Open access • Proceedings Article • DOI:10.1145/1993806.1993832

Adaptively secure broadcast, revisited — Source link

Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, Hong-Sheng Zhou

Institutions: AT&T Labs, University of Maryland, College Park

Published on: 06 Jun 2011 - Principles of Distributed Computing

Topics: Atomic broadcast, Secure two-party computation and Cryptographic protocol

Adaptively Secure Broadcast, Revisited

Juan A. Garay* Jonathan Katz† Ranjit Kumaresan†
Hong-Sheng Zhou†

Abstract

We consider the classical problem of synchronous broadcast with dishonest majority, when a public-key infrastructure and digital signatures are available. In a surprising result, Hirt and Zikas (Eurocrypt 2010) recently observed that all existing protocols for this task are insecure against an *adaptive* adversary who can choose which parties to corrupt as the protocol progresses. Moreover, they prove an impossibility result for adaptively secure broadcast in their setting.

We argue that the communication model adopted by Hirt and Zikas is unrealistically pessimistic. We revisit the problem of adaptively secure broadcast in a more natural synchronous model (with rushing), and show that broadcast *is* possible in this setting for an arbitrary number of corruptions. Our positive result holds under a strong, simulation-based definition in the universal-composability framework.

We also study the impact of adaptive attacks on protocols for secure multi-party computation where broadcast is used as a sub-routine.

1 Introduction

Broadcast [14, 12] is a fundamental primitive in fault-tolerant distributed computing. It also serves as an important subcomponent of most multi-party cryptographic protocols. Indeed, cryptographic protocols are typically designed and analyzed under the assumption that a broadcast channel is available, but in almost any real-world scenario the broadcast channel needs to be realized using a broadcast protocol. Fortunately, known composition results (including, most powerfully, those within the *universal composability* (UC) framework [3]) imply that this approach is *sound*: namely, given a protocol Π proven secure under the assumption that a broadcast channel exists, and then instantiating the broadcast channel using a secure broadcast protocol bc, the composed protocol Π^{bc} is guaranteed to be secure when run over a point-to-point network.

The construction of broadcast protocols has a long history, starting from the seminal work of Pease, Shostak, and Lamport [14, 12] who showed that broadcast (and the closely related primitive known as consensus or Byzantine agreement) is possible if and only if the number of corrupted parties t is strictly less than $1/3$ of the total number of parties n. (Here and throughout the paper, we assume a synchronous model of communication.) This holds in the "plain model" with no additional setup, but it is known that the bound on the tolerable number of corrupted parties can be exceeded if a public-key infrastructure (PKI) and digital signatures are available to the parties.

*AT&T Labs – Research. Email: garay@research.att.com.

†Dept. of Computer Science, University of Maryland. Research supported by the US DoD/ARO MURI program, and the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001. Email: {jkatz,ranjit,hszhou}@cs.umd.edu.

In this setting, it is possible to construct broadcast protocols resilient to a computationally bounded adversary corrupting $t < n$ of the parties [9].[1] In what follows, we always have in mind this setting when we speak of protocols tolerating $t \geq n/3$ corrupted parties.

As far as questions of feasibility are concerned, broadcast had appeared to be a solved problem. A recent result by Hirt and Zikas [11] therefore came as quite a surprise. They studied the problem of designing broadcast protocols with security against *adaptive* adversaries who can choose which parties to corrupt during the course of the protocol (cf. [5]). Hirt and Zikas showed explicit attacks against all existing broadcast protocols when $t \geq n/3$ and, moreover, proved the *impossibility* of realizing adaptively secure broadcast with corruption threshold $t > n/2$. (They gave constructions of adaptively secure protocols for the regime $n/3 \leq t \leq n/2$.) Their work calls into question the feasibility of realizing adaptively secure multi-party computation (MPC) for $t > n/2$ in point-to-point networks.

A closer look at the Hirt-Zikas result shows that they make a very strong assumption regarding the adversary (or, alternately, a very weak assumption regarding the communication network): namely, they assume the adversary has the ability to corrupt parties *in the middle of a round*, in between sending messages to two other parties in the network. Specifically, their impossibility result crucially relies on the fact that the following sequence of events can occur when an honest party P sends its messages in some round:

1. The adversary (who has already corrupted some of the other players) receives the message(s) sent to it by P.
2. Based on this, the adversary then decides whether to corrupt P.
3. If the adversary corrupts P, it can then send messages of its choice (on behalf of P) to the remaining parties in the same round.

