Multi-Party Computation with Hybrid Security

Matthias Fitzi[1], Thomas Holenstein[2], and Jürg Wullschleger[3]

[1] Department of Computer Science
University of California, Davis
`fitzi@cs.ucdavis.edu`

[2] Department of Computer Science
ETH Zurich, Switzerland.
`holenst@inf.ethz.ch`

[3] Département d'Informatique et Recherche Opŕationnelle
Université de Montréal, Canada.
`wullschj@iro.umontreal.ca`

Abstract. It is well-known that n players connected only by pairwise secure channels can achieve multi-party computation secure against an active adversary if and only if

- $t < n/2$ of the players are corrupted with respect to computational security, or
- $t < n/3$ of the players are corrupted with respect to unconditional security.

In this paper we examine to what extent it is possible to achieve conditional (such as computational) security based on a given intractability assumption with respect to some number T of corrupted players while simultaneously achieving unconditional security with respect to a smaller threshold $t \leq T$. In such a model, given that the intractability assumption cannot be broken by the adversary, the protocol is secure against T corrupted players. But even if it is able to break it, the adversary is still required to corrupt more than t players in order to make the protocol fail.

For an even more general model involving three different thresholds t_p, t_σ, and T, we give tight bounds for the achievability of multi-party computation. As one particular implication of this general result, we show that multi-party computation computationally secure against $T < n/2$ actively corrupted players (which is optimal) can additionally guarantee unconditional security against $t \leq n/4$ actively corrupted players "for free."

Keywords: Broadcast, computational security, multi-party computation, unconditional security.

1 Introduction

Secure distributed cooperation among mutually distrusting players can be achieved by means of general multi-party computation (MPC). Typically, the goal of such a cooperation consists of jointly computing a function on the players' inputs in a way that guarantees correctness of the computation result while keeping the players' inputs private — even if some of the players are corrupted by an adversary.

Different models for MPC have been proposed in the literature with respect to communication, corruption flavor, and adversarial power. In this paper, we restrict our view to the following parameters.

COMMUNICATION: We exclusively consider *synchronous networks* meaning that, informally speaking, the players are synchronized to common communication rounds with the guarantee that a sent message will be delivered still during the same communication round.

CORRUPTION: We assume a *central threshold adversary* with respect to a given, fixed threshold t, meaning that it can select up to arbitrary t out of the n players and corrupt them (meaning to take control over them). Such a player is then said to be *corrupted* whereas a non-corrupted player is called *correct.* An *active adversary* (*active corruption*) corrupts players by making them deviate from the protocol in an arbitrarily malicious way.

SECURITY: A protocol achieves *unconditional security* if even a computationally unbounded adversary cannot make the protocol fail — except for some negligible error probability. A protocol achieves *computational security* if an adversary restricted to probabilistic polynomial time computations cannot make the protocol fail except for some negligible error probability. In this paper, we consider both kinds of security.

1.1 Previous work

The MPC problem was first stated by Yao [Yao82]. Goldreich, Micali, and Wigderson [GMW87] gave the first complete solution to the problem with respect to computational security. For the model with a passive adversary (passive model, for short), and given pairwise communication channels, they gave an efficient protocol that tolerates any number of corrupted players, $t < n$. For the model with an active adversary (active model), and given both pairwise and broadcast channels, they gave an efficient protocol that tolerates any faulty minority, $t < n/2$, which is optimal in the sense that no protocol exists for $t \geq n/2$. Note that when not demanding security to the full extent, computationally secure MPC is also achievable in presence of an active adversary that corrupts $t \geq n/2$ players [GMW87,GHY87,BG89,GL90,FGH$^+$02,GL02,FHHW03]. However, in this case, robustness cannot be guaranteed, i.e., it can not be guaranteed that every player receives a result [Cle86].

With respect to unconditional security, Ben-Or, Goldwasser, and Wigderson [BGW88], and independently, Chaum, Crépeau, and Damgård [CCD88] gave efficient protocols for the passive model that tolerate $t < n/2$ and protocols for the active model that tolerate $t < n/3$ — assuming only pairwise communication channels in both cases. Both bounds are tight. Beaver [Bea89], Rabin, and Ben-Or [RB89] considered the active model when given both pairwise and broadcast channels among the players. They gave efficient protocols that achieve unconditional security for $t < n/2$ which is optimal. A more efficient protocol for this model was given by Cramer et al. [CDD$^+$99].

With lack of better knowledge, protocols with computational security must be based on unproven intractability assumptions, i.e., they must build up on cryptographic primitives such as trapdoor permutations that are not known to exist. Furthermore, even if such primitives existed, the particular choice of a candidate implementation of such a primitive might be a bad one.

In order to prevent complete failure in these cases, Chaum [Cha89] considered a "hybrid" security model for MPC that achieves computational security for some large threshold T but, at the same time, unconditional security for some smaller threshold $t \leq T$ — meaning that, in order to make the protocol fail, the adversary must either corrupt more than T players, or corrupt more than t players but additionally be able to break the underlying computational hardness assumption. In the passive model, given pairwise communication channels, Chaum's protocol achieves computational security with respect to $T < n$ and unconditional security with respect to $t < n/2$. Thus, this protocol simultaneously achieves the optimal bounds for computational and unconditional security.

In the active model, given pairwise and broadcast channels, his protocol achieves computational security with respect to $T < n/2$ and additionally provides unconditional privacy for all players' inputs as long as up to $t < n/3$ players are corrupted. Note that the later results in [Bea89,RB89,CDD^{+}99] strictly imply this result: unconditional security for $t < n/2$ when assuming broadcast.[4] In [WP89], the same "hybrid" model was considered with respect to the simulation of broadcast when given only pairwise communication channels.

1.2 Multi-party computation beyond t < n/3 without broadcast

The active model for MPC tolerating at least $n/3$ corrupted players typically assumes broadcast channels [GMW87,Bea89,RB89]. This is a very strong assumption and might not always be appropriate. Rather, broadcast has to be simulated by the players using the bilateral channels. But, without further assumptions, this simulation is only possible if $t < n/3$ [LSP82,DFF^{+}82].

The only known way to allow for the simulation of broadcast beyond $t < n/3$ is to use digital signatures [LSP82]. However, it is important to note that digital signatures by themselves are not enough. It must be additionally guaranteed that all correct players verify each player's signatures in the same way, i.e., that all players hold the same list of public keys. Otherwise, the transfer of a signature would not be conclusive. We call such a setup a *consistent public-key infrastructure (PKI)*. Such a PKI allows to efficiently simulate broadcast among the players secure against any number of corrupted players, $t < n$ [DS82]. Not only can a PKI be based on a computationally secure digital signature scheme but also on unconditionally secure pseudo-signatures [PW96] and thus allowing for the simulation of *unconditionally* secure broadcast. Thus the results

[4] Note that Chaum's protocol still completely relies on cryptography since the protocol's correctness is only protected by cryptographic means, i.e., by breaking the cryptographic assumption the adversary can make the protocol fail by only corrupting one single player.

in [GMW87,Bea89,RB89] are equally achievable without broadcast channels but with an appropriate PKI to be set up among the players — computationally secure for [GMW87] and unconditionally secure for the other cases.

