programs to secure multi-party computation.

Key words. Broadcast, Byzantine agreement, general adversary, multi-party computation, fault detection.

1 Introduction

1.1 Byzantine Agreement

In this paper, we focus on unconditionally secure protocols for Byzantine agreement, i.e. the security does not rely on any computational assumptions. We consider the standard model of a complete (fully connected) synchronous network among a set P of n players with pairwise authenticated communication channels.

* Appeared in the Proceedings of the Twelfth International Symposium on Distributed Computing (DISC), 1998. Research supported by the Swiss National Science Foundation (SNF), SPP project no. 5003-045293

With respect to this model, the goal of a broadcast (or Byzantine agreement) protocol is to let some specific player, the dealer, reliably distribute some value to all other players. Even in the presence of an adversary that corrupts certain players the protocol must satisfy the conditions given in Definition 1. In the sequel, corrupted players are called *faulty* whereas uncorrupted players are called *correct*.

Definition 1. *A protocol* achieves Byzantine agreement *(or* is a broadcast protocol*) if and only if the following conditions are satisfied:*

1. Termination: *After a finite number of rounds, every correct player decides on a value, i.e. the protocol terminates for all correct players.*
2. Agreement: *All correct players decide on the same value.*
3. Validity: *If the dealer is correct then all correct players decide on the dealer's original value.*

A broadcast protocol secure against an adversary who can corrupt up to arbitrary t of the involved players is called *t-resilient*.

Pease, Shostak and Lamport proved in [PSL80,LSP82] that the condition $t < n/3$ is a tight bound for the existence of a Byzantine agreement protocol. The protocol they proposed requires exponential message complexity and hence is not practical. However, their protocol is round-optimal, as Fischer and Lynch proved in [FL82] that the lower bound on the number of rounds for any perfectly secure Byzantine agreement protocol with t corrupted players is $t+1$. Efficient protocols for Byzantine agreement with resilience $t < n/3$ have been proposed in [DFF+82,BDDS87,FM88,BGP89]. Finally Garay and Moses [GM93] presented the first protocol that is both efficient and round-optimal.

1.2 General Adversaries

Ben-Or, Goldwasser and Wigderson [BGW88] and independently Chaum, Crépeau and Damgård [CCD88] proved that, in a threshold setting where up to t arbitrary players can be corrupted by an adversary, secure multi-party computation among a set of n players is possible for any function if and only if less than a third of the players are actively corrupted ($t < n/3$). Hirt and Maurer [HM97] extended this result to general (rather than threshold) adversary structures. An adversary structure is a monotone set of subsets of the player set. A similar concept was proposed independently by Malkhi and Reiter in [MR97] in the context of quorum systems. As a strict generalization of the threshold setting, any player subset of the adversary structure may be corrupted in the general adversary setting. The threshold setting ($t < n/3$) is a special case of the general setting where the adversary structure consists of all player subsets of cardinality at most t. As proven in [HM97], multi-party computation secure against a general adversary is possible for any function if and only if no three elements (player subsets) of the adversary structure cover the full player set — this is also a tight bound for the existence of a broadcast protocol as immediately follows by the optimality proof in [LSP82].

Example 1. In order to see that it is useful to handle the general adversary model for multi-party computation (or for broadcast in particular) consider, for example, the player set $P = \{d, e, f, g, h, i\}$.[1] As defined later in the text, we describe an adversary structure $\mathcal{A}$ by its basis $\overline{\mathcal{A}}$ where only all maximal player subsets of $\mathcal{A}$ are contained.

In the threshold setting, according to the condition $t < n/3$, at most one of the players in P can be tolerated to be corrupted by the adversary. Hence the (maximal) adversary structure tolerated by a threshold protocol is defined by the basis

$$\overline{\mathcal{A}} = \{\{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}\}.$$

In contrast, by considering general adversaries, strictly more corruptions can be tolerated. Consider the adversary structure defined by the basis

$$\overline{\mathcal{A}} = \{\{d, e, f\}, \{d, g\}, \{e, h\}, \{e, i\}, \{f, g\}\}.$$

No three elements of $\mathcal{A}$ cover the full player set. Hence, by the result in [HM97], there is a multi-party computation for every function that tolerates any player subset in $\mathcal{A}$ to be corrupted. Besides several player pairs that are tolerated to be corrupted, the protocol is even resilient against a player triplet ($\{d, e, f\}$). Note that for every single player there is at least one second player that can be tolerated to be corrupted at the same time.

