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 $t < n$.

Since its invention in [LSP82], broadcast has been used as a primitive in numerous multi-party protocols making it one of the fundamental primitives in the distributed-protocols literature. The security of these protocols is analyzed in a model where a broadcast primitive which behaves in an ideal way is assumed. Clearly, a definition of broadcast should allow for secure composition, namely, it should be secure to replace an assumed broadcast primitive by a protocol satisfying this definition. Following recent cryptographic reasoning, to allow secure composition the ideal behavior of broadcast can be described as an ideal functionality, and a simulation-based definition can be used.

In this work, we show that the property-based definition of broadcast does not imply the simulation-based definition for the natural broadcast functionality. In fact, most broadcast protocols in the literature do not securely realize this functionality, which raises a composability issue for these broadcast protocols. In particular, 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 who intend 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

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

H. Gilbert (Ed.): EUROCRYPT 2010, LNCS 6110, pp. 466–485, 2010.

underlying broadcast protocol. Informally, broadcast allows a sender to distribute his input among a set of players, such that every player gets the same value, even if the sender is dishonest.

1.1 Summary of Known Results

Broadcast was introduced by Pease, Shostak, and Lamport [LSP82] who showed that an adversary who can corrupt up to t players can be tolerated for perfectly secure Broadcast if and only if $3t < n$. This model has been extensively studied [DFF$^+$82, TPS87, FM88, CW89, BGP89, BDDS92, GM93] and protocols with optimal resiliency and complexity (communication and computation) polynomial in the number of players were suggested.[1] Other solutions [DS82, PW92] considered a setting where a setup allowing digital signatures is available, and showed that Broadcast tolerating an arbitrary number of cheaters ($t < n$) is possible. The suggested protocols are polynomial in the number of players and are as secure as the underlying signature scheme.[2]

Recently, Lindell, Lysyanskaya, and Rabin [LLR02] proved that, unless unique session identifiers are available, the bound $t < n/3$ is necessary for feasibility of concurrently composable Broadcast, even when a setup allowing digital signatures is given. To the positive side, they showed that when unique session IDs are available, then the protocols which achieve Broadcast and use signatures for authentication, e.g., [DS82, PW92] can be be transformed to concurrently composable Broadcast protocols.

1.2 Property-Based vs. Simulation-Based Definition

Intuitively, one could think of broadcast as a megaphone given to the sender, which every player can hear. More formally, this megaphone can be modeled as a functionality (in the sense of [Can00, Can01]), which receives an arbitrary message from the sender, and forwards this message to all players. The goal of a broadcast protocol is to realize this functionality, in the sense that in any context the abstract broadcast functionality can safely be replaced by the broadcast protocol. However, in the big body of broadcast literature, protocols are not proven to securely realize the above functionality; rather, they are shown to satisfy the following properties:

- Consistency: There exists some y such that every player outputs y.
- Validity: If the sender is honest and has input x then $y = x$.
- Termination: For every honest player the protocol terminates after a finite number of rounds.

Of course, the hope is that these properties imply security of a broadcast protocol in a simulation-based sense (i.e., any protocol satisfying these properties is expected

[1] Many of these protocols are actually Consensus protocols, from which a Broadcast protocol can be build by having the sender send his input to everybody and then invoke Consensus on the received values.

[2] In fact, feasibility of Broadcast for $t < n$ when a setup is available was also proved in [LSP82] but the suggested protocol has exponential communication complexity.

to securely realize the above broadcast functionality). However, this is not the case, as the property-based definition has a major flaw: the validity condition does not take into account the point in time when the sender gets corrupted. In particular, the definition does not rule out that the adversary can corrupt the sender *depending on the message which the sender intends to broadcast*. In fact, a broadcast protocol can satisfy the above three properties, and still allow the adversary to *first* learn the sender's message, and *then* to decide whether or not to corrupt the sender and make him broadcast a different message. This clearly contradicts the simulation-based definition, as well as the intuition with the megaphone. We stress that it is perfectly legal that the adversary can change the broadcasted message by corrupting the sender, and also it is perfectly legal that she learns the broadcasted message; however, it is counter-intuitive that she can first learn the message, without corrupting the sender, and then still be able to corrupt the sender and change it.