While the above is consistent with theoretical models for *asynchronous* cryptographic protocols, as well as some previous treatments of adaptive security in the synchronous setting (e.g., [3]), allowing such adversarial behavior seems unrealistically pessimistic: in the real world, implementing such an attack would require either an exceedingly fast adversary or an extremely slow network. A more realistic model of synchronous communication (see, e.g., [2]) is one in which messages sent by honest parties within any given round are delivered *atomically* to all other parties.[2]

Importantly, however, the attacks that were demonstrated by Hirt and Zikas [11] on existing broadcast protocols remain valid even if we assume atomic message delivery. Consider, for example, the authenticated broadcast protocol of Dolev and Strong [9] where, at a high level, in the first round the sender digitally signs and sends his message to all the other parties, while in subsequent rounds parties append their signatures and forward the result. Roughly, if any party ever observes valid signatures of the sender on two *different* messages then that party forwards both signatures to all other parties and disqualifies the sender (and all parties output some default message). The Hirt-Zikas attack against this protocol works as follows: a corrupted party P in the network waits to receive the initial message from the (uncorrupted) sender. If P likes the message sent by the

[1] With a different sort of initial setup, broadcast resilient to an unbounded adversary corrupting $t < n$ of the parties is also possible [16].

[2] We still allow *rushing*, meaning that corrupted parties may receive their messages in some round before having to send any of their own. This reflects the fact that corrupted parties can choose to delay their own communication. However, it seems unrealistic to assume that honest parties would delay sending any of their own messages.

sender then P runs the protocol honestly. If P does not like the message sent by the sender then P adaptively corrupts the sender, uses the sender's signing key to generate a valid signature on another message (in the next round), and thus ensures that the sender will be disqualified and the default message used.

While this outcome might be acceptable with respect to a *property-based* definition (since the sender is corrupted by the end of the protocol in the second case), the outcome is not something that should be possible with respect to a *simulation-based* definition (since corruption of the sender depends on the sender's initial input). Realizing the latter, stronger definition is a natural goal; moreover, a simulation-based definition is especially critical for broadcast which is typically used as a sub-protocol within some larger protocol.

Given that the Hirt-Zikas attack applies even when atomic message delivery is assumed, one might wonder whether their impossibility result holds in that model as well. Alternately, one may be willing to give up on "full" broadcast and hope that some weaker form of broadcast might be sufficient to achieve secure MPC for $t > n/2$. (Indeed, in the presence of a dishonest majority the standard definitions of secure MPC give up on guaranteed output delivery, so in particular secure MPC for $t > n/2$ does not imply broadcast for $t > n/2$.) These are the questions with which we concern ourselves in this paper.

1.1 Our Results

As our main result, we show that the Hirt-Zikas impossibility result does *not* apply in the synchronous model with atomic message delivery. That is, we show a construction of an adaptively secure broadcast protocol tolerating an arbitrary number of corruptions in this communication model. We prove security of our protocol within the UC framework [3], under the usual assumptions that a PKI and digital signatures are available. We stress that we require only a standard PKI where each party chooses their public key and all other parties know it; in particular, we do not require the stronger "registered public key" model considered in [1].

The main idea for avoiding the Hirt-Zikas attack is to design a protocol where the adversary does not learn the (honest) sender's message until agreement has already been reached; that way, the adversary must make its decision as to whether or not to corrupt the sender independently of the sender's input. This suggests the following two-stage approach: First, the signer broadcasts a *commitment* to its message; once agreement is reached, the signer then decommits. While this does prevent the above attack, it also introduces a new problem when we try to prove security, since the simulator must commit to the sender's message before knowing what the sender's message is! (Since the sender might still get corrupted in the middle of the protocol, it also does not work for the simulator to obtain the output of the broadcast functionality before starting the simulation.) This could be handled by using a universally composable commitment scheme (e.g., [6]), which satisfies even stronger properties, but we would prefer to avoid the stronger setup assumptions that are required for constructing universally composable commitments [6].

