Adaptively Secure Broadcast

Martin Hirt and Vassilis Zikas

Department of Computer Science, ETH Zurich
{hirt,vzikas}@inf.ethz.ch

Abstract. A broadcast protocol allows a sender to distribute a message through a point-to-point network to a set of parties, such that (i) all parties receive the same message, even if the sender is corrupted, and (ii) this is the sender's message, if he is honest.

Broadcast protocols satisfying these properties are known to exist if and only if $t < n/3$, where n denotes the total number of parties, and t denotes the maximal number of corruptions. When a setup allowing signatures is available to the parties, then such protocols exist even for $t < n$.

Broadcast is the probably most fundamental primitive in distributed cryptography, and is used in almost any cryptographic (multi-party) protocol. However, a broadcast protocol "only" satisfying the above properties might be insecure when being used in the context of another protocol. In order to be safely usable within other protocols, a broadcast protocol must satisfy a simulation-based security notion, which is secure under composition.

In this work, we show that most broadcast protocols in the literature do not satisfy a (natural) simulation-based security notion. We do not know of any broadcast protocol which could be securely invoked in a multi-party computation protocol in the secure-channels model. The problem is that existing protocols for broadcast do not preserve the secrecy of the message while being broadcasted, and in particular allow the adversary to corrupt the sender (and change the message), depending on the message being broadcasted. For example, when every party should broadcast a random bit, the adversary could corrupt those parties that want to broadcast 0, and make them broadcast 1.

More concretely, we show that simulatable broadcast in a model with secure channels is possible if and only if $t < n/3$, respectively $t \leq n/2$ when a signature setup is available. The positive results are proven by constructing secure broadcast protocols.

1 Introduction

1.1 Broadcast

Broadcast, as defined in the literature, allows a sender to securely distribute a message among a set of players, where security means that (i) all honest players eventually output the same message, no matter what the sender does (consistency), and (ii) this output is the sender's message, if he is honest (validity).

Broadcast was introduced by Pease, Shostak, and Lamport [PSL80]. The first efficient solution was given by Dolev and Strong [DS82]. Further research concentrated on lower bounds [FL82, FLM86], efficiency [BGP89, CW92, BDDS92], round complexities [MW88, FM89, GM98], and feasibility with statistical security [BPW91, PW92, PW96].

Broadcast is one of the most fundamental primitives in distributed cryptography. It is used in almost any task that involves multiple players, like, e.g., voting, bidding, secure function evaluation, threshold key generation, multi-party computation, etc — just to mention a few. The security of these protocols inherently relies on the security of the underlying broadcast protocol.

1.2 Simulation-based Security

Until recently, the security of most cryptographic protocols in the literature was argued by enumerating a number of properties that a good protocol should have, and proving that the new protocol satisfies each of them. This approach is very risky, as firstly, history teaches that there is always at least one desired property that was forgotten, and secondly, as illustrated by Canetti [Can00], a protocol can well satisfy all desired properties, and still behave unexpectedly in some context. In particular, property-based protocols do not compose securely.

A more accurate security notion is the simulation-based approach: Here, the desired behavior is defined as an ideal functionality, and a protocol is secure if the adversary has no advantage in attacking the protocol, as compared to attacking the functionality. Most interestingly, many simulation-based security notions allow for composition theorems [Can00, Can03, BPW03, DM00], which essentially state that a secure protocol remains secure when it invokes other secure protocols.

In the context of broadcast, the common property-based definition requires consistency and validity (formally, also termination must be required). The (natural) ideal functionality for broadcast is the following: The sender sends the message to the functionality, who forwards the received message to all players. Although broadcast protocols in the literature are argued secure only with respect to the property-based definition, it is commonly believed that they securely realize this functionality in a simulation-based notion. In fact, most (if not all) higher-level protocols in the literature (like VSS [CGMA85], MPC [GMW87, BGW88, RB89], etc.) are argued secure in a hybrid model with access to an ideal broadcast functionality, and the hope is that these protocols remain secure when the calls to the broadcast functionality are replaced by one of the property-based protocol implementations.

1.3 Our Contributions

We show that the property-based definition of broadcast does not imply simulation-based security with the natural functionality, not even in a stand-alone setting, not even in the secure-channels model with perfect security.

Furthermore, we give broadcast protocols with simulation-based security in the secure-channels model that tolerate $t < n/3$ (with perfect security, without further assumptions), respectively $t \leq n/2$ (with statistical resp. cryptographic security, when a secure signature functionality is available). Both bounds are tight. We stress that in the secure channels model, no protocol exists that securely realizes the natural broadcast functionality when $t > n/2$ (although property-based security is possible for $t < n$ [DS82, PW92]).

The negative result can easily be illustrated by an example. Consider the following broadcast protocol: First, the sender transmits the message to all players. Then, the players run a perfectly secure consensus protocol on the received values [BGP89]. The resulting broadcast protocol satisfies the consistency and validity property with perfect security when $t < n/3$. However, it is **not** a secure realization of the above natural functionality for broadcast. The main problem is that an adaptive (and rushing) adversary first learns the message to be broadcasted, and then, depending on the learned message, still can corrupt the sender and make him broadcast a different message. The authors are not aware of any broadcast protocol in the literature that does not suffer from this problem (but see related work below).

The positive result for perfect security with $t < n/3$ is rather straight-forward: First, the sender secret shares the message among the players. Then, the sharing is reconstructed. The only issue is how to do a secret-sharing without having a composable broadcast primitive. The second positive result, namely statistical and computational security for $t \leq n/2$, it more involved. Note that verifiable secret-sharing exists only for $t < n/2$ (but not for $t = n/2$).

The tightness of the bound for perfect security ($t < n/3$) follows directly from the impossibility of property-based broadcast. The tightness of $t \leq n/2$ is proven indirectly; we show that in any "broadcast protocol" for $2t = n + 1$, there exists a round in which the adversary (not corrupting the sender) obtains noticeable (i.e., not negligible) information about the message, but still can corrupt the sender and change the message.

1.4 Related Work

At first glance, our constructions of simulation-based broadcast look similar to simultaneous broadcast [CGMA85, CR87, Gen00]. However, the goal of the latter is to satisfy an additional property, not to achieve simulation-based security. Recently, Hevia [Hev06] proposed a universal composable simultaneous broadcast protocol. However, as all previous protocols in this line of research, also this protocol uses "normal" broadcast as sub-protocol, and the security analysis relies on the hope that this securely composes, which in general is not the case.[1]

Recently, Lindell, Lysyanskaya, and Rabin [LLR02] have presented a broadcast protocol for $t < n$, which is concurrently composable when unique session IDs are available. This result does not contradict ours, as it is for the authenticated communication model only, where all messages sent through the network can be read by the adversary. In particular, even in the ideal world, the adversary can read the message that the sender sends to the broadcast functionality, and (by corrupting the sender) can also change it. Hence, also this protocol shows this unexpected phenomena that the adversary can decide to corrupt the sender and change the broadcast message *after* she has seen the broadcasted message, but this is allowed, as she could do so also in the ideal world. In particular, when this protocol would be executed in a model with secure channels, it would not remain secure.