We give two examples to demonstrate the relevance of this problem: First, consider the following process for 10 players: Each player p_i ($i = 1, \ldots, 10$) in turn chooses a bit $b_i \in_R \{0, 1\}$ uniformly at random and announces it using ideal broadcast (e.g., using a megaphone), i.e., first p_1 selects $b_1 \in_R \{0, 1\}$ and announces it, subsequently p_2 selects $b_2 \in_R \{0, 1\}$ and announces it, etc. Consider an adversary who can corrupt at most three of the players, and her goal is to have only 1's broadcasted. Clearly, the probability that the output sequence consists only of 1's is at most 2^{-7}, as each of the seven bits chosen by the honest players are 1 with probability $1/2$. However, when we replace the ideal broadcast with some broadcast protocol satisfying the above three properties, then the adversary might be able to bring this probability to $46 \cdot 2^{-9}$, which is more than ten times bigger. She can achieve this by only corrupting those players p_i who intend to broadcast 0. With the mentioned probability, there are at most three such players, and the adversary can corrupt each of them and make them broadcast 1.

A more cryptography-related example is the following: Consider a prover p who uses the Fiat-Shamir (interactive) protocol to publicly prove to n players (verifiers) that he knows the square root of some publicly known y (in an RSA group). In order to do that, p executes one round of the Fiat-Shamir protocol with each verifier p_i in sequence. All executions are public, in the sense that all the messages are exchanged using ideal broadcast. Each verifier accepts if all rounds are accepting. Assume that the adversary can corrupt up to $t = n/2$ of the verifiers. Then in this protocol the probability that a malicious prover can make the players accept when he does not know the square root is negligible in n. However, along the lines of the above example, when the ideal broadcast is replaced by a broadcast protocol satisfying the above properties, a malicious prover might be able to corrupt only those verifiers who intend to challenge the bit the prover is not prepared to, which allows a malicious prover to cheat with probability $1/2$.

1.3 Broadcast in the Literature

As mentioned in the previous section, the big body of broadcast protocols in the literature are proven secure with respect to the mentioned properties, rather than with respect to a broadcast functionality. This would be only a minor issue if these protocols would securely realize the broadcast functionality. However, in the following we show that (at least most of them) fail to do so.

Most broadcast protocols in the literature [LSP82, DS82, BPW91, PW92, BHR07] proceed as follows:[3] First the sender sends the message to the players, possibly along with a signature; then, the players try to establish a consistent view on the sender's input. Obviously, any protocol following this approach cannot be secure against an adaptive adversary: Unless some kind of simultaneous multi-send assumption on the communication channels is made (see below), some corrupted player can happen to be the first to receive the message from the sender, and depending on this message, the adversary can decide whether or not to corrupt the sender (and change the message to be broadcast). Clearly, this behavior is not allowed when the above mentioned broadcast functionality is used, because as soon as some corrupted player receives (from the functionality) the broadcasted value, it is guaranteed that the honest players will also receive it (the functionality also sends it to them). Note that we do not need to assume a fully rushing adversary for the above behavior; we simply do not exclude that some corrupted player might get the message first, before it is sent to other players.

Note that many broadcast protocols can apparently be turned secure when the network offers a *simultaneous multi-send* operation. Such an operation is *atomic* and allows the sender to distribute an n-ary vector such that every player p_i receives the i-th component of the vector. More precisely, the operation is atomic in the sense that as soon as some player obtains some information about his component, then all other player must be guaranteed to receive their respective component as well. Such a network-operation is of course quite a strong assumption. Indeed, assuming such an operation implies that a player who honestly behaves at a specific point in time can broadcast a message (by multi-sending it) which seems to be closer to a broadcast channel than to a point-to-point communication network. In fact, the Universal Composition framework [Can01] which is the most widely accepted framework for arguing about the security of protocols, explicitly excludes such a simultaneous multi-send assumption. Furthermore, for broadcast protocols using signatures and tolerating $t \geq n/3$ [DS82, PW92], even this assumption does not help, as still the adversary can learn the message in the first phase of the protocol, and make the broadcast fail afterwards by corrupting the sender and introducing signatures for different messages.[4]

The major problem is that in the broadcast literature, protocols are proven secure with respect to properties, but in the cryptographic protocols literature (VSS, MPC, etc), protocols are proven secure in a hybrid world with access to an ideal broadcast functionality (e.g., "secure-channels model with broadcast"). The security of these cryptographic protocols, when the broadcast functionality is instantiated with some broadcast protocol from the literature, is doubtful.