Instead, we show that a very weak form of commitment suffices to make the approach sound. Specifically, we use commitment schemes that (informally) are hiding and binding for an *honest* sender, but where binding can be (easily) violated by a *dishonest* sender. To see why this works, note that the only time binding is needed is when the adversary corrupts the sender *after* the sender has already committed to its message. Since the sender in that case was honest at the time the commitment was generated, the binding property holds and the adversary will not be able to change the committed value. On the other hand, the simulator can behave as a dishonest sender and

generate a commitment that it can later open to any desired value, and in particular to the sender's true input in case the sender remains uncorrupted until the end of the protocol. We show that commitment schemes with the desired properties can be constructed from one-way functions (which are, in turn, implied by digital signature schemes); thus, in summary, we obtain an adaptively secure, universally-composable broadcast protocol assuming a PKI and digital signatures.

We also study the impact of adaptive attacks on secure multi-party computation protocols (where broadcast is commonly used as a subcomponent), and establish the variants of broadcast that are needed in this setting. Interestingly, we show that the full functionality of broadcast is not needed in order to obtain secure MPC for $t \geq n/2$; instead, a weaker form of broadcast — which can be realized even in the Hirt-Zikas communication model — suffices.

1.2 Organization of the Paper

In Section 2 we present our network model and elaborate on simulation-based definitions of security. Section 3 defines various notions of broadcast, and contains our construction of adaptively secure broadcast. We discuss the consequences for adaptively secure multi-party computation in Section 4.

2 Preliminaries

2.1 Network Model

We consider a network with synchronous communication, where there is a set of n players (probabilistic polynomial-time Turing machines) $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ connected by point-to-point authenticated channels. Each round of the protocol proceeds as follows. The honest parties send their messages for that round, and these messages are received by all parties (both honest and corrupted). The adversary may then choose to corrupt additional players, and then it sends messages on behalf of the parties who were corrupted at the beginning of that round. (This models a *rushing* adversary.) When it is done, the adversary must then "advance the clock" to the next round. We allow the adversary to corrupt any $t < n$ of the parties, and to behave in an arbitrary ("Byzantine") manner.

We stress that our model is different from that considered by Hirt and Zikas [11], where in each round the honest parties' messages are first delivered *to the corrupted parties only* and then the adversary is allowed to corrupt additional parties and decide what messages to send on behalf of those parties to other honest players. In contrast, we assume that honest parties' messages are delivered "atomically", which is equivalent to assuming that adversarial corruption cannot occur in the time interval between when a message is sent and when it is received. We sometimes refer to our model as "atomic", and to the Hirt-Zikas model as "non-atomic".

2.2 Simulation-Based Security

We use a simulation-based definition of security, which is in line with work in the area of cryptographic-protocol design but which differs from most of the classical work on Byzantine agreement and broadcast. Simulation-based definition are formulated by defining an "ideal" version of some desired functionality that is implemented by a trusted third-party; a protocol is secure if the protocol "emulates" this ideal world no matter what the adversary does. One advantage of a simulation-based approach is that it simultaneously captures *all* the properties that are guaranteed by the ideal

world, without having to enumerate some list of desired properties. Simulation-based definitions are also useful for applying *composition theorems* that enable proving security of protocols that use other protocols as sub-routines.

We formulate our simulation-based definitions by presenting appropriate functionalities within the UC framework. We give a brief introduction to this model, and refer readers elsewhere for more details [3]. The basic entities involved are players $P_1, \ldots, P_n$, an adversary $\mathcal{A}$, and an "environment" $\mathcal{Z}$. The environment $\mathcal{Z}$ gives inputs to and receives outputs from all the players; it also interacts with $\mathcal{A}$ in an arbitrary way throughout its execution. In the ideal world, the parties and $\mathcal{Z}$ all interact via an ideal functionality $\mathcal{F}$: the parties send their inputs to (with corrupted parties sending anything they like) and receive outputs from $\mathcal{F}$, and $\mathcal{A}$ interacts with $\mathcal{F}$ as specified by $\mathcal{F}$ itself. We let $\text{IDEAL}_{\mathcal{F},\mathcal{A},\mathcal{Z}}(n)$ denote the output of $\mathcal{Z}$ in this case. In the real world, the parties run some protocol π with the corrupted parties behaving arbitrarily as directed by $\mathcal{A}$. We let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n)$ denote the output of $\mathcal{Z}$ in that case. A protocol π *securely realizes the functionality* $\mathcal{F}$ if for any probabilistic polynomial-time (PPT) real-world adversary $\mathcal{A}$ there exists a PPT ideal-world adversary $\mathcal{S}$ (often called a *simulator*) such that for all PPT environments $\mathcal{Z}$ the following is negligible:

$$\left|\Pr[\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n) = 1] - \Pr[\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(n) = 1]\right|.$$

Say we want to design a protocol for some functionality $\mathcal{F}$. It is often helpful to design and reason about this in a *hybrid* world where the parties can run a protocol π while at the same time having access to some ideal functionality $\mathcal{G}$. We let $\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(n)$ denote the output of $\mathcal{Z}$ in that case, and say that π *securely realizes* $\mathcal{F}$ *in the* $\mathcal{G}$*-hybrid model* if for any PPT hybrid-world adversary $\mathcal{A}$ there exists a PPT ideal-world adversary $\mathcal{S}$ such that for all PPT environments $\mathcal{Z}$ we have $\left|\Pr[\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(n) = 1] - \Pr[\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(n) = 1]\right|$. In the UC framework, the following useful composition result holds: if π securely realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, and ρ is any protocol that securely realizes $\mathcal{G}$, then the composed protocol π^ρ securely realizes $\mathcal{F}$ (in the real world).

3 Adaptively Secure Broadcast

In Section 3.1 we propose two definitions of broadcast: a "strong" definition that corresponds to the intuitive notion of broadcast, and a "relaxed" definition that corresponds to the type of broadcast shown to be possible in the non-atomic communication model considered by Hirt and Zikas. (Recall that the stronger notion of broadcast was shown to be impossible in the non-atomic setting.) In Section 3.2 we introduce a special type of commitment scheme, and we show how to construct such schemes in Section 3.2.1. In Section 3.3 we show how to use such commitments to realize adaptively secure broadcast in the atomic communication model.

3.1 Definitions

Classical results show that broadcast (or even relaxed broadcast) cannot be realized for $t \geq n/3$ corrupted parties in a "plain model", and so some setup must be considered if we wish to go beyond this bound. As stated in the Introduction, we will assume a PKI and digital signatures. Within the UC framework, this is modeled by the certificate functionality $\mathcal{F}_{\text{CERT}}$ introduced in [4]. This functionality provides both message-signing capability as well as binding between a signature and a party in the network, and thus simultaneously captures both the presence of a PKI and the ability to issue signatures.

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

The functionality interacts with an adversary $\mathcal{S}$ and a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of parties.

- Upon receiving $(\mathsf{Bcast}, \mathsf{sid}, m)$ from P_i, send $(\mathsf{Bcast}, \mathsf{sid}, P_i, m)$ to all parties in $\mathcal{P}$ and to $\mathcal{S}$.

Figure 1: The broadcast functionality.

Our definitions of broadcast are induced by ideal functionalities in the UC framework. Namely, we say a protocol π achieves (strong) *broadcast* if it securely realizes the functionality $\mathcal{F}_{\text{BC}}$ shown in Figure 1; it achieves *relaxed broadcast* if it securely realizes the functionality $\mathcal{F}_{\text{RBC}}$ given in Figure 2. Our definition of broadcast is essentially standard, though one can also consider a definition where the sender's message m is not revealed to $\mathcal{S}$. (I.e., our definition does not guarantee secrecy for m; note that this only makes a difference when $\mathcal{S}$ corrupts no parties.) Our definition of relaxed broadcast is from [11].

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

The functionality interacts with an adversary $\mathcal{S}$ and a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of parties.

1. Upon receiving $(\mathsf{Bcast}, \mathsf{sid}, m)$ from P_i, send $(\mathsf{Bcast}, \mathsf{sid}, P_i, m)$ to $\mathcal{S}$.
2. Upon receiving m' from $\mathcal{S}$, do:
   - If P_i is corrupted, send $(\mathsf{Bcast}, \mathsf{sid}, P_i, m')$ to all parties in $\mathcal{P}$;
   - If P_i is not corrupted, send $(\mathsf{Bcast}, \mathsf{sid}, P_i, m)$ to all parties in $\mathcal{P}$.