We believe that assuming a PKI is more realistic than assuming broadcast channels among the players and thus follow this model.

It should be noted, though, that the use of unconditional pseudo-signatures is not very practical. The cost of broadcasting a single bit based on (computational) digital signatures is $t+1$ communication rounds and $O(n^3 s)$ bits to be sent by all players during the protocol overall — where s is the size of a signature [DS83]. The cost of broadcasting a bit using unconditional pseudo-signatures is $\Omega(n^3)$ rounds and $\Omega(n^{17})$ bits to be sent overall [PW96] — which is still polynomial but nevertheless quite impractical.[5]

1.3 Contributions

Typical ways of setting up a PKI are to run a setup protocol among the players or to involve trust management over the Internet such as, e.g., the one in PGP. Evidently, both methods can fail to achieve a consistent PKI, namely, when to many players are corrupted, or, respectively, when the trust management is built on wrong assumptions. Thus, analogously to relying on computational security, relying on the consistency of a previously set-up PKI also imposes a potential security threat since the adversary might have been able to make the PKI inconsistent.

This raises the natural question of whether MPC relying on the consistency of a PKI and/or the security of a particular signature scheme can additionally guarantee unconditional security for the case where only a small number of the players are corrupted. Thus, in this paper, we extend the considerations in [Cha89] (regarding the active model) to the case where not only the adversary might be able to break the underlying hardness assumption but where also the PKI might be inconsistent.

In particular, we consider the following model for *hybrid MPC* involving three thresholds t_p, t_σ, and T, where $t_p, t_\sigma \leq T$ with the following properties (see also Figure 1).

- If at most $f \leq \min(t_p, t_\sigma)$ players are corrupted then we demand *unconditional security.*
- If $f > t_p$ then we assume that the PKI is consistent, i.e., for $t_p < f \leq T$ the computation is only as secure as the PKI.
- If $f > t_\sigma$ then we assume that the adversary cannot forge signatures (except for some non-negligible probability), i.e., for $t_\sigma < f \leq T$ the computation is only as secure as the underlying signature scheme.

[5] Note that there is also a $(t+1)$-round variant with an overall bit complexity of approximately $\Theta(n^6)$. However, this variant is only a one-time signature scheme which basically means that the PKI only allows for a very limited number of signatures to be issued.

f players corrupted	Security
$f \leq \min(t_p, t_\sigma)$	unconditional
$f \leq t_\sigma \wedge t_p < f \leq T$	as secure as PKI, independent of signature scheme
$f \leq t_p \wedge t_\sigma < f \leq T$	as secure as signature scheme, independent of PKI
$t_p, t_\sigma < f \leq T$	as secure as PKI and signature scheme together

Fig. 1. Threshold conditions for hybrid MPC.

Or, in other words, if $f \leq t_p$ then the protocol must be secure even if the PKI is inconsistent, and, if $f \leq t_\sigma$ then the protocol must be secure even if the adversary is able to forge signatures. Thus, in order to make such a hybrid protocol fail with non-negligible probability, the adversary would have to corrupt more than $f = \min(t_p, t_\sigma)$ players and; having made for a bad PKI if $f > t_p$, or be able to forge signatures if $f > t_\sigma$.

Result. We show that hybrid MPC is achievable if and only if

$$(2T + t_p < n) \quad \wedge \quad (T + 2t_\sigma < n) \tag{1}$$

implying that, without loss of generality, we can always assume that $t_p \leq t_\sigma \leq T$.[6] See Figure 2 for a graphical representation of the tight bound. Achievability for all cases will be demonstrated by efficient protocols that neither rely on any particular signature scheme nor on any particular way of setting up a PKI.

As an interesting special case, the optimal result of [GMW87] (assuming a consistent PKI instead of broadcast — allowing to drop parameter t_p since the consistency of the PKI is granted) computationally secure against $T < n/2$ corrupted players additionally allows to guarantee unconditional security against $t_\sigma \leq n/4$ corrupted players "for free." On the other hand, when requiring optimality with respect to unconditional security, $t_\sigma = \lfloor (n-1)/3 \rfloor$, then practically no higher computational bound T can be simultaneously tolerated on top.

Finally, when basing the PKI on an unconditional pseudo-signature scheme (which is not our focus), forgery becomes impossible by definition and the tight bound collapses to $2T + t_p < n$.

Constructions. Our final MPC protocol is obtained by simulating broadcast in the unconditional MPC protocol of [CDD$^+$99]. Thus the main technical contribution in this paper is to simulate broadcast (aka Byzantine agreement) in the given models with respect to the required security aspects. The (efficient) protocol in [CDD$^+$99] is unconditionally secure against $t < n/2$ corrupted players. So, obviously, the final MPC protocol wherein broadcast is simulated is as secure as the given broadcast protocol.

[6] That is, additional forgery gives the adversary no additional power when the PKI is inconsistent.

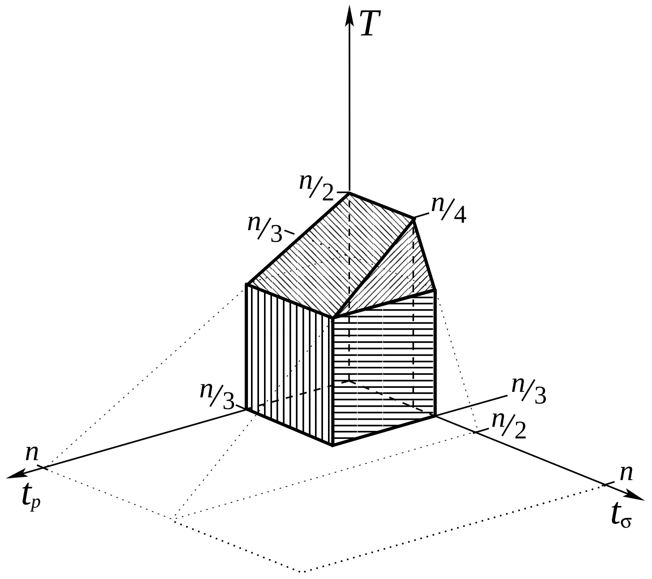

Fig. 2. Graphical representation of Bound (1).