1.3 Contributions of this Paper

Broadcast protocols have been studied extensively for the threshold setting but not for the general adversary setting introduced in [HM97]. Broadcast is a special case of multi-party computation. Hence, the protocol of [HM97] for general multi-party computation can be directly used to implement broadcast. A different such broadcast protocol is described in [MR97]. However, the computation and message complexities of these protocols are generally exponential in the number of players.

Efficient broadcast is an important primitive used as a subprotocol in various applications such as secure multi-party computation. For example, the results of Cramer, Damgård and Maurer in [CDM98] rely on the existence of a broadcast protocol, secure against a general adversary, with computation and message complexity polynomial in the number of players — without giving a solution for this problem. This paper closes this gap by proposing an algorithm to construct a secure broadcast protocol for any general adversary structure satisfying that no three elements of the structure cover the full player set. The resulting protocols have communication complexity polynomial in the number n of players. The computation complexity is polynomial in n assuming only that there exists an algorithm polynomial in n for deciding whether a given subset of the players is an element of $\mathcal{A}$.

[1] Throughout we denote the players with single letters and start enumerating them by d. In the context of broadcast d will be used to name the dealer.

Finally, some techniques for extending threshold protocols to the general adversary setting proposed in this paper are generic and can also be applied to other broadcast protocols designed for the threshold setting.

1.4 Definitions

The set of players involved in the broadcast protocol is denoted by P. An *adversary structure* $\mathcal{A}$ over the player set P is a monotone set of subsets of P, i.e.

$$\mathcal{A} \subseteq 2^P \text{ with } A \in \mathcal{A} \Longrightarrow \forall A' \subseteq A : A' \in \mathcal{A}.$$

An element $A \in \mathcal{A}$ is called an *adversary set*. For an adversary structure $\mathcal{A}$, $\overline{\mathcal{A}}$ denotes the *basis* of $\mathcal{A}$, i.e. the set of all maximal elements of $\mathcal{A}$:

$$\overline{\mathcal{A}} := \{A \in \mathcal{A} \,|\, \nexists A' \in \mathcal{A} \ : \ A \subset A'\} \ .$$

The meaning of an adversary structure $\mathcal{A}$ is that the players of exactly one element of $\mathcal{A}$ may be corrupted by the adversary.

Definition 2. *A broadcast protocol that, according to Definition 1, is secure against a general adversary that corrupts the players of an arbitrary element of a structure $\mathcal{A}$ is called $\mathcal{A}$-*resilient.

Later we will consider the properties of subsets $S \subset P$ rather than of the whole player set P. In particular, we will be interested in the properties of the structure $\mathcal{A}$ when *restricted* to some player subset $S \subset P$, i.e. by reducing all adversary sets $A \in \mathcal{A}$ to the players in S:

$$\mathcal{A}|_S := \{A \cap S \mid A \in \mathcal{A}\}.$$

Definition 3. *An adversary structure $\mathcal{A}$ over a player set P satisfies the predicate $Q^{\mathsf{k}}(P, \mathcal{A})$ if no k sets in $\mathcal{A}$ cover the full player set:*

$$Q^{\mathsf{k}}(P, \mathcal{A}) \quad :\Longleftrightarrow \quad \forall A_{i_1}, A_{i_2}, ..., A_{i_k} \in \mathcal{A} : \bigcup_{j=1}^{k} A_{i_j} \neq P.$$

We will mostly be interested in these predicates applied to subsets $S \subset P$ rather than to P. Instead of writing $Q^{\mathsf{k}}(S, \mathcal{A}|_S)$ for "no k adversary sets (restricted to S) cover the player subset S" we will use the shorthand notation $Q^{\mathsf{k}}(S, \mathcal{A})$ or even $Q^{\mathsf{k}}(S)$ if $\mathcal{A}$ is evident from the context.

1.5 Outline

In Section 2 simple but inefficient broadcast protocols secure against general adversaries are discussed, more precisely, for any given player set P and adversary structure $\mathcal{A}$ satisfying $Q^3(P, \mathcal{A})$ we describe an $\mathcal{A}$-resilient protocol — we refer to a particular protocol of this kind as a *basic protocol.* A basic protocol is obtained by modifying the "exponential" threshold protocol of [BDDS87]. Section 3 describes efficient broadcast protocols based on the basic protocols of Section 2, i.e. for any given player set P and adversary structure $\mathcal{A}$ satisfying $Q^3(P, \mathcal{A})$ we describe an $\mathcal{A}$-resilient protocol with communication and computation complexity polynomial in the number of players — a particular protocol of this kind will be referred to as an *efficient protocol.* The paper ends with some conclusions in Section 4.