1.4 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

[3] This also includes any broadcast protocol which first has the sender send his input to everybody and then invokes a Consensus protocol, e.g. [DFF$^+$82, TPS87, FM88, CW89, BGP89, BDDS92, GM93], on the received values.

[4] This would essentially correspond to a Broadcast functionality with partial fairness and unanimous abort [GL02].

in the secure-channels model with perfect security. We also describe a weaker functionality which is realized by the known broadcast protocols, and, under certain conditions, can instantiate a broadcast primitive within a high level protocol. These conditions are, for example, satisfied by the VSS protocol from [BGW88]. Note however, that in many of the known protocols which assume broadcast, e.g., [CDD$^+$99], these conditions are not guaranteed. Hence, if one would be willing to make a compromise and accept the weaker functionality as the ideal functionality for broadcast, then he would need to (re-)prove the security of such protocols with this functionality in mind.

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 in 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 adversary might first learn 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, a 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.

1.5 Comparison with Previous Work

The idea to use VSS to get more out of a broadcast protocol was used in the context of simultaneous broadcast [CGMA85, CR87, Gen95, Gen00, HM05]. However, the goal of these works is to satisfy an additional property, namely to allow different parties to broadcast values in parallel while guaranteeing mutual independence of the broadcast values. This does not imply simulation-based composable security. Recently, Hevia [Hev06] proposed a simultaneous broadcast protocol which he proved to be universally composable. 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, as we show here, in general is not the case. In fact, the protocol in [Hev06] employs the verifiable secret-sharing scheme from [CDD$^+$99], which in turn employs some broadcast primitive which (hopefully) securely composes in the secure-channels model for $t < n/2$. To our knowledge, our work is the first to present such a composable broadcast protocol.

In [LLR02] a broadcast protocol for $t < n$ was described, which is concurrently composable when unique session IDs are available. This result does not contradict ours, as it implicitly assumes that the players can simultaneously multi-send messages (c.f. [LLR02, Sect. 2.1]). In fact, this protocol is a "transformation from almost any Broadcast protocol to a protocol that concurrently composes". Because all known broadcast protocols have the above mentioned problem, also this construction has it when run in a model without simultaneous multi-send.

2 The Model

We consider the well-known secure channels model introduced in [BGW88, CCD88], where the players in $\mathcal{P} = \{p_1, \ldots, p_n\}$ are connected by a complete network of bilateral secure channels. In such a network the only way that the adversary can get information on a sent message is by corrupting the sender or the receiver.

2.1 Synchronous Communication (No Multi-send)

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 every player can send a message to every other player.

There are several variations of the synchronous channels model suggested in the literature. In some works [Nie03, LLR02] it is implicitly assume that honest players can simultaneously multi-send messages, i.e, simultaneously send messages to several recipients. Such a multi-send operation is atomic and guarantees that for a sender who is honest upon sending, if one of the messages is delivered to its recipient then all the messages will be delivered (unchanged) to the corresponding recipients. As we already pointed out, such a simultaneous multi-send operation is a quite strong assumption on the communication network.

In this work we only assume bilateral communication and, in particular, *we do not assume simultaneous multi-send*: a player who is instructed to send a message to more than one players can do so one player at a time. This is consistent with the formulation of [Can01] where it is required that the processes are activated in turns, where at any point only a single process can be active, and it can send a message to one other process which becomes now the active process, and so on.

2.2 The Adversary

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 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*.

There are several adversarial models in the literature which restrict the power of the adversary. For example a *static* adversary is one who chooses the players to corrupt at the beginning of the protocol.