2 Preliminaries

We consider the well-known secure channels model introduced in [BGW88, CCD88]: the players in $\mathcal{P} = \{p_1, \ldots, p_n\}$ are connected by a complete network of bilateral secure channels. The communication is synchronous, i.e., all players have synchronized clocks and there is a known upper bound on the delivery time of any sent message. In such a synchronous model, the protocols proceed in rounds, where in each round several players might be instructed to send messages to several other players. All the messages sent in some round are delivered at the beginning of the next round.

2.1 The Adversary model

We consider a threshold adversary who can actively corrupt up to t players (we refer to this adversary as t*-adversary*). When some player p_i is actively corrupted then the adversary has full control on p_i. A player who is not corrupted is called *uncorrupted* or *honest*. Analogously, the corrupted players are also called *dishonest*.

Adaptive We consider an *adaptive* adversary. As opposed to a *static* adversary who chooses the players to corrupt at the beginning of the protocol, an adaptive adversary can corrupt additional players in the flow of the protocol depending on messages seen so far, with the only restriction that the total number of players she corrupts has to be at most t. As one would expect, an adaptive adversary is stronger than a static [CDD+01]. Our results confirm this belief.

Rushing We consider a *rushing* adversary [Can00, Can03]. Intuitively, such an adversary is allowed to first receive all the messages sent to the corrupted players in a round, and then decide its strategy. A bit more formally, a rushing adversary can decide in each round the order in which the messages of this round are sent. When, in addition to being rushing, the adversary is also adaptive, then after each message which is sent, the adversary is allowed to corrupt more players depending on the messages she has seen so far. In particular, the adversary can, within some round, have an honest p_i send a message to some corrupted p_j, and depending on this message corrupt p_i before he sends any further message. In this work, we consider an adversary who is both rushing and adaptive.

2.2 Composable Security

Following the [Can00, Can03] methodology security of protocols is argued via the ideal-world/real-world paradigm. In the real-world the players execute the protocol. The ideal-world is a specification of the task which we want the protocol to implement. More concretely, in the ideal-world the players can invoke a fully trusted party, called the *functionality*, denoted as $\mathcal{F}$, in the following way: the player sends their input(s) to $\mathcal{F}$; $\mathcal{F}$ runs its program on the received inputs (while running the program, $\mathcal{F}$ might receive additional inputs

[1] The protocol in [Hev06] employs the verifiable secret-sharing scheme from [CDD+99], which in turn employs some broadcast primitive which securely composes in the secure-channels model for $t < n/2$. So far, no such protocol is known.

from the players or the adversary or send values to the adversary), and returns to the players their specified outputs. The specification of $\mathcal{F}$ is such that this ideal-evaluation captures, as good as possible, the goals of the designed protocol.

Intuitively, a protocol securely realizes a functionality $\mathcal{F}$, when the adversary cannot achieve more in the protocol than what she could achieve in an ideal-evaluation of $\mathcal{F}$. To formalize this statement, we assume an environment $\mathcal{Z}$ which decides the inputs of all players, and, at the end of the computation, gets to see their outputs. $\mathcal{Z}$ also sees the full view of the adversary $\mathcal{A}$ who is attacking the protocol. We denote the view of $\mathcal{Z}$ for an invocation of protocol π with adversary $\mathcal{A}$ as $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. A protocol π *t-securely realizes functionality* $\mathcal{F}$ when for any t-adversary $\mathcal{A}$ attacking protocol π, there exists an ideal-world adversary $\mathcal{S}$ (also called the *simulator*) such that no environment $\mathcal{Z}$ cannot tell whether it is interacting with $\mathcal{A}$ and the players running π or with $\mathcal{S}$ and the players running the ideal-world protocol (we denote the view of $\mathcal{Z}$ in an ideal-evaluation of $\mathcal{F}$ as $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$).

The three typical security notions are: *perfect security* ($\mathcal{A}$ is computationally unbounded, and the random variables $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are identically distributed), *statistical security* ($\mathcal{A}$ is computationally unbounded, and $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are statistically close), and *computational security* ($\mathcal{A}$ is efficient, and $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable)

The $\mathcal{F}$-hybrid model The power of the simulation-based definition is that it allows to argue about security of protocols in a composable way. In particular, let π_1 be a protocol which securely realizes a functionality $\mathcal{F}_1$. If we can prove that π_2 securely realizes a functionality $\mathcal{F}_2$ using *ideal-calls* to $\mathcal{F}_1$, then it follows automatically that the protocol which results by replacing, in π_2, the calls to $\mathcal{F}_1$ by invocations of π_1 also securely realizes $\mathcal{F}_2$. Therefore we only need to prove the security of π_2 in the so-called $\mathcal{F}_1$-*hybrid* model, where the players run π_2 and are allowed to make ideal-calls to $\mathcal{F}_1$. For more details on composability of protocols and a formal handling of both sequential and parallel composition (and also of universal composability), the reader is referred to [Can00, Can03].

3 Perfect Security

In this section we consider the case of perfect security, i.e., information theoretic (i.t.) with no error probability. We show that perfectly secure broadcast tolerating a t-adversary is possible if and only if $t < n/3$. Although this bound already appears in the literature, to the best of our knowledge, none of suggested synchronous broadcast protocols for perfect security satisfies the simulation-based definition when secure-channels and an adaptive adversary are considered. Also, in addition to handling the perfect security case, this section serves as a good way to introduce many of our ideas.

The ideal functionality for broadcast $\mathcal{F}_{\mathrm{BC}}$ when synchronous secure channels are assumed is quite intuitive; nevertheless, to keep our analysis complete, in the following we give a description. For simplicity we describe the functionality in terms of an ideal-world protocol. A UC version of this functionality can be found in the appendix.

Functionality $\mathcal{F}_{\mathrm{BC}}$

1. p_s sends his input x_s to the Trusted Party (denoted as $\mathcal{F}_{\mathrm{BC}}$).
2. $\mathcal{F}_{\mathrm{BC}}$ sends x_s to every $p \in \mathcal{P}$.