2 The Basic Broadcast Protocol

Using the terminology of [BDDS87], the basic protocol proceeds in two subsequent phases, *information gathering (IG)* and *data conversion (DC).* The information gathering phase consists of a fixed number of communication rounds among the players. Whenever a player p is expected to send a message to some other player q but fails to do so, then q decides for the default value 0 to be his received value. In the data conversion phase, every player locally computes his result (i.e. the broadcast value he decides on) with no further communication. For the sequel, let P be the player set with cardinality $|P| = n$ and let d be the dealer. The remaining players will be enumerated by the letters e, f, etc.

2.1 Information Gathering

Every player maintains a local tree called *information gathering* tree or *IG-tree* for short which is used to keep track of all messages received during information gathering. The structure of this tree is exactly the same for every player. Every node of the IG-tree corresponds to a player $p \in P$. The nodes are labelled by a list α of players — the list of all players corresponding to the nodes in the path from the root to the node itself, i.e. the last element of this list is the node's corresponding player.

Throughout this paper, players are denoted by small Roman letters whereas lists of players are denoted by small Greek letters (or by strings if their elements are mentioned explicitly). For a node αp, player p is called the *corresponding player* of αp and, conversely, we call αp a *corresponding node* of player p. The set of all players corresponding to a node α or to one of its predecessor nodes are called the *corresponding players of α's message path.* Let the *height* of a node α in the IG-tree be defined as its number of predecessors in the tree and let the *tree level* of a node α be $\mathit{level}(\alpha) = \mathit{height}(\alpha) + 1$ (hence the root node is of level 1).

The root node of the IG-tree corresponds to the dealer d. Every node α of some tree level k is either a leaf or an internal node with $n - k$ children — one child αp for each player $p \in P$ that does not occur in α. A node α is defined to be an internal node exactly if there exists an adversary set $A \in \mathcal{A}$ that contains all players in α, else the node is defined to be a leaf. Hence every path from the root to a leaf consists of a sequence of internal nodes that correspond to the

Fig. 1. IG-Tree for Example 1

players of an adversary set $A \in \mathcal{A}$ and finally of a leaf corresponding to a player $p \notin A$. There is no adversary set $\mathcal{A}'$ containing all players corresponding to the nodes in the path ($\nexists \mathcal{A}' : \mathcal{A}' \supseteq \mathcal{A} \cup \{p\}$). Note that the height of the IG-tree is by one greater than the cardinality of an adversary set $A \in \mathcal{A}$ of maximal size (and hence at most n since $P \notin \mathcal{A}$). Figure 1 illustrates this construction for the adversary structure $\mathcal{A}$ over the player set $P = \{d, e, f, g, h, i\}$ of Example 1 (defined by the basis $\overline{\mathcal{A}} = \{\{d, e, f\}, \{d, g\}, \{e, h\}, \{e, i\}, \{f, g\}\}$).

The IG-tree describes the local view of the message flow during information gathering by the individual players. The nodes of the tree are used to store the received messages whereas the edges of the tree describe the dependencies of these messages on each other. Every message received during round k is stored in exactly one node of level k. The local tree of player p is denoted by $tree_p$. The value that player p stores at the node α of his IG-tree is denoted by $tree_p(\alpha)$.

In the first communication round of information gathering, the dealer d distributes his original value (i.e. the value to be broadcast) to every other player and decides on this value as his result of the broadcast. The dealer will not be involved in any subsequent communication round. Every player p stores the value received by the dealer at the root node d of his IG-tree. According to the structure of the IG-tree, in the k-th round ($k > 1$) of the protocol every player p distributes to every other player the value $tree_p(\alpha)$ for each node α of level $k - 1$ that has a child node αp and a receiver r stores this message at this node αp of his IG-tree.[2]

Hence, any message received by a player p originates from the dealer and is passed on step by step through some specific message path involving other players[3], and such a message is stored in the node that is labelled by the string corresponding to its message path (excluding the receiver). For example assume the IG-tree in Figure 1 to be $tree_g$. Then the meaning of the value $tree_g(defi) = x$ stored by player g is that "player i told g that player f told i that player e told f that the dealer d told e that x was his original value".