In this work we do not put any such restrictions on the adversary's corruption power. In particular, the assumed adversary is *adaptive*, i.e., in contrast to a static adversary, she 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. Because no simultaneous multi-send is assumed, it might happen that some corrupted player receives his message from an honest player p in some round, before p has finished sending all his messages for this round.[5] If this happens, the adversary can corrupt p after learning the message which was sent to the corrupted player, and force him change the remaining messages which he intended to send in that round.

2.3 Security Definition

Following the [Can00, Can01] 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 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 also sees 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 $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. 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 $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$).

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

[5] Note that if one would assume a rushing adversary then this would be the case "by definition".

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, Can01].

3 Perfect Security (No Setup)

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 the 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 some of our ideas.

The ideal functionality for broadcast $\mathcal{F}_{\text{BC}}$ 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-type version of this functionality can be found in the full version of this paper.

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

1. p_s sends his input x_s to the functionality $\mathcal{F}_{\text{BC}}$.
2. $\mathcal{F}_{\text{BC}}$ sends x_s to every $p \in \mathcal{P}$.

To show that the above functionality is not realized by known protocols, we observe that known broadcast protocols have the following pattern: 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 sending his input to everybody and then invoke a consensus protocol on the received value, e.g., [CW89, BGP89, BDDS92], are of the above type. However, even the protocols where the second phase is not a self-contained consensus protocol, e.g., the broadcast protocols from [DS82, PW92], 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, there is a good probability that a corrupted player is the first to receive the input x_s from p_s and the adversary can, depending on the received value, decide whether or not to corrupt the sender p_s (and possibly change the broadcasted value).[6] 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 $\mathcal{F}_{\mathrm{BC}}$) by having p_s share his input x_s, and, subsequently, having the players publicly reconstruct the sharing.

It might look that we are done, as one could use the perfectly secure VSS from [BGW88] to achieve broadcast. But this is not quite true. The reason is that [BGW88] (and all other known VSS schemes with perfect security) use broadcast as a primitive. If we instantiate this primitive by one of the known broadcast protocols then we can no longer argue about the security of the full construction using composition. Nevertheless, we show in the following that replacing all broadcast invocations in the [BGW88] VSS scheme by executions of the [BGP89] broadcast protocol[7] does not cause any loss of security; we denote this VSS scheme by $\mathrm{VSS}_{\mathrm{BGW}}^{(\mathrm{BGP})}$.

The security of $\mathrm{VSS}_{\mathrm{BGW}}^{(\mathrm{BGP})}$ is argued in two steps. In a first step, we show that although the [BGP89] broadcast protocol, denoted in the following as $\mathrm{BC}_{\mathrm{BGP}}$, does not securely realize $\mathcal{F}_{\mathrm{BC}}$, it does realize a weaker functionality, denoted as $\mathcal{F}_{\mathrm{UBC}}$ (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 $\mathcal{F}_{\mathrm{BC}}$ by $\mathcal{F}_{\mathrm{UBC}}$ without loosing security.

The functionality $\mathcal{F}_{\mathrm{UBC}}$ is described in the following. Intuitively, the difference to the functionality $\mathcal{F}_{\mathrm{BC}}$ is that $\mathcal{F}_{\mathrm{UBC}}$ 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 $\mathcal{F}_{\mathrm{UBC}}$

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

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

Proof. (sketch) As shown in [BGP89], the protocol $\mathrm{BC}_{\mathrm{BGP}}$ satisfies the property-based definition of broadcast (i.e., it satisfies validity, consistency, and termination). We show

[6] In fact, if one assumes a rushing adversary then she can, by definition, always perform such an attack, as she first learns the messages sent to corrupted recipients.

[7] 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.

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 no input, knowing x_s allows $\mathcal{S}$ to perfectly simulate all the messages sent by honest players.[8] $\mathcal{S}$ 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 $\mathcal{F}_{\text{UBC}}$. □

Remark 1. One can verify that most broadcast protocols in the literature, including those that assume a setup and tolerate $t < n$ corrupted players, securely realize the ideal functionality $\mathcal{F}_{\text{UBC}}$. The proof is along the lines of the above proof. One might even be willing to make a compromise and accept this functionality as a tight description of what one would expect from broadcast. However, we point out that this functionality allows the counter-intuitive behavior explained in the introduction. Furthermore, the security of protocols which assume broadcast should be (re-)analyzed with this functionality in mind.