To show that the above functionality is not realized by known protocols, we observe that these protocols follow the same paradigm: At the beginning of the protocol the sender p_s sends his input x_s to the players in $\mathcal{P} \setminus \{p_s\}$; in a second phase the players try to establish a consistent view on the sender's input. Clearly all protocols which start by the sender multisending his input and then invoke a consensus protocol on the received value, e.g., [BGP89, CW92, BDDS92], are of the above type. However, even the protocols where the second phase is not a self-contained consensus protocol, e.g., the Dolev-Strong [DS82] or the Pfitzmann-Waidner [PW92] broadcast, also follow the above paradigm.

The fact that any protocol following the above paradigm is insecure against an adaptive adversary can be seen as follows: In any such protocol, because the adversary is rushing, she can corrupt the first player who receives the input x_s from p_s and depending on the received value decide whether or not to also corrupt the sender p_s (and possibly change the broadcasted value). However, this behavior cannot be simulated, as by the time the simulator learns x_s from the functionality it is already too late to change it (the functionality also sends it to all honest players).

A direct way to deal with the above problem is to make sure that before any player (or the adversary) learns any information on x_s, the value x_s is secret shared in a robustly reconstructible way. More concretely, when a secure Verifiable Secret Sharing (VSS) scheme is given, then one can easily construct a secure broadcast protocol (i.e., a protocol realizing $\mathcal{F}_{\text{BC}}$) by invoking the VSS protocol with dealer p_s on input x_s (where x_s is reconstructed towards all players). Hence, in order to build a perfectly secure broadcast protocol it suffices to provide a perfectly secure VSS.[2]

It might look that we are done, as one could use the perfectly secure VSS from [BGW88] to achieve broadcast (for a proof of security of this VSS scheme see [BGW88, Can00]). But this is not quite true. The reason is that [BGW88] (and all other known VSS schemes with perfect security) use broadcast as a component. If we instantiate this component by one of the known broadcast protocols then we can no longer argue about the security of the full construction using composability. Nevertheless, we show in the following that replacing all broadcasting of values in the [BGW88] VSS scheme by invocations to the [BGP89] broadcast protocol[3] does not cause any loss of security; we shall denote this VSS scheme by $\text{VSS}_{\text{BGW}}^{(\text{BGP})}$.

The security of $\text{VSS}_{\text{BGW}}^{(\text{BGP})}$ is argued in two steps. In a first step, we show that although the [BGP89] broadcast protocol, denoted in the following as BC_{BGP}, does not securely realize $\mathcal{F}_{\text{BC}}$, it does realize a weaker functionality, denoted as $\mathcal{F}_{\text{UBC}}$ (we refer to this functionality as *unfair broadcast*). In a second step, we show that under certain conditions (which are satisfied by the [BGW88] VSS protocol) , we can replace $\mathcal{F}_{\text{BC}}$ by $\mathcal{F}_{\text{UBC}}$ without loosing security.

The functionality $\mathcal{F}_{\text{UBC}}$ is described in the following. Intuitively, the difference to the functionality $\mathcal{F}_{\text{BC}}$ is that $\mathcal{F}_{\text{UBC}}$ allows the adversary to first receive the sender's p_s input (even without corrupting p_s) and then, depending on the received value, decide whether or not she wants to corrupt p_s and possibly modify the broadcasted value.

Functionality $\mathcal{F}_{\text{UBC}}$

1. p_s sends his input x_s to the Trusted Party (denoted as $\mathcal{F}_{\text{UBC}}$).
2. $\mathcal{F}_{\text{UBC}}$ sends x_s to the adversary.
3. If p_s is corrupted then the adversary sends a value to $\mathcal{F}_{\text{UBC}}$; $\mathcal{F}_{\text{UBC}}$ denotes the received value by x'_s (if p_s is not corrupted then $\mathcal{F}_{\text{UBC}}$ sets $x'_s := x_s$).
4. $\mathcal{F}_{\text{UBC}}$ sends x'_s to every $p \in \mathcal{P}$

Lemma 1. *Protocol BC_{BGP} perfectly t-securely realizes the functionality $\mathcal{F}_{\text{UBC}}$ for $t < n/3$.*

Proof. (sketch) As shown in [BGP89], the protocol BC_{BGP} satisfied the property-based definition of broadcast (i.e., it satisfies validity, consistency, and termination). We show that it perfectly securely realizes $\mathcal{F}_{\text{UBC}}$. Let $\mathcal{A}$ be an adversary attacking BC_{BGP}; a corresponding simulator $\mathcal{S}$ can be built as follows: First S waits for the input x_s from $\mathcal{F}_{\text{UBC}}$. Note that, because BC_{BGP} is fully deterministic and the players in $\mathcal{P} \setminus \{p_s\}$ have

[2] The idea of using VSS to improve the properties of broadcast has also been considered in the past for achieving simultaneous broadcast [CGMA85, CR87, Gen95, Gen00, HM05, Hev06].

[3] In fact [BGP89] describes a consensus protocol. A protocol for broadcast can be constructed by having the sender send his value to everybody and then invoke consensus on the received values.

no input, knowing x_s allows $\mathcal{S}$ to perfectly simulate all the messages sent by honest players.[4] $\mathcal{S}$ invokes $\mathcal{A}$ and does the following:

1. $\mathcal{S}$ simulates all the players in the computation.
2. Whenever $\mathcal{A}$ requests to corrupt some $p_i \in \mathcal{P}$, $\mathcal{S}$ corrupts p_i and sends (the simulated) internal state of p_i to $\mathcal{A}$. From that point on, $\mathcal{S}$ has (the simulated) p_i follow $\mathcal{A}$'s instruction.
3. Whenever $\mathcal{A}$ sends a message to the environment $\mathcal{Z}$, $\mathcal{S}$ forwards this message to $\mathcal{Z}$.
4. At the end of the simulation, if some (simulated) uncorrupted player p_i outputs $x_i = x_s$, then in the ideal evaluation p_s sends x_s to $\mathcal{F}_{\text{UBC}}$ (even when he is corrupted). Otherwise, i.e., if $x_i \neq x_s$, $\mathcal{S}$ instructs p_s to send x_i to the functionality $\mathcal{F}_{\text{UBC}}$ in Step 3 (the correctness property of BC_{BGP} implies $x_i \neq x_s$ only when p_s is actively corrupted.).