[2] Actually, messages must consist of a value and additional information about the node of the IG-tree which is referred to by the message. In the sequel we just assume a message to consist of this value and we neglect this additional information.

[3] Note that faulty intermediary players may alter the contents of the message.

2.2 Data Conversion

After information gathering, in the data conversion phase, every player p computes his result (i.e. the broadcast value he decides on) on his IG-tree in a "bottom up manner" by starting with the leaves and recursively, for every internal node, computing a value that is determined by the values previously computed for its child nodes. The function to perform this computation is called *resolve*. In order to define the function *resolve*, we introduce the new value $\perp$ which is defined to be outside the domain of the value to be broadcast. $\perp$ will only result for nodes with its corresponding player being faulty, i.e. it is used to indicate "severe" inconsistencies for such a node. Indication of inconsistencies is only needed for the efficient protocol in Section 3 and the resolve-function could in fact be simplified for the basic protocol discussed here.

Definition 4. *The set of all players corresponding to the children of a node α of the IG-tree is denoted by $C(\alpha)$. The subset of all correct players among the players corresponding to the children nodes of α is denoted by $C_c(\alpha)$.*

The resolve function takes a node of the IG-tree as an input and outputs the value to be assigned as the resolved value for the given node (described here for the local tree of player p):

$$resolve_p(\alpha) := \begin{cases} tree_p(\alpha) & \text{if } \alpha \text{ is a leaf,} \\ v & \exists!\ v \neq \perp\colon Q^1(\{c \in C(\alpha) \mid resolve_p(\alpha c) = v\}), \\ 0 & \text{else if } \alpha = d, \\ \perp & \text{else.} \end{cases}$$

In other words, an internal node α is resolved to the value v if there is a unique value v such that the children of α resolving to v are not covered by any adversary set $A \in \mathcal{A}$ (more precisely, if the set of their corresponding players satisfies Q^1) — else the default value $\perp$ (or 0) is assigned. The distinction between the last two cases in the definition of the resolve function is only needed for the correctness of the efficient protocol in Section 3 (besides this, we want the broadcast value to be in the original domain).

Since $\perp$ is not in the domain of the broadcast, a correct player is not to distribute $\perp$ during information gathering. If, for some node, a correct player receives any value outside the domain during information gathering he will store the default value 0 instead.

2.3 Protocol Analysis

Definition 5. *A node α of the IG-tree is* common with value v *if every correct player computes the same value v for α in the data conversion phase. The subtree rooted at node α has a* common frontier *if every path from α to a leaf contains at least one common node.*

Lemma 1. *For any adversary structure $\mathcal{A}$ over a player subset $S \subseteq P$ satisfying $Q^{\mathsf{k}}(S, \mathcal{A})$ and any adversary set $A \in \mathcal{A}$, the restricted adversary structure $\mathcal{A}|_{(S \setminus A)}$ satisfies $Q^{\mathsf{k}\text{-}1}(S \setminus A, \mathcal{A}|_{(S \setminus A)})$.*

Proof. Let $A \in \mathcal{A}$ be an adversary set. Then $Q^{\mathsf{k}}(S, \mathcal{A})$ implies $\nexists A_{i_1}, \cdots, A_{i_{k-1}}$: $A \cup A_{i_1} \cup \cdots \cup A_{i_{k-1}} = P$ and hence $\nexists A_{i_1}, \cdots, A_{i_{k-1}} : A_{i_1} \cup \cdots \cup A_{i_{k-1}} = P \setminus A$ which implies $Q^{\mathsf{k}\text{-}1}(S \setminus A, \mathcal{A}|_{(S \setminus A)})$.

Corollary 1. *If the adversary structure $\mathcal{A}$ over the player set P satisfies $Q^{\mathsf{k}}(P, \mathcal{A})$ then for every internal node α of the IG-tree, $\mathcal{A}|_{C(\alpha)}$ satisfies $Q^{\mathsf{k}\text{-}1}(C(\alpha), \mathcal{A}|_{C(\alpha)})$.*

Proof. By construction, for every internal node α there is an adversary set $A \in \mathcal{A}$ that contains exactly the players corresponding to α's message path, and $C(\alpha) = P \setminus A$. Hence, by Lemma 1, we have $Q^{\mathsf{k}\text{-}1}(C(\alpha), \mathcal{A}|_{C(\alpha)})$.