2 Definitions and Notations

2.1 Multi-party computation

In MPC a set of players want to distributedly evaluate some agreed function(s) on their inputs in a way preserving privacy of their inputs and correctness of the computed result. More precisely, in an MPC among a player set P with respect to a collection of functions $(f_1, \ldots, f_n)$, every player $p_i \in P$ holds a secret input (vector) x_i and secretly receives an output (vector) $y_i = f_i(x_1, \ldots, x_n)$.

From a qualitative point of view, the security of MPC is often broken down to the conditions "privacy", "correctness", "robustness", and "fairness", and ideally, a protocol should satisfy all these properties.

PRIVACY. A protocol achieves *privacy* if the adversary cannot learn more about the correct players' inputs than given by the inputs and outputs of the corrupted players.

CORRECTNESS. A protocol achieves *correctness* if the correct players' outputs are indeed computed as defined by the functions f_i.

ROBUSTNESS. A protocol achieves *robustness* if every correct player finally receives his outputs.

FAIRNESS. A protocol achieves *fairness* if the adversary gets no information about the correct players' inputs in case that robustness is not achieved.

More formally, MPC is modeled by an *ideal process* involving a mutually trusted party τ where the players secretly hand their inputs to τ, followed by

τ computing the players' outputs and secretly handing them back to the corresponding players [Bea91,Can00,Gol01]. This model is referred to as the *ideal model* The goal of MPC is now to achieve the same functionality in the so-called *real model* where there is no such trusted party such that an adversary gets no advantage compared to an execution of the ideal protocol. An MPC protocol is defined to be secure if, for every adversary $\mathcal{A}$ in the protocol, there is an adversary $\mathcal{S}$ in the ideal model that, with similar costs, achieves (essentially) the same output distribution as the adversary in the protocol [Bea91,Can00,Gol01].

2.2 Broadcast

Broadcast is the special case of an MPC. In broadcast, one player p_s is given an initial value which everybody is required to receive. The definition is as follows:

Definition 1 (Broadcast). *A protocol among n players, where player $p_s \in P$ (called the* sender*) holds an input value x_s and every player p_i ($i \in \{1, \ldots, n\}$) computes an output value y_i, achieves* broadcast *if it satisfies:*

- VALIDITY: *If the sender is correct then all correct players p_i compute output $y_i = x_s$.*
- CONSISTENCY: *All correct players compute the same output value y.*

Often, it is also added to the definition that the protocol is always required to terminate. We will not mention this property explicitly since termination is obvious for our protocols.

Note that broadcast for any finite domain easily reduces to binary broadcast (where the sender sends a value from $\{0, 1\}$) as, e.g., shown by Turpin and Coan [TC84]. We will thus focus on binary broadcast.

During the simulation of broadcast, using signatures, it must be avoided that previous signatures can be reused by the adversary in a different context, i.e., an independent phase or another instance of the protocol. This fact was observed in [GLR95] and more profoundly treated in [LLR02]. To avoid such "replay attacks" values can be combined with unique sequence numbers before signing. The sequence numbers themselves do not have to be transferred since they can be generated in a predefined manner (encoding the protocol instance and the communication round). However, we will not explicitly state these details in the descriptions of the sequel.

2.3 Setting

We consider a set $P = \{p_1, \ldots, p_n\}$ of n players that are connected via a complete synchronous network of pairwise secure channels. We assume a PKI to be set up among the players.

A given PKI is *consistent* if every player p_i ($i \in \{1, \ldots, n\}$) has a secret-key/public-key pair (SK_i,PK_i) which was chosen by p_i with respect to the key-generation algorithm of a digital signature scheme and, additionally, that each respective public key PK_i is known to all players as p_i's public key to be exclusively used for the verification of p_i's signatures. Asserting that, with respect to every signer p_i, each player holds the same public key PK_i guarantees that p_i's signatures can be transferred between the players without losing conclusiveness. In our model, the PKI may or may not be consistent.[7]

We assume the existence of an active threshold adversary to corrupt some of the players. The adversary may or may not be able to forge signatures. Furthermore, for our protocols, the adversary is assumed to be adaptive (but non-mobile). In contrast, our proofs of optimality even hold with respect to a static adversary. As we have three bounds with respect to player corruption, t_p, t_σ, and T, we will make a stringent distinction between these bounds and the actual number of players corrupted at the end of the protocol, which we will denote by f.[8]

2.4 Protocol notation

Protocols are specified with respect to a player set P and stated with respect to the local view of player p_i, meaning that all players $p_i \in P$ execute this code in parallel with respect to their own identity i.

With respect to pairwise communication we also consider reflexive channels among the players for simplicity. Thus, when a player p_i sends a value to each player then p_i also receives a copy himself.

For simplicity, in our protocol notation, we do not explicitly state how to handle values received from corrupted players that are outside the specified domain. Such a value is always implicitly assumed to be replaced by a default value or by any arbitrary value inside the specified domain.

3 Generic Broadcast Simulation for $t < n/2$

Our broadcast simulations are based on the "phase king" protocol in [BGP89]. In [FM00], it was observed that any protocol for "weak broadcast" is sufficient in order to achieve broadcast secure against a faulty minority — as secure as the given protocol for weak broadcast. Since all of our tight bounds imply that strictly less than half of all players are corrupted we thus only need to give respective protocols for weak broadcast.

Weak broadcast (as called in [FM00]) was originally introduced in [Dol82] under the name *crusader agreement*. Weak broadcast is the same as broadcast for the case that the sender is correct but, if the sender is corrupted, then some players might end up with the "invalidity symbol" $\perp$ — but still, it guarantees that no two correct players end up with two different values in $\{0, 1\}$.

[7] In case of unconditional pseudo-signatures the situation is slightly different since, instead of the same public key PK_i, each player p_j holds a different "public key" PK_{ij} (which is in fact secret).

[8] As the adversary is adaptive, the number might increase during the execution of the protocol, and reach its maximum at the end.

Definition 2 (Weak broadcast). *A protocol where one player p_s has an input $x_s \in \{0,1\}$ and every player p_i computes an output $y_i \in \{0,1,\perp\}$ achieves weak broadcast if it satisfies the following conditions:*

- VALIDITY: *If p_s is correct then every correct player p_i computes output $y_i = x_s$.*
- CONSISTENCY: *If player p_i is correct and computes $y_i \in \{0,1\}$ then every correct player p_j computes $y_j \in \{y_i, \perp\}$.*

In Appendix A, we describe a reduction from broadcast to weak broadcast that is simpler and more efficient than the one in [FM00], yielding the following theorem.

Theorem 1. *If at most $t < n/2$ players are corrupted then efficient achievability of weak broadcast implies efficient achievability of broadcast.*

4 Tight Bounds

We now demonstrate the tightness of the bound given in Bound (1).