It is easy to verify that $\text{EXEC}_{\mathcal{F}_{\text{UBC}},\mathcal{S},\mathcal{Z}} \equiv \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, i.e., the protocol perfectly securely realizes $\mathcal{F}_{\text{UBC}}$. □

For the second step, we show that if a protocol has a special property, then it is safe to replace $\mathcal{F}_{\text{BC}}$ by $\mathcal{F}_{\text{UBC}}$. The property is the following: For any value v which is supposed to be broadcasted, the adversary *"knows v in advance"*, i.e., there exists a *deterministic* strategy for this adversary to compute v based on the contents of her view *before* the call to the broadcast primitive. We formalize this in the following lemma. Note that the lemma holds for any security level and is not restricted to perfect security.[5]

Lemma 2. *Let Π be an $\mathcal{F}_{BC}$-hybrid protocol which securely realizes a given functionality $\mathcal{F}$, and let Π' denote the protocol which results by replacing in Π all the calls to $\mathcal{F}_{BC}$ with calls to $\mathcal{F}_{UBC}$. If Π uses calls to $\mathcal{F}_{BC}$ only to broadcast values which the adversary knows in advance, then Π' securely realizes $\mathcal{F}$.*

Proof. (sketch) Let $\mathcal{A}'$ be an adversary attacking Π' in the $\mathcal{F}_{\text{UBC}}$-hybrid model. We show how to construct an adversary $\mathcal{A}$ attacking Π in the $\mathcal{F}_{\text{BC}}$-hybrid model such that $\text{Exec}_{\mathcal{Z},\mathcal{A}',\Pi'} \equiv \text{Exec}_{\mathcal{Z},\mathcal{A},\Pi}$. This is sufficient as then we can use the simulator for $\mathcal{A}$ (which is guaranteed to exist by the security of Π) as a simulator for $\mathcal{A}'$. $\mathcal{A}$ behaves exactly as $\mathcal{A}'$ except in the invocations of $\mathcal{F}_{\text{UBC}}$: when $\mathcal{F}_{\text{UBC}}$ is to be called, in order to simulate the first message of $\mathcal{F}_{\text{UBC}}$ towards $\mathcal{A}'$ (corresponding to the broadcasted value) $\mathcal{A}$ computes the value to be broadcasted (using the deterministic strategy on his view which is guaranteed to exist by the fact that she knows the broadcasted value in advance)[6] and sends this value to $\mathcal{A}'$. $\mathcal{A}'$ is now allowed to corrupt the sender and (possibly) change the value he is supposed to broadcast. $\mathcal{A}$ acts accordingly and then invokes $\mathcal{F}_{\text{BC}}$. It is straightforward to verify that $\text{Exec}_{\mathcal{Z},\mathcal{A}',\Pi'} \equiv \text{Exec}_{\mathcal{Z},\mathcal{A},\Pi}$. □

We point out that the [BGW88] VSS protocol satisfies the pre-condition of Lemma 2. Indeed, the protocol uses broadcast only for the complaints and the accusations issued by players and for the dealer to reply to them. By careful inspection of the protocol one can verify that all these broadcasted values can be computed from the view of the adversary before they are broadcasted. Because this VSS is secure for $t < n/3$, combining Lemmas 1 and 2 we get the following corollary.

Corollary 1. *If $t < n/3$ then there exists a protocol which perfectly t-securely realizes the functionality $\mathcal{F}_{BC}$.*

Remark 1. Although replacing $\mathcal{F}_{\text{BC}}$ by $\mathcal{F}_{\text{UBC}}$ did not affect the security of [BGW88] VSS, this is not in general true for any protocol using broadcast. In fact, even when the broadcasted message is uniformly and independently chosen (intuitively this implies that the adversary gets not information by learning it) it is not

[4] In fact, for any adversary $\mathcal{A}$, $\mathcal{S}$ can generate exactly the same messages as the uncorrupted players would if the protocol would be run with this adversary.

[5] However, for the case of computational security we will have to require that the strategy of the adversary to compute the value which is to be broadcasted is efficient.

[6] Wlog we can assume that $\mathcal{A}'$ forwards his entire view to $\mathcal{A}$ [Can00, Can01].

always the case that we can do this replacement. For example, consider the protocol where each player p_i in turns broadcasts a uniformly random bit and the output is the vector of the broadcasted bits. If broadcasting is done by $\mathcal{F}_{\text{UBC}}$ then the adversary can corrupt the p_i's (up to t players in total) who are about to broadcast 0 and change the broadcasted value to 1. This will create a bias towards 1 in the output distribution, which cannot be simulated if $\mathcal{F}_{\text{BC}}$ is used for broadcasting, as the bits are chosen randomly. Note that this is *not* a simultaneous-broadcast problem, as the we do not require that the adversary's decisions are independent of what the honest players broadcast.

To complete this section we show that $t < n/3$ is tight for perfectly secure broadcast. We use the following impossibility result from [LSP82, KY84, FLM86] (for a nice proof see also [Fit03]).

Lemma 3 ([LSP82, KY84, FLM86, Fit03]). *If $t \geq n/3$ then there exists no protocol which simultaneously satisfies correctness, consistency, and termination, even in the presence of a non-adaptive adversary.*

The impossibility proof for the functionality $\mathcal{F}_{\text{BC}}$ follows directly from the above lemma and the fact that any protocol securely realizing $\mathcal{F}_{\text{BC}}$ has to satisfy the given three properties.

Corollary 2. *If $t \geq n/3$ then there exists no protocol which perfectly t-securely realizes the functionality $\mathcal{F}_{BC}$.*

We point out that Lemma 3, hence also the impossibility for $\mathcal{F}_{\text{BC}}$, holds even for the cases of computational and statistical security *when no setup is available*. This implies the following:

Corollary 3. *If $t \geq n/3$ and no setup is available then there exists no protocol which computationally t-securely realizes the functionality $\mathcal{F}_{BC}$. The statement holds also for statistical security.*

4 Statistical and Computational Security (with a trusted setup)

In this section we consider the cases of statistical security, i.e., information theoretic with negligible error-probability, and computational security. For these security notions, it is widely believed that when a setup allowing digital signatures is assumed, then broadcast is possible for an arbitrary number of cheaters (i.e., $t < n$), e.g., by using the Dolev-Strong broadcast protocol [DS82] for computational security or using [PW92] for statistical security. We show that this folklore belief is wrong when an adaptive adversary is considered. We already argued in the previous section that the Dolev-Strong broadcast protocol is not adaptively secure. In this section we show that the condition $t \leq n/2$ is necessary and sufficient for broadcast both for computational and statistical security.

We start by proving the sufficiency of the condition $t \leq n/2$; this is done by providing a protocol which securely realizes $\mathcal{F}_{\text{BC}}$. We handle the two security notions, i.e., computational and statistical, in parallel. In our protocol, the players will need to digitally sign messages they send. This is modeled by assuming that the protocol has access to an ideal functionality for digital signatures $\mathcal{F}_{\text{SIG}}$ (for definition and properties of such a functionality see [Can03]).