Lemma 2. *All nodes $\alpha = \beta r$ that correspond to a correct player r are common with the value $tree_r(\beta) = v$.*

Proof. Let p and r be correct players. We have to show that $resolve_p(\alpha) = tree_p(\alpha) = tree_r(\beta)$ for player p. Since r is correct, he distributes the same value $tree_r(\beta) = v$ to all other players, and hence $tree_p(\alpha) = tree_r(\beta) = v$.

It remains to prove that $resolve_p(\alpha) = tree_p(\alpha)$. If α is a leaf then by definition $resolve_p(\alpha) = tree_p(\alpha)$. Assume the lemma holds for all nodes of level k of the IG-tree and let $\alpha = \beta r$ be a node of level $k-1$. By condition $Q^3(P)$ and by Corollary 1 we have $Q^2(C(\alpha))$. The adversary corrupts the players of exactly one adversary set $A \in \mathcal{A}$. But by discounting any such adversary set A from the children player set $C(\alpha)$, the set $C_c(\alpha) = C(\alpha) \setminus A$ of all correct players in $C(\alpha)$ still satisfies $Q^1(C_c(\alpha))$ (by Corollary 1). Since these players are correct their corresponding nodes (of level k) are common with value v by induction, and hence $Q^1(\{c \in C(\alpha) \mid resolve_p(\alpha c) = v\})$. On the other hand, for at most all players $a \in A$ of one adversary set $A \in \mathcal{A}$, $resolve_p(\alpha a) \neq v$ and hence $\neg Q^1(\{c \in C(\alpha) \mid resolve_p(\alpha c) \neq v\})$. Hence by the definition of the resolve function, since v is unique, player p computes $resolve_p(\alpha) = v$.

Lemma 3. *Let α be a node of the IG-tree. If there is a common frontier in the subtree rooted in α then α is common.*

Proof. A leaf node has a common frontier exactly if it is common, hence the lemma holds for all leaves. It remains to prove the lemma for internal nodes. For the sake of contradiction, assume an internal node α not to be common but to have a common frontier. Since α is not common there must be a non-common child of α — otherwise, every correct player would compute the same value for α during data conversion and α would be common. This argument holds for such a non-common node as well, and can be recursively applied down to the leaves of the IG-tree. Thus there is a path from α to one of the leaves of its subtree that contains no common node and hence α has no common frontier. This contradicts the assumption.

Theorem 1. *For any player set P and adversary structure $\mathcal{A}$ satisfying $Q^3(P)$ the basic protocol is $\mathcal{A}$-resilient.*

Proof. By the construction of the IG-tree, for every path from the root d to a leaf node there exists no adversary set $A \in \mathcal{A}$ that contains all players that correspond to the nodes in this path. Hence, on every such path there is a correct player, so by Lemma 2 the root has a common frontier and by Lemma 3 the root is common. Finally, if the dealer is correct, then by Lemma 2, the root node is common with the original value of the dealer.

3 The Efficient Broadcast Protocol

It is evident that for any adversary structure $\mathcal{A}$ with exponential size in $|P|$ the basic protocol is not efficient since the number of messages to be sent is of size $\Omega(|\mathcal{A}|)$ (the size of the IG-tree). In order to make the basic broadcast protocol efficient, we generalize and apply the shifting technique introduced in [BDDS87].

We fix some protocol parameter b with $4 \leq b < n$. The original IG-tree is reduced by pruning all nodes of level $l > b$. This reduced tree defines a protocol with b communication rounds which is called *reduced protocol*. In the reduced protocol, information gathering consists of exactly that subset of messages of the basic protocol for which there exists a node in the reduced tree to store the received message. Data conversion is defined in the same way as for the basic protocol. In the sequel we assume that the original IG-tree is of height $h \geq b$ and that hence the reduced IG-tree is not identical to the original IG-tree of the basic protocol. In the other case we define the efficient protocol to be equal to the basic protocol (since we do not need the protocol extensions described in this section).

3.1 Protocol Overview

The efficient protocol consists of executing the reduced protocol in sequence for $\left\lceil \frac{n-3}{b-3} \right\rceil + 1$ runs as follows[4]: The first run of the reduced protocol proceeds in the same way as the basic protocol (with the difference that it only proceeds for b communication rounds). In every subsequent run (say the m-th run) of the reduced protocol the first communication round of the protocol is omitted. Instead of receiving a value from the dealer, every player p assigns the resolved root value computed at the end of the previous run to the root node of his reduced IG-tree:

$$tree_p^{(m)}(d) := resolve_p^{(m-1)}(d) \ .$$

Apart from this the subsequent runs proceed in the same way as the first one.