Figure 2: The relaxed broadcast functionality.

It is instructive to examine the two functionalities in light of the Hirt-Zikas attack. Observe that $\mathcal{F}_{\text{BC}}$ does not allow their attack (and so any protocol securely realizing $\mathcal{F}_{\text{BC}}$ must not be susceptible to the attack) since the adversary cannot change the sender's message m unless the adversary corrupts the sender P_i in advance, before it learns m. On the other hand, $\mathcal{F}_{\text{RBC}}$ allows their attack: this is so because the adversary can first learn m (in step 1) and then decide whether to corrupt the sender P_i based on that information; if the adversary decides to corrupt P_1 then the adversary is allowed change the message that will be received by all the other parties in step 2.

The following result was proved in [11]:

Lemma 3.1 *The Dolev-Strong protocol [9] securely realizes $\mathcal{F}_{\text{RBC}}$ in the $\mathcal{F}_{\text{CERT}}$-hybrid model against an adaptive adversary corrupting any $t < n$ parties.*

In fact, the above result holds even in the non-atomic communication model.

It is also possible to define a stronger variant of $\mathcal{F}_{\text{RBC}}$, called $\mathcal{F}^{+}_{\text{RBC}}$, that more closely corresponds to what is actually accomplished by the Hirt-Zikas attack. The difference between $\mathcal{F}_{\text{RBC}}$ and $\mathcal{F}^{+}_{\text{RBC}}$ is that the latter only allows the adversary to have $m' = \perp$. That is, the adversary is allowed to adaptively corrupt the sender (based on the sender's original message) and thereby cause agreement on an error, but is unable to cause agreement on some other valid message. $\mathcal{F}^{+}_{\text{RBC}}$

can be realized fairly easily in the $\mathcal{F}_{\text{RBC}}$-hybrid model using the commitment scheme defined in the following section. Alternately, it can be realized directly in the $\mathcal{F}_{\text{CERT}}$-hybrid model using an appropriate variant of the Dolev-Strong protocol. We omit the details.

3.2 Honest-Binding Commitment Schemes

Commitment schemes are a standard cryptographic tool. Roughly, a commitment scheme allows a sender S to generate a commitment com to a message m in such a way that (1) the sender can later open the commitment to the original value m (*correctness*); (2) the sender cannot generate a commitment that can be opened to two different values (*binding*); and (3) the commitment reveals nothing about the sender's value m until it is opened (*hiding*). For our application, we need a variant of standard commitments that guarantees binding when the sender is honest but ensures that binding can be violated if the sender is dishonest. (In the latter case, we need some additional properties as well; these will become clear in what follows.) Looking ahead, we will use such commitment schemes to construct a broadcast protocol in the following way: the sender will first generate and send a *commitment* to its message, and then send the decommitment information needed to open the commitment. In the simulation for the case when the sender P_i starts out uncorrupted, we will have the simulator $\mathcal{S}$ generate a commitment *dishonestly*. This will give $\mathcal{S}$ the flexibility to break binding and open the commitment to any desired message (if needed), while also being able to ensure binding (when desired) by claiming that it generated the commitment honestly. We defer the details to the next section.

We consider only non-interactive commitment schemes. For simplicity, we define our schemes in such a way that the decommitment information consists of the sender's random coins ω that it used when generating the commitment.