4.1 Efficient Protocol

We first give an efficient protocol for broadcast and then show how to plug it into the MPC protocol in [CDD$^+$99] in order to get out final protocol for efficient hybrid MPC.

Broadcast. For the constructive part, according to Theorem 1, it is sufficient to give a construction for weak broadcast. The following protocol is designed for any selection of thresholds t_p, t_σ, and T, ($t_p \leq t_\sigma \leq T$), satisfying Bound (1).

Let x_s be p_s's input value, and let $\sigma_s(x_s)$ be a signature by p_s on the value x_s. Furthermore, let V be the signature verification algorithm with respect to the underlying signature scheme computing $V(x, \sigma, \mathrm{PK}) = 1$ if σ is a valid signature on x with respect to public key PK, and $V(x, \sigma, \mathrm{PK}) = 0$ otherwise. Let PK_i^s be player p_i's version of p_s's public key. We use $V_i^s(x, \sigma)$ as a short cut for $V(x, \sigma, \mathrm{PK}_i^s)$. With respect to player p_i, we say that a given signature σ is *valid* if it is valid with respect to p_i's view, i.e., $V_i^s(x, \sigma) = 1$. In particular, a valid signature with respect to player p_i's view might in fact not have been issued by the respective signer.

The protocol works as follows. The sender p_s signs his input value and sends his input together with its signature to every other player: $(x_s, \sigma_s(x_s))$. Every player except for the sender now redistributes this information to everybody (but without signing this new message himself). Now, every player received n values, one from every player. Each player p_i now decides on the outcome of the protocol:

- Let x_i^s be the bit he directly received from the sender. If the bit x_i^s is received from at least $n - t_p$ different players overall then he computes output $y_i = x_i$.

– Otherwise, if he received the bit x_i^s together with a valid signature by p_s from the sender and at least $n - t_\sigma$ different players overall then he decides on $y_i = x_i$.
– Otherwise, if he received the bit x_i together with a valid signature by p_s from the sender and at least $n - T$ different players overall — but no single correct signature by p_s for bit $1 - x_i$ — then he decides on $y_i = x_i$.
– Otherwise, he decides on $y_i = \bot$.

Protocol 1 `WeakBroadcast`$_{p_s}$ (P, x_s)

```
1. if $i = s$ then SendToAll($x_s, \sigma_s(x_s)$) fi;                    Receive($x_i^s, \sigma_i^s$);
2. if $i \neq s$ then SendToAll($x_i^s, \sigma_i^s$) fi;           $\forall j \neq s$: Receive($x_i^j, \sigma_i^j$);
3. $U_i^0 := \{p_j \in P | x_i^j = 0\}$; $U_i^1 := \{p_j \in P | x_i^j = 1\}$;
4. $S_i^0 := \{p_j \in P | x_i^j = 0 \ \wedge \ V_i^s(0, \sigma_i^j) = 1\}$;
   $S_i^1 := \{p_j \in P | x_i^j = 1 \ \wedge \ V_i^s(1, \sigma_i^j) = 1\}$;
5. if $\left|U_i^{x_i^s}\right| \geq n - t_p$ then $y_i := x_i^s$                                      (A)
   elseif $p_s \in S_i^{x_i^s} \ \wedge \ \left|S_i^{x_i^s}\right| \geq n - t_\sigma$ then $y_i := x_i^s$                 (B)
   elseif $p_s \in S_i^{x_i^s} \ \wedge \ \left|S_i^{x_i^s}\right| \geq n - T \ \wedge \ S_i^{1-x_i^s} = \emptyset$ then $y_i := x_i^s$   (C)
   else $y_i := \bot$ fi;                                                  (D)
6. return $y_i$
```

Lemma 1 (Weak Broadcast). *Protocol 1 among the players $P = \{p_1, \ldots, p_n\}$ achieves efficient weak broadcast with sender $p_s \in P$ if $2T + t_p < n$ and $T + 2t_\sigma < n$.*

Proof. We show that the validity and consistency properties are satisfied. For this, let f be the number of corrupted players at the end of the protocol. Efficiency is obvious.

VALIDITY: Suppose that the sender p_s is correct. Hence, every correct player p_i receives the sender's input x_s during Step 1 of the protocol, $x_i^s = x_s$, and a signature σ_i^s.

If $f \leq t_p$ then every correct player p_i receives the value x_s from at least $n - t_p$ different players (including the sender) during Steps 1 and 2. Hence, $|U_i^{x_s}| \geq n - t_p$, and p_i computes $y_i = x_s$ according to Condition (A) in Step 5.

If $t_p < f \leq t_\sigma$ then every correct player p_i receives the value x_s together with a valid signature by p_s (note that the PKI is consistent in this case) from at least $n - t_\sigma$ different players (including the sender) during Steps 1 and 2. Hence, $|S_i^{x_s}| \geq n - t_\sigma$, and p_i computes $y_i = x_s$ according to Conditions (A) or (B) in Step 5.

If $t_\sigma < f \leq T$ then every correct player p_i receives the value x_s together with a valid signature by p_s from at least $n - T$ different players (including the sender) during Steps 1 and 2. Hence, $|S_i^{x_s}| \geq n - T$ and $S_i^{1-x_s} = \emptyset$ since the adversary cannot forge signatures in this case. Hence, p_i computes $y_i = x_s$ according to Conditions (A), (B), or (C).

CONSISTENCY: Suppose that some correct player p_i computes output $y_i \neq \bot$. We have to show that hence, every correct player p_j computes an output $y_j \in \{y_i, \bot\}$.

Suppose first, that p_i decides according to Condition (A) in Step 5, i.e., $|U_i^{y_i}| \geq n - t_p$. For p_j this implies that $|U_j^{y_i}| \geq |U_i^{y_i}| - T \geq n - t_p - T > T$ and hence that $|S_j^{1-y_i}| \leq |U_j^{1-y_i}| < n - T$ and thus that p_j cannot compute $y_j = 1 - y_i$, neither according to Conditions (A), (B), nor (C).

Second, suppose that p_i decides according to Condition (B) in Step 5, i.e., $|S_i^{y_i}| \geq n - t_\sigma$. It remains to show that p_j does not decide on $y_j = 1 - y_i$ according to Conditions (B) or (C) (the rest is out-ruled by the last paragraph). For p_j the assumption implies that

$$|U_j^{y_i}| \geq |U_i^{y_i}| - f \geq n - t_\sigma - f \geq \begin{cases} n - 2t_\sigma > T & \text{, if } f \leq t_\sigma \text{ ,} \\ n - t_\sigma - T > t_\sigma & \text{, if } f \leq T \text{ .} \end{cases}$$

Now, if $f \leq t_\sigma$ then $|S_j^{1-y_i}| \leq |U_j^{1-y_i}| < n - T$, and p_j cannot compute $y_j = 1 - y_i$ according to Conditions (B) or (C). If $t_\sigma < f \leq T$ then $|U_j^{1-y_i}| < n - t_\sigma$, and $S_j^{y_i} \neq \emptyset$ (since the PKI is consistent and p_i holds and redistributes a valid signature on y_i), and thus p_j still cannot compute $y_j = 1 - y_i$ according to Conditions (B) or (C).