The reason for pruning the IG-tree to obtain the reduced protocol is to reduce the communication complexity of the protocol to polynomial in the number of players. However, as a consequence, there is not necessarily a common frontier for the root node when the reduced protocol is run and hence the root node is

[4] Note the difference between the terms *round* and *run*. The term *round* refers to a single communication round of a protocol whereas *run* refers to an execution of the reduced protocol which consists of several communication rounds.

not necessarily common. This problem can be solved by repeating the reduced protocol for a sufficient number of times. The following important properties will be proven:

- If the dealer is correct then the root node is common with his original value after the first run of the reduced protocol (this directly follows since the node d has a common frontier).
- If after some run m of the reduced protocol the root node of the IG-tree is common with some value v then the root node will be common with value v for every subsequent run $m' > m$, i.e. the agreed value will remain persistent.
- There is a method for detecting faulty players such that if after m runs of the reduced protocol the root node is not common, then there are $m(b-3)+1$ players that are globally detected (i.e. detected by all correct players) to be faulty.[5]

Finally, after $\left\lceil \frac{n-3}{b-3} \right\rceil + 1$ runs, either all faulty players have been globally detected or Byzantine agreement has been achieved. If we let every correct player replace all messages from detected players by the default value 0, then Byzantine agreement is also achieved in the former case because all globally detected players are treated like consistently distributing the default value in the last round.

We now describe additional details of the efficient protocol.

3.2 Fault Handling

Every player p maintains a player list L_p and adds to it in each round all players he reliably detects to be faulty. This list of detected players will never contain any correct player.

Let p be a correct player. In order to find conditions for the detection of faulty players by p, we first derive a condition that must be satisfied for every internal node of the IG-tree corresponding to a correct player. Let αr be an internal node corresponding to a correct player r. During information gathering, r distributes the same value $tree_r(\alpha) = v$ for his node α to every player. Accordingly, in the following round, only a faulty player q may send to p a distinct value $w \neq v$ for his node αr. Hence there exists an adversary set $A \in \mathcal{A}$ and a unique value v such that all players q with $tree_p(\alpha r q) \neq v$ are covered by A. Moreover, the set of already detected players L_p must be a subset of such an A, i.e. the condition

$$\exists v : \exists A \in \mathcal{A} : A \supseteq (\{c \in C(\alpha r) \mid tree_p(\alpha r c) \neq v\} \cup L_p)$$

is satisfied. Hence player r can be reliably detected to be faulty by player p if there is no such value v.

This rule can be applied during information gathering: player p adds player r to L_p if there is no such value v. By Lemma 2, this even holds for the converted values of αr's child nodes. Hence we can apply the same rule for the resolved

[5] A player does not necessarily know which of the players he detected are globally detected and which are not.

values as well and we obtain the following fault detection rules to be applied by player p for every internal node αr:

Fault detection during information gathering (FD1):

$$L_p := L_p \cup \{r\} \qquad \text{if } \nexists v : \neg Q^1(\{c \in C(\alpha r) \mid tree_p(\alpha r c) \neq v\} \cup L_p).$$

Fault detection during data conversion (FD2):

$$L_p := L_p \cup \{r\} \qquad \text{if } \nexists v : \neg Q^1(\{c \in C(\alpha r) \mid resolve_p(\alpha r c) \neq v\} \cup L_p).$$

Fault Masking:
After a player r has been added to the list L_p by player p in some communication round k, then every message by player r in round k and any subsequent round is replaced by the default value 0. Once a player has been globally detected (by all correct players), he will be masked to send the same values to all correct players. Thus, every node αr for which a value is received after player r's global detection will be common.

We are ready to summarize the complete protocol (the description is given for the view by player p).

Efficient Protocol:

1. Dealer distributes original value to all players.
2. FOR $i := 1$ TO $\left\lceil \frac{n-3}{b-3} \right\rceil + 1$
3. Information gathering with fault detection FD1 and fault masking for $b-1$ rounds.
4. Local data conversion with fault detection FD2 by every player p.
5. $tree_p(d) := resolve_p(d)$.
6. END.
7. Decide on $tree_p(d)$ for the broadcast result and halt.

3.3 Protocol Analysis

We first show that if in any run (FOR-loop in the above protocol) Byzantine agreement is achieved, then the agreement remains persistent (with the same value) for any following run. The following lemma follows immediately by Lemma 2.