For the second step, we show that if a protocol Π (which assumes broadcast) satisfies an appropriate pre-condition, then it is safe to instantiate broadcast in Π by calls to $\mathcal{F}_{\text{UBC}}$. The pre-condition 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.[9]

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

[8] 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.

[9] 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.

$\text{Exec}_{\mathcal{Z},\mathcal{A}',\Pi'} \equiv \text{Exec}_{\mathcal{Z},\mathcal{A},\Pi}$. 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)[10] 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. *Protocol* $VSS_{BGW}^{(BGP)}$ *perfectly* t*-securely realizes the functionality* $\mathcal{F}_{BC}$ *for* $t < n/3$.

Remark 2. Although replacing $\mathcal{F}_{\text{BC}}$ by $\mathcal{F}_{\text{UBC}}$ did not affect the security of [BGW88] VSS, this is not necessarily true for other protocols using broadcast. For example, the VSS in [CDD$^+$99] does not satisfy the pre-condition of Lemma 2. In fact, in [CDD$^+$99] uniformly random values are broadcasted. As demonstrated in the examples given in the introduction, broadcasting random values by a protocol which only securely realizes $\mathcal{F}_{\text{UBC}}$ can have unexpected results. In fact, it is unclear whether or not [CDD$^+$99] is secure if we instantiate the assumed broadcast-channel by calls to $\mathcal{F}_{\text{UBC}}$.

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]). *For* $t \geq n/3$ *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 $\mathcal{F}_{\text{BC}}$ satisfies the given three properties.

Corollary 2. *For* $t \geq n/3$ *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. *When* $t \geq n/3$ *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.*

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

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, denoted in the following as Π_{DS}, 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]).

Analogously to the case of perfect security, our approach proceeds in two steps, 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, on the one hand, all known protocols for statistical VSS are only secure for $t < n/2$ which is stronger than $t \leq n/2$. On the other hand, these protocols do not satisfy the pre-condition of Lemma 2. Before describing how to overcome these difficulties 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 possible by the definition of $\mathcal{F}_{\text{SIG}}$.[11]

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 realizing $\mathcal{F}_{\text{BC}}$, 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

[11] The idea of using [DS82] with i.t. secure signatures to get an i.t. secure broadcast protocol appears also in [PW92, Fit03].

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 or i.t. security, respectively.

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]. The main difference from a standard sharing protocol is that the correctness of the output-sharing is guaranteed only when the dealer is honest until the end of the protocol. 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).

Lemma 5. *Protocol* HD-Share *invoked in the $\{\mathcal{F}_{BC}, \mathcal{F}_{SIG}\}$-hybrid model achieves the following: The view of any d-adversary attacking the protocol can be perfectly simulated (privacy);*[12] *When the dealer is honest until the end of* HD-Share *then the output is a d-sharing of s (honest-dealer correctness). Furthermore, in all calls to $\mathcal{F}_{BC}$, the adversary "knows in advance" the value to be broadcasted.*

Proof. (sketch) Clearly the adversary "knows in advance" all the values to be broadcasted, as these are accusations and replies which might only occur if at least one of the disputing players is corrupted. The privacy of HD-Share can be argued along the lines of [CDD$^+$99]. Nevertheless we sketch how the simulator $\mathcal{S}$ simulates the view of the adversary $\mathcal{A}$: The simulation of the signatures is trivial, as the functionality $\mathcal{F}_{\text{SIG}}$

[12] This ensures that no information about s leaks to a d-adversary.