4.1 The Broadcast Protocol

Our approach is analogous to the case of perfect security, namely we first show that there exists a secure realization of $\mathcal{F}_{\text{UBC}}$ for $t \leq n/2$, and then use Lemma 2 to derive a protocol for $\mathcal{F}_{\text{BC}}$ from an $\mathcal{F}_{\text{UBC}}$-hybrid protocol. However, this last step is more involved than simply using a statistically secure VSS protocol satisfying the preconditions of Lemma 2. Indeed, all known protocols for statistical VSS are only secure for $t < n/2$ which is stronger than $t \leq n/2$. Before describing how to overcome this difficulty we state the following lemma which will allow us to use Π_{DS} as a secure realization of $\mathcal{F}_{\text{UBC}}$. The proof is along the lines of the proof of Lemma 1; the only difference is that the simulator needs also to simulate the digital signatures of honest players in a run of the protocol, which is guaranteed to be able to do by the definition of $\mathcal{F}_{\text{SIG}}$.[7]

[7] The idea that when using [DS82] with i.t. secure signatures we get an i.t. secure broadcast protocol appears also in [PW92, Fit03].

Lemma 4. *Protocol Π_{DS} perfectly t-securely realizes $\mathcal{F}_{UBC}$ for $t < n$ in the $\mathcal{F}_{SIG}$-hybrid model, where the signatures are replaced by calls to an ideal signature functionality $\mathcal{F}_{SIG}$.*

To implement the second step, namely construct the $\mathcal{F}_{\text{UBC}}$-hybrid protocol, we use as starting point the VSS from [CDD+99]. In [CDD+99] IC-signatures are used to ensure that some p_j who receives a value v from some p_i can, at a later point, publicly prove that p_i indeed send him v. Because IC-signatures are secure only when $t < n/2$, in this work we use digital signatures for the same purpose. The signatures are generated and verified by calls to the assumed digital signatures functionality $\mathcal{F}_{\text{SIG}}$. We point out that the signed message should include enough information to uniquely identify for which message in the flow of the protocol the signature was issued (e.g., a unique message ID associated with every message sent in the protocol). Depending on whether the calls to $\mathcal{F}_{\text{SIG}}$ are instantiated by a computationally or an i.t. secure signature-scheme, our broadcast protocol will achieve computational, respectively i.t. security.

In the following, we first describe our sharing, which is along the lines of [CDD+99], and specify some useful security properties, and then we describe and analyze our broadcast protocol.

Secret Sharing Following the terminology of [CDD+99], we say that a vector $v = (v_1, \ldots, v_m) \in \mathbb{F}^m$ is d-consistent, if there exists a polynomial $p(\cdot)$ of degree d such that $p(i) = v_i$ for $i = 1, \ldots, m$. A value s is said to be *d-shared* among the players in $\mathcal{P}$ when every (honest) player $p_i \in \mathcal{P}$ holds a degree-d polynomial $g_i(\cdot)$ and for each $p_j \in \mathcal{P}$ p_i also holds p_j's signature on $g_i(j)$, where the following condition holds: there exists a degree-d polynomial $q(\cdot)$ with $q(0) = s$ and $g_i(0) = q(i)$ for all p_i. The polynomials $g_1(\cdot), \ldots, g_n(\cdot)$ along with the corresponding signatures constitute a *d-sharing* of s.

We describe the protocols HD-Share (the HD stands for Honest Dealer) and Reconstruct which allow for a dealer p_D to d-share a value s, and for public reconstruction of a shared value, respectively.

The protocol HD-Share is along the lines of the sharing protocol from [CDD+99]. We describe HD-Share (see next page) in the $\{\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{BC}}\}$-hybrid model, i.e., HD-Share uses calls to $\mathcal{F}_{\text{BC}}$ for broadcasting and calls to $\mathcal{F}_{\text{SIG}}$ for signature generation and verification. To ensure that the output of HD-Share matches the form of our sharing, i.e., every honest p_i holds a degree-d polynomial $g_i(\cdot)$ and signatures from all other players, we do the following: for every message transmission, the receiver p_j confirms when he receives a well-formed message from p_i or, otherwise, p_j complains and p_i is expected to answer the complaint by broadcasting the message. If some p_i is publicly caught to misbehave, e.g., by broadcasting a malformed message, then p_i is disqualified. Because dishonest players cannot be forced to sign the messages they send, we make the following convention: when p_i is disqualified, then every player takes a default value, denoted as $\perp$, to be p_i's signature on any message ($\perp$ will always be accepted as valid signature of disqualified players on any message). The proof of the lemma is along the lines of the security analysis in [CDD+99] and is, therefore, moved to the appendix.

Lemma 5. *Protocol* HD-Share *invoked in the $\{\mathcal{F}_{BC}, \mathcal{F}_{SIG}\}$-hybrid model has the following properties. Privacy: The view of any d-adversary attacking the protocol can be perfectly simulated. (This property ensures that no information about s leaks to a d-adversary); Honest-dealer correctness: When the dealer is honest until the end of* HD-Share *then the output is a d-sharing of s. Furthermore, in all calls to $\mathcal{F}_{BC}$, the adversary "knows in advance" the value to be broadcasted.*

To reconstruct a sharing the protocol Reconstruct is invoked. The idea is the following: every p_i announces his share and the corresponding signatures from the players in $\mathcal{P}$; if some signature is invalid or the the announced share is not d-consistent, then p_i is excluded from the reconstruction, otherwise his share-polynomial is interpolated; The zero-coefficients of the share-polynomials of the players that have not been excluded are used to reconstruct the shared value. Depending on the actual choice of d the sharing might not uniquely define a value. In any case the players adopt the value which is output by the interpolation algorithm. The consistency of the output is guaranteed as it is decided on publicly seen values.

Protocol HD-Share$_d$ (p_D, s)