Lemma 4. *If all correct players store the same value at the root of the IG-tree at the beginning of information gathering (i.e. before step 3 of some run of the reduced protocol) then the root will be common with this value after data conversion.*

The following lemmas will be needed to argue about global detections of faulty players.

Lemma 5. *Let α be an internal node. If all players corresponding to α's message path are faulty, then the subset $C_c(\alpha) \subseteq C(\alpha)$ of all correct players among the players corresponding to the children of α satisfies $Q^2(C_c(\alpha))$.*

Proof. By construction, $C(\alpha)$ consists of all players except for the players corresponding to α's message path. Let $A \in \mathcal{A}$ contain all players that correspond to α's message path and let A be the corrupted adversary set. Then $C_c(\alpha) = C(\alpha)\setminus A = P\setminus A$ holds and by Lemma 1 we have $Q^2(P\setminus A)$ and hence $Q^2(C_c(\alpha))$.

Lemma 6. *Let p and q be correct players and let αr be an internal node of the IG-tree but not the root node. Assume all players corresponding to αr's message path to be faulty. If p and q obtain different values for αr after data conversion neither of which is $\perp$, then $r \in L_p \cap L_q$.*

Proof. For any value v let $C_v \subseteq C_c(\alpha r)$ denote the set of correct players c for which $tree_p(\alpha rc) = v$ and let $C_{\overline{v}} = C_c(\alpha r)\setminus C_v$.

Suppose that $r \notin L_p$ after data conversion. According to the fault detection rule FD1, since p has not detected r to be faulty, there is a value v such that $\neg Q^1(\{c \in C(\alpha r) \mid tree_p(\alpha rc) \neq v\} \cup L_p)$. This still holds when restricted to all correct players in $C(\alpha r)$: $\neg Q^1(C_{\overline{v}} \cup L_p)$.

By Lemma 5 the correct children of αr satisfy $Q^2(C_c(\alpha r))$. Since there is an adversary set $A \in \mathcal{A}$ with $C_{\overline{v}} \subseteq A$, we obtain $Q^1(C_v)$ for $C_v = C_c(\alpha r)\setminus C_{\overline{v}}$ by Lemma 1.

The correct children of αr are common due to Lemma 2 and hence all correct players will resolve value v for the children nodes corresponding to the players in C_v:

$$Q^1\big(\{c \in C_c(\alpha r) \mid resolve_p(\alpha rc) = resolve_q(\alpha rc) = v\}\big).$$

By the uniqueness condition in the definition of the *resolve* function it follows that $resolve_q(\alpha r) \in \{v, \perp\}$. This contradicts the assumption.

Lemma 7. *Let αr be an internal node of the IG-tree, but not the parent of a leaf. If all players corresponding to the path of αr are faulty and there is a correct player p who does not detect r to be faulty by either of the fault detection rules, then αr is common.*

Proof. Since p does not detect r during data conversion, there is a value v with $resolve_p(\alpha r) = v$ and

$$\neg Q^1\big(\{c \in C(\alpha r) \mid resolve_p(\alpha rc) \neq v\} \cup L_p\big).$$

Let q be any other correct player and let $s \notin L_p$ be a player with $resolve_p(\alpha rs) = v$. By Lemma 6 $resolve_q(\alpha rs) \in \{v, \perp\}$ (in fact v by Lemma 2 if s is correct), and hence for any $w \notin \{v, \perp\}$, the subset of the children of αr for which w is resolved by player q satisfies

$$\{c \in C(\alpha r) \mid resolve_q(\alpha rc) = w\} \quad \subseteq \quad \{c \in C(\alpha r) \mid resolve_p(\alpha rc) \neq v\} \cup L_p.$$

Therefore for all values $w \notin \{v, \perp\}$ we have

$$\forall w, w \notin \{v, \perp\} : \neg Q^1(\{c \in C(\alpha r) \mid resolve_q(\alpha rc) = w\}).$$

On the other hand, by Lemma 5 we have $Q^2(C_c(\alpha r))$ and hence $Q^1(\{c \in C_c(\alpha r) \mid resolve_p(\alpha rc) = v\})$ for all correct children for which player p resolves v. These children are common by Lemma 2. Hence q computes $resolve_q(\alpha r) = v$ because v is a unique value according to the definition of the resolve function.