Third, suppose that p_i decides according to Condition (C) in Step 5, i.e., $|S_i^{y_i}| \geq n - T$ and $S_i^{1-y_i} = \emptyset$. It remains to show that p_j cannot decide on $y_j = 1 - y_i$ according to Condition (C). Now, $f \leq t_p$ implies $|U_j^{1-y_i}| < n - T$ (since $|U_j^{y_i}| \geq n - T - t_p > T$), and $f > t_p$ implies $S_j^{y_i} \neq \emptyset$ (since the PKI is consistent). Finally, both implications rule out that p_j computes $y_j = 1 - y_i$ according to Condition (C). □

Multi-party computation. The MPC protocol in [CDD$^+$99] unconditionally tolerates an (adaptive) adversary that corrupts up to $t < n/2$ players — but assuming broadcast channels to be available.

Theorem 2. *Hybrid MPC is efficiently achievable if* $2T + t_p < n$ *and* $T + 2t_\sigma < n$.

Proof. Efficient achievability of hybrid broadcast for $2T + t_p < n$ and $T + 2t_\sigma < n$ follows from Lemma 1 and Theorem 1. We can now simulate each invocation of a broadcast channel in [CDD$^+$99] with an instance of such a hybrid broadcast protocol. Since Bound (1) implies $2T < n$, we have that $t_p \leq t_\sigma \leq T < n/2$. Thus, an adversary that is tolerated in the broadcast protocol is automatically tolerated in the MPC protocol. □

4.2 Tightness

We now show that Bound (1) is tight. We do this in three steps. First, we show that hybrid broadcast is impossible if $t_p > 0$, $t_\sigma = 0$, and $2T + t_p \geq n$. Second, we show that hybrid broadcast is impossible if $t_p = 0$, $t_\sigma > 0$, and $T + 2t_\sigma \geq n$. Third, we use the fact that MPC is impossible whenever $2T \geq n$ [Cle86].

Impossibility of broadcast for $\mathbf{t_p + 2T \geq n}$ when $\mathbf{t_p > 0}$. The proof proceeds along the lines of the proof in [FLM86] that unconditional broadcast for $t \geq n/3$ is impossible. The idea is to assume any protocol among n players that (possibly) achieves broadcast for $n \leq 2T + t_p$ ($t_p > 0$, $t_\sigma = 0$) and to use it to build a different distributed system whose behavior demonstrates that the original protocol must be insecure. It is important to note that this new system is not required to achieve broadcast. It is simply a distributed system whose behavior is determined by the protocol or, more precisely, by the corresponding local programs of the involved players. Also, it is assumed that no adversary is present in this system. Rather, with respect to some of the players, the way the new system is composed simulates an admissible adversary with respect to these players in the original system. Thus, with respect to these players, all conditions of broadcast are required to be satisfied among them even in this new system. Finally, it is shown that all of these players' respective conditions cannot be satisfied simultaneously and thus that the protocol cannot achieve broadcast.

Building the new system. Assume any protocol Ψ for a player set P with sender p_0 and $|P| = n \geq 3$ that tolerates $2T + t_p \geq n$ (with $t_p > 0$).

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs sharing a consistent PKI where player p_i's secret-key/public-key pair is $(\mathrm{SK}_i, \mathrm{PK}_i)$ and player p_j's copy of the respective public key is PK_{ij}. Since $0 < t_p \leq T$, it is possible to partition the processors into three sets, $\Pi_0 \dot\cup \Pi_1 \dot\cup \Pi_2 = \Pi$, such that $1 \leq |\Pi_0| \leq t_p$, $1 \leq |\Pi_1| \leq T$, and $1 \leq |\Pi_2| \leq T$.

For each $\pi_i \in \Pi_0$, let π'_i be an identical copy of processor π_i. Let the number i denote the *type* of any processor π_i (or π'_i, respectively). Furthermore, let $\Pi'_0 = \{\pi'_i | \pi_i \in \Pi_0\}$ form an identical copy of set Π_0. For all $\pi_i \in \Pi'_0$, generate a new secret-key/public-key pair $(\mathrm{SK}'_i, \mathrm{PK}'_i)$ and overwrite π_i's own secret key $\mathrm{SK}_i := \mathrm{SK}'_i$. Additionally, for all $\pi_j \in \Pi_2 \cup \Pi'_0$, overwrite π_j's copy of π_i's public key: $\mathrm{PK}_{ij} := \mathrm{PK}'_i$ (and $\mathrm{PK}_i := \mathrm{PK}'_i$). See Figure 3.

Instead of connecting the original processors as required for broadcast, we build a network involving all processors in $\Pi_0 \cup \Pi_1 \cup \Pi_2 \cup \Pi'_0$ with their pairwise communication channels connected in a way such that each processor π_i (or π'_i) communicates with at most one processor of each type $j \in \{1, \ldots, n\} \setminus \{i\}$.

Consider Figure 3. Exactly all pairs in $(\Pi_0 \cup \Pi_1) \times (\Pi_0 \cup \Pi_1)$, $(\Pi_1 \cup \Pi_2) \times (\Pi_1 \cup \Pi_2)$, and $(\Pi_2 \cup \Pi'_0) \times (\Pi_2 \cup \Pi'_0)$ are connected by pairwise channels. There are no connections between the sets Π_0 and Π_2, and no connections between the sets Π_1 and Π'_0. Messages that originally would have been sent from a processor in Π_0 to a processor in Π_2 are discarded. Messages that originally would have been sent from a processor in Π_2 to a processor in Π_0 are delivered to the corresponding processor in Π'_0. Messages sent from a processor in Π'_0 to a processor in Π_1 are discarded.