1. The dealer p_D chooses a uniformly random bivariate polynomial $f(\cdot,\cdot)$ of degree d in each variable, such that $f(0,0) = s$. For each $p_i \in \mathcal{P}$:
   (a) For $j = 1, \ldots, n$: p_D sends p_i the values $s_{i,j} = f(i,j)$ and $s_{j,i} = f(j,i)$ along with his signature on them; p_i denotes the received values as $s_{i,j}^{(i)}, s_{j,i}^{(i)}, sig_{p_D}(s_{i,j}^{(i)})$, and $sig_{p_D}(s_{j,i}^{(i)})$.
   (b) p_i broadcasts a complaint if any of the vectors $\left(s_{i,1}^{(i)}, \ldots, s_{i,n}^{(i)}\right)$ and $\left(s_{1,i}^{(i)}, \ldots, s_{n,i}^{(i)}\right)$ is not d-consistent or for some value no valid signature was received.
   (c) p_D answers each complaint by broadcasting the values he sent to p_i in Step 1a. If p_D broadcasts a message of the wrong form or invalid signatures then p_D is disqualified; otherwise p_i adopts the broadcasted messages as the messages he should have received in Step 1a.
2. For each $p_i \in \mathcal{P}$:
   (a) For $j = 1, \ldots, n$: p_i sends $s_{i,j}^{(i)}$ to p_j along with his signature $sig_{p_i}(s_{i,j}^{(i)})$ and the dealer's signature $sig_{p_D}(s_{i,j}^{(i)})$.
   (b) Each $p_j \in \mathcal{P}$ broadcasts a complaint if he did not receive a message along with valid signatures from p_i and p_D in Step 2a.
   (c) p_i answers each complaint by broadcasting $(s_{i,j}^{(j)}, sig_{p_D}(s_{i,j}^{(j)}), sig_{p_j}(s_{i,j}^{(i)}))$. If p_i does not broadcast a message or any of the signatures is invalid then p_i is disqualified, every player replaces all p_i's signatures by $\perp$, and p_j sets $s_{i,j}^{(j)} := s_{i,j}^{(i)}$; otherwise p_j adopts the broadcasted messages as the messages he should have received in Step 2a.
3. Every p_i checks if he received a $s_{i,j}^{(j)}$ from some p_j in Step 2 which is inconsistent with his own view of $s_{i,j}$, i.e., $s_{i,j}^{(j)} \neq s_{i,j}^{(i)}$, and if so, broadcasts $\left(s_{i,j}^{(i)}, s_{i,j}^{(j)}, sig_{p_D}(s_{i,j}^{(i)}), sig_{p_D}(s_{i,j}^{(j)})\right)$; every player verifies that $s_{i,j}^{(j)} \neq s_{i,j}^{(i)}$ and that the signatures are valid and if so p_D is disqualified.[a]

[a] The signature includes information to uniquely identify for which message in the flow of the protocol it was generated, e.g. a unique message ID, and also includes a unique session ID.

Protocol Reconstruct

1. Each $p_i \in \mathcal{P}$ broadcasts $(s_{i,1}, \ldots, s_{i,n})$ along with the corresponding signatures $sig_{p_1}(s_{i,1}), \ldots, sig_{p_n}(s_{i,n})$; if any of the broadcasted signatures is invalid or if the broadcasted vector is not d-consistent, then p_i is disqualified. Otherwise a polynomial $g_i(\cdot)$ is defined by interpolating the components of the vector.
2. Let $P_{\text{“ok”}} = \{p_{i_1}, \ldots, p_{i_\ell}\}$ denote the set of non-disqualified players. The values $g_{i_1}(0), \ldots, g_{i_\ell}(0)$ are used to interpolate a polynomial $g'(\cdot)$ and every player outputs $g'(0)$.

Lemma 6. *Protocol* Reconstruct *invoked to the $\{\mathcal{F}_{BC}, \mathcal{F}_{SIG}\}$-hybrid model outputs (the same) $y \in \mathbb{F}$ towards every player. Furthermore, if $d < n - t$ (where t is the number of corrupted players) and the input is a d-consistent sharing of some s, then $y = s$.*

We next describe our broadcast protocol for $t \leq n/2$. The idea is to have the sender p_s share his input x_s by a degree-$(t-1)$ sharing using HD-Share with $d = t-1$ and subsequently invoke Reconstruct on the output of HD-Share. The intuition is the following: When p_s is honest until the end of HD-Share, then HD-Share outputs a $(t-1)$-sharing of x_s (Lemma 5: honest-dealer correctness); as $t \leq n/2$ implies $d = t-1 < n-t$, Lemma 6 guarantees that Reconstruct will output x_s. Hence, the only way the adversary can change the output to some $s' \neq s$ is by corrupting p_s during (or before) protocol HD-Share. As there are at most t corrupted players, if the adversary wishes to corrupt p_s then she can corrupt at most $t-1$ of the remaining players; as a $(t-1)$-adversary gets no information on x_s (Lemma 5: privacy), this decision, i.e., whether or not to corrupt p_s, has to be taken independently of x_s, which is a behavior that can be easily simulated.

As already mentioned, both HD-Share and Reconstruct use calls to $\mathcal{F}_{\text{BC}}$ for broadcasting. In order to replace $\mathcal{F}_{\text{BC}}$ by $\mathcal{F}_{\text{UBC}}$, we need to make sure that the precondition of Lemma 2 is satisfied (i.e., the adversary "knows in advance" all broadcasted values). For protocol HD-Share this is guaranteed by Lemma 5. However, the values which are broadcasted in Reconstruct are not necessarily known to the adversary in advance. We resolve this by a technical trick, namely we introduce a *dummy* step between HD-Share and Reconstruct where every player sends to every other player his output from protocol HD-Share. Observe that this modification might give an advantage to the adversary only when she has not corrupted p_s by the end of HD-Share, as otherwise she knows all the outputs. But in this case at the end of HD-Share the output is already fixed to x_s and even giving the adversary access to the full transcript does not allows her to change the output. For completeness we include a description of our broadcast protocol and state its achieved security. Due to space restrictions the formal security proof has been moved to the appendix.

Protocol Broadcast (p_s, x_s)

1. Invoke HD-Share$_{t-1}(p_s, x_s)$; if p_s is disqualified then every player outputs a default value, e.g., 0 and halts.
2. *(dummy step)* Every $p_i \in \mathcal{P}$ sends his output from HD-Share to every $p_j \in \mathcal{P}$.
3. Invoke Reconstruct on the output of HD-Share.

Lemma 7. *Protocol* Broadcast *perfectly t-securely realizes the functionality $\mathcal{F}_{BC}$ in the $\{\mathcal{F}_{UBC}, \mathcal{F}_{SIG}\}$-hybrid model, for $t \leq n/2$.*

Combining the above lemma with Lemma 4 we get the following.

Corollary 4. *If $t \leq n/2$ and a statistically (resp. computationally) secure signature scheme is available then there exists a protocol which statistically (resp. computationally) t-securely realizes the functionality $\mathcal{F}_{BC}$.*

To complete this section, we show that the condition $t \leq n/2$ is necessary for adaptively secure synchronous broadcast both for i.t. and for computational security. The idea of the proof is the following: Because the adversary can corrupt half of the players in $\mathcal{P} \setminus \{p_s\}$, she can be the first to learn noticealbe information on the dealers input, before the honest players in $\mathcal{P} \setminus \{p_s\}$ jointly learn noticable information. Depending on this information the adversary can corrupt the sender and, with overwhelming probability, change the output to some other value. However this behaviour cannot be simulated. A detailed proof can be found in the appendix.