Due to Lemma 7, after the initial run of the reduced protocol of $b \geq 4$ rounds, either the root is common or at least $b-2$ faulty players have been globally detected: If the root is not common then there is no common frontier and hence, by Lemma 2, there is a path from the root to a leaf in the IG-tree that contains only non-common nodes (corresponding to faulty players). All of these corresponding players are globally detected — except for the leaf and its parent node which are not necessarily detected.

In every subsequent run, if Byzantine agreement is not achieved, then further $b-3$ players are globally detected. Nodes that correspond to players that have been globally detected will be common since fault masking guarantees that all correct players consistently store the default value at such nodes. Hence it is guaranteed that the detections of a new run correspond to players that have not been detected before. On the other hand, the dealer does not distribute any values after the first run of the basic protocol because the values for the root node are locally computed by the players. Hence, there is no masking for the dealer and the dealer will be redetected in every subsequent run not achieving Byzantine agreement. This is the reason why only $b-3$ further global detections are guaranteed in the subsequent runs.

Theorem 2. *For any player set P and adversary structure $\mathcal{A}$ satisfying $Q^3(P)$ the efficient protocol achieves Byzantine agreement. The algorithm has message complexity polynomial in the size n of the player set and round complexity less than $2n$. Assuming an algorithm polynomial in n for deciding whether a given player set is an element of $\mathcal{A}$, the computation complexity is polynomial in n.*

Proof. Suppose that Byzantine agreement has not been achieved after the given protocol has terminated. According to Lemma 4, agreement was not achieved in any previous round, i.e. the root of the IG-tree has never been common. Hence, due to Lemma 7, $b-2$ faulty players have been detected in the first run and for every subsequent run, another $b-3$ players have been globally detected. Hence the number k of globally detected players is

$$k = (b-2) + \left\lceil \frac{n-3}{b-3} \right\rceil (b-3) \geq (b-2) + (n-3) \geq n-1.$$

Since at least $n-1$ players have been globally detected, there is at most one single correct player. Hence, by definition, all nodes of the IG-tree are common (for all correct players), contradicting the assumption.

The message and computation complexities follow immediately from the construction of the protocol. The round complexity is $r = b + (b-1)\left\lceil \frac{n-3}{b-3} \right\rceil < 2n$.

3.4 Optimizations

The efficient protocol can be optimized. For simplicity the protocol was described in a way that (except for the dealer) every message that is distributed, must be distributed to every player. In fact this is not necessary. If a player p must distribute a value $tree_p(\alpha)$ of his IG-tree then it suffices that he distributes this value only to those players that are not corresponding to α or any of its predecessor nodes. In other words, the IG-tree of a player p will contain no node $\alpha = \beta p \gamma$ with $|\gamma| > 0$. The efficient protocol can further be optimized by analyzing the concrete structure of the given adversary set which can yield a protocol that needs less runs of the basic protocol than described above.

4 Conclusions

For any adversary structure $\mathcal{A}$ for a player set P for which no three elements of $\mathcal{A}$ cover the full player set we have given an $\mathcal{A}$-resilient broadcast protocol with communication complexity polynomial in the cardinality $n = |P|$ of the player set. The computation complexity is polynomial in n assuming only the existence of an algorithm polynomial in n for deciding whether a given subset of the players is an element of $\mathcal{A}$. Moreover, our methods for generalizing the threshold protocol of [BDDS87] are universal and we expect them to be directly applicable for other broadcast protocols such as those of [FM88,GM93].

5 Acknowledgments

The authors would like to thank Ronald Cramer, Ivan Damgård, Juan Garay and Martin Hirt for helpful hints and interesting discussions.

References

[BDDS87] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 42–51, 1987.

[BGP89] P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 410–415. IEEE, 1989.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pages 1–10, 1988.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pages 11–19, 1988.

[CDM98] R. Cramer, I. Damgård, and U. Maurer. Span programs and general secure multi-party computation, Manuscript, 1998.

[DFF$^+$82] D. Dolev, M. J. Fischer, R. Fowler, N. A. Lynch, and H. R. Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 52(3):257–274, March 1982.

[FL82] M. J. Fischer and N. A. Lynch. A lower bound on the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

[FM88] P. Feldman and S. Micali. Optimal algorithms for Byzantine agreement. In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pages 148–161, 1988.

[GM93] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement in $t+1$ rounds (extended abstract). In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 31–41, 1993.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *Proc. 16th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–34, August 1997.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[MR97] D. Malkhi and M. Reiter. Byzantine quorum systems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 569–578, 1997.

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