Definition 1 *A (non-interactive)* commitment scheme *for message space $\{\mathcal{M}_k\}$ is a pair of* PPT *algorithms S, R such that for all $k \in \mathbb{N}$, all messages $m \in \mathcal{M}_k$, and all random coins ω it holds that $R(m, S(1^k, m; \omega), \omega) = 1$.*

A commitment scheme for message space $\{\mathcal{M}_k\}$ is honest-binding *if it satisfies the following:*

Binding (for an honest sender) *For all* PPT *algorithms $\mathcal{A}$ (that maintain state throughout their execution), the following is negligible in k:*

$$\Pr\left[\begin{array}{c} m \leftarrow \mathcal{A}(1^k); \omega \leftarrow \{0,1\}^*; \\ \mathsf{com} \leftarrow S(1^k, m; \omega); (m', \omega') \leftarrow \mathcal{A}(\mathsf{com}, \omega) \end{array} \;:\; R(m', \mathsf{com}, \omega') = 1 \bigwedge m' \neq m\right].$$

Equivocation *There is an algorithm $\tilde{S} = (\tilde{S}_1, \tilde{S}_2)$ such that for all* PPT *$\mathcal{A}$ (that maintain state throughout their execution) the following is negligible:*

$$\left| \begin{array}{l} \Pr\left[m \leftarrow \mathcal{A}(1^k); \omega \leftarrow \{0,1\}^*; \mathsf{com} \leftarrow S(1^k, m; \omega) : \mathcal{A}(1^k, \mathsf{com}, \omega) = 1\right] \\ \qquad - \Pr\left[(\mathsf{com}, \mathsf{st}) \leftarrow \tilde{S}_1(1^k); m \leftarrow \mathcal{A}(1^k); \omega \leftarrow \tilde{S}_2(\mathsf{st}, m) : \mathcal{A}(1^k, \mathsf{com}, \omega) = 1\right] \end{array} \right|.$$

Equivocation implies the standard hiding property, namely, that for all PPT algorithms $\mathcal{A}$ (that maintain state throughout their execution) the following is negligible:

$$\left|\Pr\left[(m_0, m_1) \leftarrow \mathcal{A}(1^k);\, b \leftarrow \{0,1\};\, \mathsf{com} \leftarrow S(1^k, m_b) : \mathcal{A}(\mathsf{com}) = b\right] - \frac{1}{2}\right|.$$

We also observe that if (com, ω) are generated by $(\tilde{S}_1, \tilde{S}_2)$ for some message m as in the definition above, then binding still holds: namely, no PPT adversary given m, com, ω can find (m', ω') with $m' \neq m$ such that $R(m', \mathsf{com}, \omega') = 1$.

3.2.1 Constructing Honest-Binding Commitment

We show two constructions of honest-binding commitment schemes. The proofs that these schemes satisfy Definition 3.2 are relatively straightforward, and are therefore omitted.

The first construction, based on the commitment scheme of Naor [13], relies on the minimal assumption that one-way functions exist. We describe the scheme for committing single-bit messages, though it could be extended to arbitrary length messages in the obvious way. In the following, G is a length-tripling pseudorandom generator.

$\underline{S(1^k, m; \omega)}$
parse ω as $\mathsf{crs} \| r$,
  with $|\mathsf{crs}| = 3k$
  and $|r| = k$
$c := G(r) \oplus (\mathsf{crs} \cdot m)$
$\mathsf{com} := (\mathsf{crs}, c)$
return com

$\underline{R(m, (\mathsf{crs}, c), \omega)}$
parse ω as $\mathsf{crs} \| r$,
  with $|\mathsf{crs}| = 3k$
  and $|r| = k$
if $c \stackrel{?}{=} G(r) \oplus (\mathsf{crs} \cdot m)$
  return 1
else return 0

$\underline{\tilde{S}_1(1^k)}$
$r_0, r_1 \leftarrow \{0,1\}^k$
$\mathsf{crs} := G(r_0) \oplus G(r_1)$
$c := G(r_0)$
$\mathsf{com} := (\mathsf{crs}, c)$
$\mathsf{st} := (r_0, r_1, \mathsf{com})$
return $(\mathsf{com}, \mathsf{st})$

$\underline{\tilde{S}_2(\mathsf{st}, m)}$
parse st as (r_0, r_1, com)
parse com as (crs, c)
if $m \stackrel{?}{=} 0$
  $\omega := \mathsf{crs} \| r_0$
else
  $\omega := \mathsf{crs} \| r_1$
return ω

Next, we show an efficient scheme that allows for direct commitments to strings. This construction, based on the Pedersen commitment scheme [15], relies on the discrete-logarithm assumption. In the following, we let $\mathbb{G}$ be a cyclic group of order q, with generator $g \in \mathbb{G}$. (For simplicity, we view $(\mathbb{G}, q, g)$ as public parameters, though they could just as well be generated by the sender.)