Lemma 8. *There exists no protocol which computationally t-securely realizes the functionality $\mathcal{F}_{BC}$ for $t > n/2$. The statement holds also for statistical security.*

5 Conclusions

We considered the problem of securely realizing broadcast in the secure-channels model. In this model, it has been shown that there exist protocols satisfying the property-based definition of broadcast and tolerating a t-adversary, if and only if $t < n/3$ when perfect security is considered. For unconditional and computational security, when a setup allowing digital signatures is given, the corresponding bound is $t < n$.

We showed that, when a rushing and adaptive adversary is considered, known broadcast protocols do not realize the natural ideal functionality for broadcast in the secure-channels model. As a result, if one replaces the broadcast invocations in any of the known muli-party protocols for the secure-channels model, e.g., [BGW88, RB89, CDD+99], by one of the known broadcast protocols, the security of the resulting protocol cannot be argued using the composition theorems.

We described protocols which securely realize the (natural) ideal functionality for broadcast for each of the three security notions. For the case of perfect security, we showed that the tight bound matches the corresponding bound for the property-based definition, i.e., $t < n/3$. However, for the cases of statistical and computational security (with setup assumptions) the necessary and sufficient bound is $t \leq n/2$.

References

[BDDS92] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. *Information and Computation*, 97(2):205–233, 1992.

[BGP89] P. Berman, J. Garray, and J. Perry. Towards optimal distributed consensus. In *FOCS '89*, pp. 410–415, 1989. Full version in *Computer Science Research*, 1992.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88*, pp. 1–10, 1988.

[BPW91] B. Baum-Waidner, B. Pfitzmann, and M. Waidner. Unconditional Byzantine agreement with good majority. In *STAC '91*, volume 480 of *LNCS*, pp. 285–295, 1991.

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library, 2003.

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

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS 2001*, pp. 15–17, 2001.

[Can03] R. Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. http://eprint.iacr.org/.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC '88*, pp. 11–19, 1988.

[CDD$^+$99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pp. 311–326, 1999.

[CDD$^+$01] R. Canetti, I. Damgaard, S. Dziembowski, Y. Ishai, and T. Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pp. 262–279, 2001.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS '85*, pp. 383–395, Washington, DC, USA, 1985.

[CR87] B. Chor and M. Rabin. Achieving independence in logarithmic number of rounds. In *PODC '87*, pp. 260–268, New York, NY, USA, 1987. ACM.

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

[DM00] Y. Dodis and S. Micali. Parallel reducibility for information-theoretically secure computation. In *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pp. 74–92, 2000.

[DS82] D. Dolev and H. Strong. Polynomial algorithms for multiple processor agreement. In *STOC '82*, pp. 401–407, 1982. Full version in *SIAM Journal on Computing*, 12(4):656–666, 1983.

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

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

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

[FM89] P. Feldman and S. Micali. An optimal probabilistic algorithm for synchronous Byzantine agreement. In *Automata, languages and programming*, volume 372 of *Lecture Notes in Computer Science*, pp. 341–378, 1989.

[Gen95] R. Gennaro. Achieving independence efficiently and securely. In *PODC '95*, pp. 130–136, New York, NY, USA, 1995. ACM.

[Gen00] R. Gennaro. A protocol to achieve independence in constant rounds. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):636–647, 2000.

[GM98] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM Journal on Computing*, 27(1):247–290, Feb. 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *STOC '87*, pp. 218–229, 1987.

[Hev06] A. Hevia. Universally composable simultaneous broadcast. In *SCN 2006*, pp. 18–33, 2006.

[HM05] A. Hevia and D. Micciancio. Simultaneous broadcast revisited. In *PODC '05*, pp. 324–333, New York, NY, USA, 2005. ACM.

[KY84] A. Karlin and A. Yao. Manuscript, 1984.

[LLR02] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. In *STOC 2002*, pp. 514–523, 2002.

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

[MW88] Y. Moses and O. Waarts. Coordinated traversal: $(t+1)$-round Byzantine agreement in polynomial time. In *FOCS '88*, pp. 246–255, 1988.

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

[PW92] B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *STACS '92*, Lecture Notes in Computer Science, pp. 339–350, 1992.

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

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC '89*, pp. 73–85, 1989.

Appendix

Functionality $\mathcal{F}_{\text{BC}}$

1. Upon receiving an input $(\texttt{Broadcast}, sid, x_s)$ from p_s record x_s.
2. Upon receiving a message $(\texttt{ReceiveOutput}, sid)$ from some party p_i, if x_s was recorded send x_s to p_i.

Functionality $\mathcal{F}_{\text{UBC}}$

1. Upon receiving an input $(\texttt{Broadcast}, sid, x_s)$ from p_s record x_s.
2. Upon receiving a message $(\texttt{ReceiveOutput}, sid)$ from some party p_i, if x_s was recorded send x_s to p_i.
3. Upon receiving a message $(\texttt{Modify}, sid, x'_s)$ from the adversary if p_s has been corrupted, and x_s was recorded and is not yet written on the tape of some uncorrupted player, then change the recorded value to x'_s.

Lemma 5. *Protocol* HD-Share *invoked in the* $\{\mathcal{F}_{BC}, \mathcal{F}_{SIG}\}$*-hybrid model has the following properties. Privacy: The view of any* d*-adversary attacking the protocol can be perfectly simulated. (This property ensures that no information about* s *leaks to a* d*-adversary); Honest-dealer correctness: When the dealer is honest until the end of* HD-Share *then the output is a* d*-sharing of* s*. Furthermore, in all calls to* $\mathcal{F}_{BC}$*, the adversary "knows in advance" the value to be broadcasted.*

Proof. (sketch) The privacy of HD-Share can be argued along the lines of [CDD+99]. Nevertheless we give a sketch of the simulator. Simulating the signatures is of no issue as the functionality $\mathcal{F}_{\text{SIG}}$ allows $\mathcal{S}$ to choose the actual signature. For the remaining values we consider two cases: (1) the adversary $\mathcal{A}$ does not corrupt p_D, and (2) $\mathcal{A}$ corrupts p_D. In Case 1, the shares of the corrupted players are simulated by $\mathcal{S}$ chosing uniformly random and independent values (if p_i and p_j are both corrupted then $\mathcal{S}$ makes sure that $s_{i,j}^{(i)} = s_{i,j}^{(j)}$ and $s_{j,i}^{(i)} = s_{j,i}^{(j)}$). The acusations and the answers can be easily simulated given the (simulated) view and the stategy of $\mathcal{A}$. In Case 2, as soon as $\mathcal{A}$ requests to corrupt p_D, $\mathcal{S}$ learns the input x_D. It is straight-forward to see that in that case $\mathcal{S}$ can simulate all the remaining values in the adverary's view (in fact, because no other player has input, $\mathcal{S}$ can simulate the full transcript including even the views of honest players). The honest-dealer correctness property is proved as follows: When the dealer is honest, then only values which lie on the actual polynomial $f(\cdot, \cdot)$ appear in the output of honest players. Moreover, every honest p_i holds all the signatures he should hold, as otherwise p_i would have complained in Step 5 and exposed the inconsistency. Therefore, the output will be a d-sharing of the dealers value. $\square$