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 $s^{(i)}_{i,j}, s^{(i)}_{j,i}, sig_{p_D}(s^{(i)}_{i,j})$, and $sig_{p_D}(s^{(i)}_{j,i})$.
   (b) p_i broadcasts a complaint if any of the vectors $\big(s^{(i)}_{i,1},\ldots,s^{(i)}_{i,n}\big)$ and $\big(s^{(i)}_{1,i},\ldots,s^{(i)}_{n,i}\big)$ is not d-consistent or if 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 $s^{(i)}_{i,j}$ to p_j along with his signature $sig_{p_i}(s^{(i)}_{i,j})$ and the dealer's signature $sig_{p_D}(s^{(i)}_{i,j})$.
   (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 $(s^{(i)}_{i,j}, sig_{p_D}(s^{(i)}_{i,j}), sig_{p_i}(s^{(i)}_{i,j}))$. 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 adopts $s^{(j)}_{i,j}$ as the value he should have received in Step 2a; 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 $s^{(j)}_{i,j}$ from some p_j in Step 2 which is inconsistent with his own view of $s_{i,j}$, i.e., $s^{(j)}_{i,j} \neq s^{(i)}_{i,j}$, and if so, broadcasts $\big(s^{(i)}_{i,j}, s^{(j)}_{i,j}, sig_{p_D}(s^{(i)}_{i,j}), sig_{p_D}(s^{(j)}_{i,j})\big)$; every player verifies that $s^{(j)}_{i,j} \neq s^{(i)}_{i,j}$ and that the signatures are valid and if so p_D is disqualified.[a]

[a] Recall that 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.

allows $\mathcal{S}$ to choose the actual signature. As long as $\mathcal{A}$ does not corrupt p_D, the simulator proceeds as follows: whenever $\mathcal{A}$ requests to corrupt some p_i, $\mathcal{S}$ creates a simulated view for p_i (up to the current simulated round) by choosing all the values in p_i's view uniformly at random except those that have already appeared in the view of $\mathcal{A}$ (e.g., if p_j has been already corrupted then the values $s^{(i)}_{i,j} = s^{(j)}_{i,j}$ and $s^{(i)}_{j,i} = s^{(j)}_{j,i}$ have been already given to the adversary). Note that, because the sharing polynomial is of degree d, from the point of view of the d-adversary $\mathcal{A}$ the simulated views are distributed as in a real run of the protocol HD-Share. If at some point $\mathcal{A}$ requests to corrupt p_D, then at that point $\mathcal{S}$ learns p_D's input s and, can simulate the view of all the remaining players while making sure that all simulated values are consistent with some degree-d polynomial $f'(\cdot,\cdot)$ with $f'(0,0) = s$. Honest-dealer correctness is proved as follows:

When the dealer is honest at the end of HD-Share, 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 dealer's value. □

To reconstruct a sharing the protocol Reconstruct is invoked (see below). The idea is the following: every p_i broadcasts 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 and the number of corrupted parties, 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 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 $y = s$.*

Proof. (sketch) As the output is decided based on values which are agreed upon using $\mathcal{F}_{\text{BC}}$, 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 $g_i(\cdot)$. Because $\mathcal{F}_{\text{SIG}}$ never verifies as valid a signature on a value which was not signed by the corresponding player, the adversary cannot announce a polynomial other than $g_i(\cdot)$ 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. □

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), the decision whether or not to corrupt p_s has to be taken independently of x_s, which is a behavior that can be easily simulated.

In the above, we managed to tweak the VSS protocol from [CDD$^+$99] (which is secure if and only if $t < n/2$) so that we can use it for broadcast when $t \leq n/2$. However, 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 such a modification could potentially give an advantage to the adversary. But this might only happen in case the adversary has not corrupted p_s by the end of HD-Share, as otherwise she knows all the outputs by then. However, even in this bad case, because p_s is honest until the end of HD-Share, by the time the dummy step is executed the output is already fixed to x_s, and the adversary cannot change it even with access to the full transcript. For completeness we include a description of our broadcast protocol and state its achieved security. The proof of the lemma can be found in 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. 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 $\leq n/2$.*