We now show that for the sets $\Pi_0 \cup \Pi_1$, $\Pi_1 \cup \Pi_2$, and $\Pi_2 \cup \Pi'_0$, and for inputs $x_0 = 0$ and $x'_0 = 1$, each set's joint view is indistinguishable from its view in the original setting for an adversary corrupting the remaining processors in

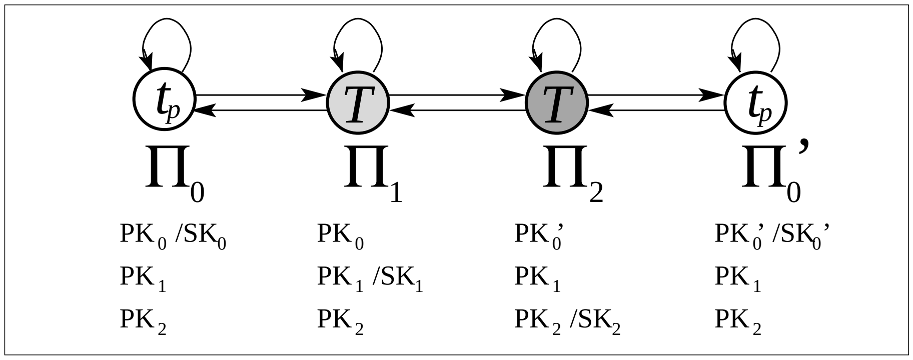

Fig. 3. *Rearrangement of processors in proof of Theorem 3.*

an admissible way, and possibly have made for a bad PKI if it corrupts at most $f \leq t_p$ processors.

Lemma 2. *If the input of π_0 is $x_0 = 0$ then the joint view of the processors in $\Pi_0 \cup \Pi_1$ is indistinguishable from their view in the original system when the adversary corrupts the processors in Π_2 in an admissible way.*

Proof. By corrupting all processors in Π_2 in the original system the adversary simulates all processors in $\Pi_2 \cup \Pi'_0$ of the new system. For all $\pi_i \in \Pi'_0$ it generates a new secret-key/public-key pair $(\mathrm{SK}'_i, \mathrm{PK}'_i)$ and overwrites π_i's own secret key $\mathrm{SK}_i := \mathrm{SK}'_i$ and, for all $\pi_j \in \Pi_2 \cup \Pi'_0$, overwrites π_j's copy of π_i's public key: $\mathrm{PK}_{ij} := \mathrm{PK}'_i$ (and $\mathrm{PK}_i := \mathrm{PK}'_i$). Initially, the adversary overwrites input $x'_0 := 1$. The PKI among the processors in $\Pi_0 \cup \Pi_1$ is still fully consistent and thus the joint view of the processors in $\Pi_0 \cup \Pi_1$ in the original system is exactly the same as their view in the new system. □

Lemma 3. *If the input of π'_0 is $x'_0 = 1$ then the joint view of the processors in $\Pi_2 \cup \Pi'_0$ is indistinguishable from their view in the original system when the adversary corrupts the processors in Π_1 in an admissible way.*

Proof. By symmetry, this case follows from Lemma 2.[9] □

Lemma 4. *The joint view of the processors in $\Pi_1 \cup \Pi_2$ is indistinguishable from their view in the original system when the adversary corrupts the processors in Π_0 in an admissible way.*

Proof. Since $|\Pi_0| \leq t_p$ the adversary can have previously made the PKI inconsistent by generating and respectively distributing the key pairs $(\mathrm{SK}'_i, \mathrm{PK}'_i)$ for

[9] The only difference in this case is that Π'_0 takes the role of the original set and Π_0 the role of its copy. Accordingly, the initial key pairs are $(\mathrm{SK}'_i, \mathrm{PK}'_i)$, the pairs $(\mathrm{SK}_i, \mathrm{PK}_i)$ are newly generated by the adversary, and $x_0 := 0$ is overwritten.

all $\pi_i \in \Pi'_0$ (according to Figure 3). By corrupting all processors in Π_0 in the original system the adversary can now simulate all processors in $\Pi_0 \cup \Pi'_0$ of the new system whereas, initially, it overwrites $x_0 := 0$ and $x'_0 := 1$. Thus the joint view of the processors in $\Pi_1 \cup \Pi_2$ in the original system is exactly the same as their view in the new system. □

Theorem 3. *If $2T + t_p \geq n$ and $t_p > 0$ then there exists no hybrid broadcast protocol. In particular, for every protocol there exists a sender input $x_0 \in \{0, 1\}$ such that a computationally bounded adversary can make the protocol fail with some non-negligible probability — by either corrupting T players, or by corrupting t_p players and additionally having made for an inconsistent PKI.*

Proof. Assume that $x_0 = 0$ and $x'_0 = 1$. Then, by Lemmas 2, 3, and 4, each mentioned set's joint view in the new system is indistinguishable from their view in the original system. However, for each run of the new system, either validity is violated among the processors in $S_{01} = \Pi_0 \cup \Pi_1$ or $S_{20'} = \Pi_2 \cup \Pi'_0$, or consistency is violated among the processors in $S_{12} = \Pi_1 \cup \Pi_2$.

Thus there is a sender input ($x_0 = 0$ or $x'_0 = 1$) such that the adversary can make the protocol fail with non-negligible probability by uniformly randomly choosing a processor set Π_i and corrupting the respective processors correspondingly. □

Impossibility of broadcast for $\mathbf{2t_\sigma + T \geq n}$ when $\mathbf{t_\sigma > 0}$. The proof of this case is very similar to the proof of Theorem 3.

Theorem 4. *If $T+2t_\sigma \geq n$ and $t_\sigma > 0$ then there exists no hybrid broadcast protocol. In particular, for every protocol there exists a sender input $x_0 \in \{0, 1\}$ such that the adversary can make the protocol fail with some non-negligible probability — by either corrupting T players, or by corrupting t_σ players and additionally being able to forge signatures with non-negligible probability.*

Proof. Assume any protocol Ψ for a player set P with sender p_0 and $|P| = n \geq 3$ that tolerates $2T + t_\sigma \geq n$ (with $t_\sigma > 0$).

Let $\Pi = \{\pi_0, \dots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs. Since $0 < t_\sigma \leq T$, it is possible to partition the processors into three sets, $\Pi_0 \dot\cup \Pi_1 \dot\cup \Pi_2 = \Pi$, such that $1 \leq |\Pi_0| \leq T$, $1 \leq |\Pi_1| \leq t_\sigma$, and $1 \leq |\Pi_2| \leq t_\sigma$.

For each $\pi_i \in \Pi_0$, let π'_i be an identical copy of processor π_i and, as in the proof of Theorem 3, let $\Pi'_0 = \{\pi'_i | \pi_i \in \Pi_0\}$ form an identical copy of set Π_0.

Consider Figure 4. Exactly all pairs in $(\Pi_0 \cup \Pi_1) \times (\Pi_0 \cup \Pi_1)$, $(\Pi_1 \cup \Pi_2) \times (\Pi_1 \cup \Pi_2)$, and $(\Pi_2 \cup \Pi'_0) \times (\Pi_2 \cup \Pi'_0)$ are connected by pairwise channels.

Again, we show that for the sets $\Pi_0 \cup \Pi_1$, $\Pi_1 \cup \Pi_2$, and $\Pi_2 \cup \Pi'_0$, and for inputs $x_0 = 0$ and $x'_0 = 1$, each set's joint view is indistinguishable from its view in the original setting.