Lemma 6. *Protocol* Reconstruct *invoked to the* $\{\mathcal{F}_{BC}, \mathcal{F}_{SIG}\}$*-hybrid model outputs (the same)* $y \in \mathbb{F}$ *towards every player. Furthermore, if* $d < n - t$ *(where* t *is the number of corrupted players) and the input is a* d*-consistent sharing of some* s*, then* $y = s$*.*

Proof. (sketch) As the output is decided based on values which are announce by $\mathcal{F}_{\text{BC}}$, all players output the same value y. Furthermore, when $d < n-t$ then there are at least $t+1$ honest players. When additionally the input is a d-consistent sharing then the values which the honest players have signed uniquely define all the share-polynomials $g_i(\cdot)$. As $\mathcal{F}_{\text{SIG}}$ never verifies as valid a signature of a value which was not signed by the corresponding player, the adversary cannot announce a polynomial other that $g_i(\cdot)$ for any $p_i \in \mathcal{P}$. Hence, every corrupted p_i either announces the correct value or is disqualified. However, the honest players always announce the correct values, hence the correct polynomials are interpolated. Because there are at least $d+1$ honest players there will always be at least $d+1$ values to interpolate the correct $g'(\cdot)$ and recover the shared value. □

Lemma 7. *Protocol* Broadcast *perfectly t-securely realizes the functionality $\mathcal{F}_{BC}$ in the $\{\mathcal{F}_{UBC}, \mathcal{F}_{SIG}\}$-hybrid model, for $t \leq n/2$.*

Proof. (sketch) By inspection of the protocol on can verify that the preconditions of Lemma 2 are satisfied for every value which is broadcasted. Therefore we can use $\mathcal{F}_{\text{UBC}}$ for broadcasting values. The security of our protocol is argued as follows: First observe that in any point of the protocol, if the simulator $\mathcal{S}$ learns x_s, then for any strategy of the adversary $\mathcal{A}$, $\mathcal{S}$ can perfectly simulate all the remaining values in the transcript, namely the values which should still appear in the view of $\mathcal{A}$ and the values appearing in the view of honest players.[8] We consider three cases: **Case 1.** The sender p_s is correct until the end of (the simulated) HD-Share: as soon as (the simulation of) HD-Share is complete, the simulator allows p_s to give his input to $\mathcal{F}_{\text{BC}}$, receives x_s as his output and uses it to simulate the full transcript. **Case 2.** The adversary corrupts p_s in (the simulated) HD-Share (this implies that at most $t-1$ players in $\mathcal{P}\setminus\{p_s\}$ can be corrupted): the simulator does not invoke $\mathcal{F}_{\text{BC}}$ yet, but learns x_s from the adversary; this way $\mathcal{S}$ can simulate the full transcript; denote by y the output of some (simulated) honest player; $\mathcal{S}$ instructs p_s to send y to the $\mathcal{F}_{\text{BC}}$ in the ideal evaluation. **Case 3.** The adversary corrupts t of the players in $\mathcal{P}\setminus\{p_s\}$ during (the simulated) HD-Share: For as long as at most $t-1$ players are corrupted, Lemma 5 ensures that the simulator can perfectly simulate without knowledge of x_s. At the point when the adversary requests to corrupt the t-th player, the simulator knows that the output of Broadcast will be x_s (since $\mathcal{A}$ can no loger corrupt p_s); $\mathcal{S}$ allows p_s to give his input to $\mathcal{F}_{\text{BC}}$, and, as in Case 1, $\mathcal{S}$ receives x_s and can now simulate the full transcript. □

Lemma 8. *There exists no protocol which computationally t-securely realizes the functionality $\mathcal{F}_{BC}$ for $t > n/2$. The statement holds also for statistical security.*

Proof. To arrive at a contradiction, assume that there exists a computationally (resp. statistically) t-secure SFE protocol Π. Wlog, assume that p_s uses Π to broadcast a uniformly random x_s. For every round i, protocol Π implicitly assigns to every set $\mathcal{P}' \subseteq \mathcal{P}$ a probability $\Pr_{\mathcal{P}',x_s,\Pi,i}$, which is the probability of the best efficient adversary corrupting $\mathcal{P}'$ to output x_s based only on her view on Π up to round i. For all $\mathcal{P}' \subseteq \mathcal{P}\setminus\{p_s\}$ this probability is negligible if i is the first round of Π and overwhelming if i is the last round of Π. As the total number of rounds in Π is polynomial, for each $\mathcal{P}' \subseteq \mathcal{P}\setminus\{p_s\}$ there exists a round $i_{\mathcal{P}'}$ where this probability from negligible becomes noticeable, i.e., not negligible. The adversary corrupts the set $A \subseteq \mathcal{P}\setminus\{p_s\}$ with $|A| = t-1$ such that $i_A = \min\{i_{\mathcal{P}'} \mid \mathcal{P}' \subseteq \mathcal{P} \ \wedge \ |\mathcal{P}'| \leq t-1\}$. In round i_A, the adversary gets the values which are sent to corrupted players and runs the best (efficient) protocol for the players in A to compute x_s; denote be x' the ouput of this protocol (by our assumption, $x' = x_s$ with noticeable probability). Let $\mathbb{F}_{1/2}$ denote the set of first $|\mathbb{F}|/2$ (in any ordering) elements in $\mathbb{F}$. If $x \in \mathbb{F}_{1/2}$ then the adversary acts as a passive adversary (i.e., all corrupted players are instructed to correctly execute their protocol). Otherwise, i.e., if $x \notin \mathbb{F}\setminus\mathbb{F}_{1/2}$, then the adversary actively corrupts p_s and forces all the actively corrupted players to crash before sending any message in round i; as $|\mathcal{P}\setminus A| \leq t-1$, with overwhelming probability the output of the honest players will be in $\mathbb{F}\setminus\{x\}$. With this strategy the adversary achieves that when $x \notin \mathbb{F}\setminus\mathbb{F}_{1/2}$ the output of Π is different than x_s with noticeable probability. However the simulator cannot simulate this behaviour as he has to decide whether or not to corrupt p_s independent of p_s's input. □

[8] As in the case of Lemma 4 the $\mathcal{F}_{\text{SIG}}$ functionality guarantees that the signatures can be simulated.