Proof. (sketch) By inspection of the protocol one can verify that the pre-conditions of Lemma 2 are satisfied for every value which is broadcasted, hence using $\mathcal{F}_{\text{UBC}}$ for broadcasting values in protocol Broadcast is as secure as using $\mathcal{F}_{\text{BC}}$. Therefore, it suffices to argue the security of Broadcast in the $\{\mathcal{F}_{\text{BC}}, \mathcal{F}_{\text{SIG}}\}$-hybrid model. We sketch the simulator $\mathcal{S}$ for a given adversary $\mathcal{A}$. During the execution of HD-Share, the simulator behaves as the simulator in the proof of Lemma 5. In the subsequent steps, if $\mathcal{A}$ has already corrupted p_s before the end of the simulated run of HD-Share, then at that point $\mathcal{S}$ has learned the sender's input x_s and can simulate the remaining transcript as in the proof of Lemma 5 ($\mathcal{S}$ can clearly simulate all the messages exchanged in Steps 2 and 3 as they appear in the transcript of HD-Share). Otherwise, i.e., if by the end of the simulated run of HD-Share the adversary $\mathcal{A}$ has not requested to corrupt p_s, then

$\mathcal{S}$ allows for the invocation of $\mathcal{F}_{\text{BC}}$, where p_s gives his input x_s; $\mathcal{S}$ learns x_s from $\mathcal{F}_{\text{BC}}$ (as the output of any corrupted player) and can, same as before, simulate the remaining transcript. □

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 the above protocol 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 noticeable information on the dealers input, before the honest players in $\mathcal{P} \setminus \{p_s\}$ jointly learn noticeable information. Depending on this information the adversary can corrupt the sender and, with overwhelming probability, change the output to some other value. However this behavior cannot be simulated.

Lemma 8. *There exists no protocol which computationally t-securely realizes the functionality $\mathcal{F}_{BC}$ for $t > n/2$, not even in the $\{\mathcal{F}_{SIG}\}$-hybrid model. The statement holds also for statistical security.*

Proof. To arrive at a contradiction, assume that there exists a computationally (resp. statistically) t-secure Broadcast protocol Π. Wlog, assume that p_s uses Π to broadcast a uniformly random $x_s \in_R \mathbb{F}$. 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 in 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, denoted as $i_{\mathcal{P}'}$, 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) strategy to compute x_s on input the view of the players in A; denote by x' the output 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 $x' \in \mathbb{F} \setminus \mathbb{F}_{1/2}$, then the adversary actively corrupts p_s and forces all the actively corrupted players to crash before sending any message in round i; as $|\mathcal{P} \setminus A| \leq t - 1$ we know that $i_{\mathcal{P} \setminus A} \geq i_A$, hence, because the players in $\mathcal{P} \setminus A$ do not get the messages from round i_A, with overwhelming probability the output of the honest players will be in $\mathbb{F} \setminus \{x_s\}$. With this strategy the adversary achieves that when $x_s \in \mathbb{F} \setminus \mathbb{F}_{1/2}$, then the output of the honest players in Π is different than x_s with noticeable probability. However the simulator cannot simulate this behavior as he has to decide whether or not to corrupt p_s and change the output independent of x_s. □

5 Other Models

We presented our solutions in the secure-channels model, because in this model it is clear that the only way the adversary can learn a transmitted message is by corrupting the sender or the receiver (after the message has been received). In particular, this implies that for a message sent to the trusted party/functionality, as long as the sender is honest, the simulator cannot learn the sent message before the functionality learns it.

In the authenticated-channels model (without privacy), the same composability issue appears as the one we deal with in this work. However, as it is typically the case, one could pretend to solve this issue by giving the simulator additional power on the communication network. For example, if the simulator is allowed to read the sent message, delete it from the channel, and then corrupt the sender and re-send the message, then the above problem disappears "by definition". It is arguable, however, how consistent such a model is with the synchronicity assumption on the communication network. Furthermore, when defining such an authenticated communication model which eliminates the composability issue presented in this work, one has to keep in mind that the described protocols typically are not composable when the authenticated-channels are replaced by secure-channels.

Also, in the case of asynchronous communication, the same problem appears. Take for example the asynchronous secure-channels model as defined in [BCG93, BKR94]. As in the synchronous case, unless we assume some kind of asynchronous atomic multisend, when a player p is instructed to send a message to several other players,[13] then it might happen that the adversary first learns the message by corrupting one of the receivers, and still is able to corrupt p and change it. In fact, as already mentioned, this is the case in the UC framework [Can01]. Clearly this behavior cannot be simulated in the ideal world. As in the synchronous case, one might be willing to make a compromise and accept an asynchronous version of $\mathcal{F}_{\text{UBC}}$ to be the desired ideal functionality for broadcast.