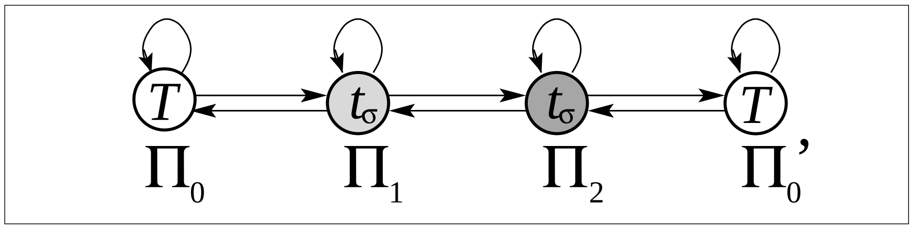

Fig. 4. *Rearrangement of processors in proof of Theorem 3.*

Joint view of $\Pi_0 \cup \Pi_1$ with $x_0 = 0$. By corrupting all processors in Π_2 in the original system the adversary simulates all processors in $\Pi_2 \cup \Pi_0'$ of the new system. Since $|\Pi_2| \leq t_\sigma$, the adversary can forge all signatures by processors in Π_0' required for the simulation. Initially, the adversary overwrites input $x_0' := 1$. Thus the joint view of the processors in $\Pi_0 \cup \Pi_1$ in the original system is exactly the same as their view in the new system.

Joint view of $\Pi_2 \cup \Pi_0'$ with $x_0' = 1$. By symmetry, this case follows from the above paragraph.

Joint view of $\Pi_1 \cup \Pi_2$. By corrupting all processors in Π_0 in the original system the adversary can simulate all processors in $\Pi_0 \cup \Pi_0'$ of the new system whereas, initially, it overwrites $x_0 := 0$ and $x_0' := 1$. Note that, by corrupting the processors in Π_0, the adversary gains access to all corresponding secret keys and thus is not required to forge any signatures for the simulation. Thus the joint view of the processors in $\Pi_1 \cup \Pi_2$ in the original system is exactly the same as their view in the new system.

The theorem now follows along the lines of the proof of Theorem 3. □

Multi-party computation. In order to complete our tightness argument, we require the following proposition.

Proposition 1 ([Cle86]). *There is no protocol for MPC secure against $T \geq n/2$ actively corrupted players. In particular, fairness cannot be guaranteed.*

Theorem 5. *Hybrid MPC is impossible if either $2T + t_p \geq n$ or $T + 2t_\sigma \geq n$.*

Proof. The theorem follows from Theorems 3 and 4, and Proposition 1. □

5 Conclusion and Open Problems

We can now conclude tight bounds for the achievability of hybrid MPC with respect to thresholds t_p, t_σ, and T.

Theorem 6. *Hybrid MPC is (efficiently) achievable if and only if* $2T + t_p < n$ *and* $T + 2t_\sigma < n$.

Proof. The theorem follows from Theorems 2 and 5. □

In particular, assuming the PKI to be consistent in any case (as in the alternative model for [GMW87] assuming a PKI instead of broadcast) we can drop parameter t_p and immediately get the tight bound $2T < n \ \wedge\ T + 2t_\sigma < n$. This means that, with respect to this model, computational security against $f \leq T < n/2$ corrupted players can be combined with unconditional security against $f \leq t_\sigma$ corrupted players.

The characterization given in Theorem 6 is tight with respect to fully secure, robust MPC. However, as mentioned in the introduction, non-robust MPC is also possible in presence of a corrupted majority. Thus, for the case $t_p = 0$, it remains an open question whether hybrid non-robust MPC can be achieved for any $T + 2t_\sigma < n$.

References

[Bea89] Donald Beaver. Multiparty protocols tolerating half faulty processors. In *Advances in Cryptology: CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 560–572. Springer-Verlag, 1989.

[Bea91] Donald Beaver. Foundations of secure interactive computation. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer-Verlag, 1991.

[BG89] Donald Beaver and Shafi Goldwasser. Multiparty computation with faulty majority. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS '89)*, pages 468–473, 1989.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS '89)*, pages 410–415, 1989.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 1–10. Springer-Verlag, 1988.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 11–19. ACM Press, 1988.

[CDD$^+$99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, 1999.

[Cha89] David Chaum. The spymasters double-agent problem. In *Advances in Cryptology: CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 591–602. Springer-Verlag, 1989.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *ACM Symposium on Theory of Computing (STOC '86)*, pages 364–369, Baltimore, USA, May 1986. ACM Press.

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

[Dol82] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[DS82] Danny Dolev and H. Raymond Strong. Polynomial algorithms for multiple processor agreement. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC '82)*, pages 401–407, 1982.

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

[FGH$^+$02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. Detectable Byzantine agreement secure against faulty majorities. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02)*, pages 118–126, 2002.

[FHHW03] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT '03*, Lecture Notes in Computer Science. Springer-Verlag, May 2003.

[FLM86] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1:26–39, 1986.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, August 1997.

[FM00] Matthias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pages 494–503, 2000.

[GHY87] Zvi Galil, Stuart Haber, and Moti Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Advances in Cryptology: CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 135–155. Springer-Verlag, 1987.

[GL90] Shafi Goldwasser and Leonid Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 11–15. Springer-Verlag, 1990.

[GL02] Shafi Goldwasser and Yehuda Lindell. Secure computation without agreement. 16th International Symposium on Distributed Computing (DISC '02). Preliminary version on http://www.research.ibm.com/people/l/lindell, 2002.

[GLR95] Li Gong, Patrick Lincoln, and John Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. In *Proceedings of the 5th Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 79–90, 1995.

[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. ACM Press, 1987.

[Gol01] Oded Goldreich. Secure multi-party computation, working draft, version 1.3, June 2001.

[LLR02] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated Byzantine agreement. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 514–523. ACM Press, 2002.

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

[PW96] Birgit Pfitzmann and Michael Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t >= n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 73–85, 1989.

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

[WP89] Michael Waidner and Birgit Pfitzmann. Unconditional sender and recipient untraceability in spite of active attacks — some remarks. Technical Report 5/89, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1989.

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

A Reducing Broadcast to Weak Broadcast

In this section we describe how to efficiently reduce broadcast to weak broadcast in a way that is more direct than in [FM00]. Given that at most $t < n/2$ players are corrupted the resulting protocol for broadcast is as secure as the given protocol for weak broadcast.

In a first step, weak broadcast is transformed into a protocol for graded consensus (Section A.1), the "consensus variant" of graded broadcast introduced by Feldman and Micali in [FM97]; and finally, graded consensus is transformed into broadcast (Section A.2).