6 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 the property-based definition of broadcast does not imply the simulation-based definition for the natural broadcast functionality. Furthermore, we showed that most known broadcast protocols do not realize this functionality in the secure-channels model. As a result, if one replaces the broadcast invocations in any of the known multi-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.

[13] Observe that this is the way in which most asynchronous broadcast protocols start, e.g., [Bra84].

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 $t \leq n/2$. Furthermore, we described a weaker ideal functionality for broadcast which is securely realized by the known protocols but achieves less than what one expects from a broadcast protocol. Of course, one might be willing to make a compromise and accept this as the desired ideal functionality for broadcast. But in that case, all known protocols should be (re-)analyze with this weaker ideal functionality in mind.

References

[BCG93] Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: STOC 1993, pp. 52–61 (1993)

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

[BGP89] Berman, P.J., Garray, J., Perry, J.: Towards optimal distributed consensus. In: FOCS 1989, pp. 410–415 (1989); Full version in Computer Science Research (1992)

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

[BHR07] Beerliova-Trubiniova, Z., Hirt, M., Riser, M.: Efficient Byzantine agreement with faulty minority. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 393–409. Springer, Heidelberg (2007)

[BKR94] Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: PODC 1994, pp. 183–192. ACM, New York (1994)

[BPW91] Baum-Waidner, B., Pfitzmann, B., Waidner, M.: Unconditional Byzantine agreement with good majority. In: Jantzen, M., Choffrut, C. (eds.) STACS 1991. LNCS, vol. 480, pp. 285–295. Springer, Heidelberg (1991)

[Bra84] Bracha, G.: An asynchronou [(n-1)/3]-resilient consensus protocol. In: PODC 1984, pp. 154–162 (1984)

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

[Can01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)

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

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

[CDD$^+$99] Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)

[CGMA85] Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: FOCS 1985, pp. 383–395 (1985)

[CR87] Chor, B., Rabin, M.O.: Achieving independence in logarithmic number of rounds. In: PODC 1987, pp. 260–268 (1987)

[CW89] Coan, B.A., Welch, J.L.: Modular construction of nearly optimal Byzantine agreement protocols. In: PODC 1989, pp. 295–305 (1989); Full version in Information and Computation (1992)

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

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

[Fit03] Fitzi, M.: Generalized Communication and Security Models in Byzantine Agreement. PhD thesis, ETH Zurich (2003)

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

[FM88] Feldman, P., Micali, S.: Optimal algorithms for Byzantine agreement. In: STOC 1988, pp. 148–161 (1988)

[Gen95] Gennaro, R.: Achieving independence efficiently and securely. In: PODC 1995, pp. 130–136 (1995)

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

[GL02] Goldwasser, S., Lindell, Y.: Secure computation without agreement. In: Malkhi, D. (ed.) DISC 2002. LNCS, vol. 2508, pp. 17–32. Springer, Heidelberg (2002)

[GM93] Garay, J.A., Moses, Y.: Fully polynomial Byzantine agreement in t+1 rounds. In: STOC 1993, pp. 31–41 (1993)

[Hev06] Hevia, A.: Universally composable simultaneous broadcast. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 18–33. Springer, Heidelberg (2006)

[HM05] Hevia, A., Micciancio, D.: Simultaneous broadcast revisited. In: PODC 2005, pp. 324–333 (2005)

[KY84] Karlin, A., Yao, A.C.: Manuscript (1984)

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

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

[Nie03] Nielsen, J.B.: On Protocol Security in the Cryptographic Model. PhD thesis, BRICS (2003)

[PW92] Pfitzmann, B., Waidner, M.: Unconditional Byzantine agreement for any number of faulty processors. In: STACS 1992. LNCS, vol. 577, pp. 339–350 (1992)

[RB89] Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: STOC 1989, pp. 73–85 (1989)

[TPS87] Toueg, S., Perry, K.J., Srikanth, T.K.: Fast distributed agreement. SIAM J. Comput. 16(3), 445–457 (1987)