A.1 Graded Consensus

In graded consensus, every player has an input x and receives two outputs, a value $y \in \{0, 1\}$ and a grade $g \in \{0, 1\}$. If all correct players start with the same value x then all players output $y = x$ and $g = 1$. Additionally, if any correct player ends up with grade $g = 1$ then all correct players output the same value y, i.e., computing $g = 1$ means "detecting agreement."

Definition 3 (Graded Consensus). *A protocol where every player p_i has an input $x_i \in \{0, 1\}$ and computes two output values $y_i, g_i \in \{0, 1\}$ achieves graded consensus if it satisfies the following conditions:*

- Validity: *If all correct players have the same input value x then every correct player p_i computes $y_i = x$ and $g_i = 1$.*
- Consistency: *If any correct player p_i computes $g_i = 1$ then every correct player p_j computes $y_j = y_i$.*

The following protocol for graded consensus basically consists of two consecutive rounds wherein each player weak-broadcasts a value. Note that, in Step 4 of the protocol, the domain of weak broadcast is ternary, namely $\{0, 1, \perp\}$. Following the restriction to focus on protocols with binary domains we can simply interpret such a protocol as being simulated by two parallel invocations of binary weak broadcast.

Protocol 2 `GradedConsensus` (P, x_i)

```
1. $\forall j \in \{1, \ldots, n\} : x_i^j :=$ WeakBroadcast$_{p_j}$ $(P, x_j)$;
2. $S_i^0 := \{j \in \{1, \ldots, n\} | x_i^j = 0\}$; $S_i^1 := \{j \in \{1, \ldots, n\} | x_i^j = 1\}$;
3. if $|S_i^{x_i}| \geq n - t$ then $z_i := x_i$ else $z_i := \perp$ fi;
4. $\forall j \in \{1, \ldots, n\} : z_i^j :=$ WeakBroadcast$_{p_j}$ $(P, z_j)$;
5. $T_i^0 := \{j \in \{1, \ldots, n\} | z_i^j = 0\}$; $T_i^1 := \{j \in \{1, \ldots, n\} | z_i^j = 1\}$;
6. if $|T_i^0| > |T_i^1|$ then $y_i := 0$ else $y_i := 1$ fi;
7. if $|T_i^{y_i}| \geq n - t$ then $g_i := 1$ else $g_i = 0$ fi;
8. return $(y_i, g_i)$
```

Lemma 5 (Graded Consensus). *If Protocol* `WeakBroadcast` *achieves weak broadcast then Protocol 2 achieves graded consensus secure against $t < n/2$ corrupted players.*

Proof.
Validity: If all correct players hold the same value x at the beginning of the protocol then, by the validity property of weak broadcast, $|S_i^{x_i}| \geq n - t > t$ for every correct player p_i and thus $z_i = x_i = x$. Finally, $|T_i^x| \geq n - t > t$, $|T_i^{1-x}| < n - t$, and $y_i = x$ and $g_i = 1$.

Consistency: Note that every correct player p_i that does not compute $z_i = \perp$ (in Step 3) holds the same value $z_i = z$: By the validity property of weak broadcast, $|S_i^{x_i}| \geq n - t$ implies that $|S_j^{1-x_i}| \leq t < n - t$.

Now, let p_i and p_j be two correct players and suppose that p_i decides on $y_i = y \in \{0, 1\}$ and $g_i = 1$. We have to show that $y_j = y$.

From $g_i = 1$ it follows that $|T_i^y| \geq n - t > t$ and thus that at least one correct player p_k must have sent $z_k = y$ during Step 4, and with the above remark, that no correct player p_k can have sent $z_k = 1 - y$ during Step 4.

Let ℓ be the number of corrupted players who distributed value y during Step 4. Now, $|T_i^y| \geq n - t$ implies $|T_j^y| \geq n - t - \ell > t - \ell$ and $|T_j^{1-y}| \leq t - \ell$ since only the remaining $t - \ell$ corrupted players can have sent value y during Step 4. Thus, p_j computes $y_j := y = y_i$. □

A.2 Broadcast

For simplicity, without loss of generality, assume $s = 1$, i.e., that p_1 is the sender of the broadcast.

Protocol 3 $\texttt{Broadcast}_{p_1}(P, x_1)$

```
1. $p_1$: Send $x_1$;                    Receive($y_i$)
2. $p_i$: for $k = 2$ to $t+1$ do
3.      $(y_i, g_i) :=$ GradedConsensus $(P, y_i)$;
4.      $p_k$: Send $y_k$;               Receive($y_i^k$)
5.      if $g_i = 0$ then $y_i := y_i^k$ fi
6. od; return $y_i$
```

Lemma 6. *Suppose that Protocol* GradedConsensus *achieves graded consensus. If in Protocol 3, for some $k \in \{2, \ldots, t+1\}$, every correct player p_i holds the same value $y_i = b$ at the beginning of Step 3 then $y_i = b$ holds at the end of the protocol.*

Proof. Suppose that $y_i = b \in \{0, 1\}$ for every correct player p_i before Step 3 (for some k). Because of the validity property of graded consensus, after Step 3, (for k), every correct player p_i holds $y_i = b$ and $g_i = 1$, and thus ignores Step 5, (for k). Thus, by induction, every correct player p_i ends the protocol with $y_i = b$. □

Lemma 7 (Broadcast). *If Protocol* GradedConsensus *achieves graded consensus then Protocol 3 achieves broadcast with sender p_1 (for any $t < n$).*

Proof. We show that the validity and consistency properties of broadcast are satisfied.

VALIDITY: Suppose the sender p_1 to be correct with input $x_s = b$. Hence, every correct player p_i holds value $y_i = b$ before Step 3 for $k = 2$. And by Lemma 6, every correct player p_i ends the protocol with $y_i = b$.

CONSISTENCY: If the sender is correct then consistency is implied by the validity property. Assume now that p_1 is corrupted. Hence there is a correct player p_k ($k \in \{2, \ldots, t+1\}$). We now argue that, for such a k where p_k is correct, every correct player p_i holds the same value y_i after Step 5. Then, together with Lemma 6, the consistency property follows.

First, suppose that every correct player p_i holds $g_i = 0$ after Step 3. Then all of them adopt p_k's value, $y_i = y_i^k$, and consistency follows from Lemma 6. Suppose now, that any correct player p_i holds $g_i = 1$ after Step 3. Then, by the consistency property of graded consensus, p_k and every other correct player p_j hold $y_k = y_i = y_j$, and consistency follows from Lemma 6. □

Theorem 1. *If at most $t < n/2$ players are corrupted then efficient achievability of weak broadcast implies efficient achievability of broadcast.*

Proof. Since the given construction for broadcast involves a polynomial number of invocations of weak broadcast $(2n(t+1))$, the theorem follows directly from Lemma 7. □