$\underline{S(1^k, m; \omega)}$
Parse ω as $h \| x$,
  with $h \in \mathbb{G}$
  and $x \in \mathbb{Z}_q$
return $\mathsf{com} := (h, g^m h^x)$

$\underline{R(m, \mathsf{com}, \omega)}$
Parse ω as $h \| x$,
  with $h \in \mathbb{G}$ and $x \in \mathbb{Z}_q$
if $\mathsf{com} \stackrel{?}{=} (h, g^m h^x)$
  return 1
else return 0

$\underline{\tilde{S}_1(1^k)}$
$r, y \leftarrow \mathbb{Z}_q$
$\mathsf{com} := (g^r, g^y)$
return $(\mathsf{com}, (r, y))$

$\underline{\tilde{S}_2((r, y), m)}$
if $r \stackrel{?}{=} 0$ return $\perp$
$x := (y - m) \cdot r^{-1} \bmod q$
return $\omega := g^r \| x$

3.3 An Adaptively Secure Broadcast Protocol

In this section we show a protocol that securely realizes $\mathcal{F}_{\text{BC}}$ in the $\mathcal{F}_{\text{CERT}}$-hybrid model, in the presence of $t < n$ adaptive corruptions. The challenge of realizing $\mathcal{F}_{\text{BC}}$, and the property that is exactly exploited in the Hirt-Zikas attack on existing protocols, is that when the sender is uncorrupted then the adversary should not learn the sender's message unless all honest parties will (eventually) *agree* on that message (cf. Figure 1). In [11], the authors construct a broadcast protocol for $t < n/2$ by having the sender use verifiable secret sharing (VSS) to "commit" to its message before revealing it. (For $t = n/2$ they use a slight variant of this idea.) This approach works even in the non-atomic communication setting; however, it requires at least half of the parties to be honest.

Our approach is to use computationally secure commitment schemes in place of VSS. That is, we first have the sender announce a *commitment* to its message; once agreement on this commitment is reached, the sender then decommits. (We add an additional stage in which the sender's decommitment is "echoed" by all parties; this prevents a dishonest sender from sending valid decommitment information to some honest parties but not others.) In order to simulate this protocol, we have the sender use honest-binding commitments as introduced in the previous section.

The details of our protocol π_{BC} are presented in Figure 3. We describe our protocol in the $\mathcal{F}_{\text{RBC}}$-hybrid model. Since $\mathcal{F}_{\text{RBC}}$ can be securely realized in the $\mathcal{F}_{\text{CERT}}$-hybrid model (cf. Lemma 3.1), this implies that $\mathcal{F}_{\text{BC}}$ can be securely realized in the $\mathcal{F}_{\text{CERT}}$-hybrid model as well.

Theorem 3.2 *If (S, R) is an honest-binding commitment scheme, then π_{BC} securely realizes $\mathcal{F}_{\text{BC}}$ in the $\mathcal{F}_{\text{RBC}}$-hybrid model against an adaptive adversary corrupting any $t < n$ of the parties.*

The above theorem holds in the atomic communication model considered in this paper; protocol π_{BC} does *not* securely realize $\mathcal{F}_{\text{BC}}$ in the non-atomic communication model of [11]. (Indeed, by the impossibility result proven in [11], it cannot.) Atomic communication is used crucially in the second stage of our protocol when the sender transmits decommitment information to all the parties. (Observe this is the only step in our protocol in which parties communicate directly, rather than via the ideal functionality $\mathcal{F}_{\text{RBC}}$.) If non-atomic communication were assumed, then the adversary could learn the decommitment information (and thus the sender's message) first, and then decide to corrupt the sender and *not* transmit the decommitment information to any of honest parties.

Proof Let $\mathcal{A}$ be an active, adaptive adversary that interacts with players running the above protocol in the $\mathcal{F}_{\text{RBC}}$-hybrid model. We construct an adversary (simulator) $\mathcal{S}$ running in the ideal world with access to functionality $\mathcal{F}_{\text{BC}}$, such that no PPT environment $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ and parties running π_{BC} in the $\mathcal{F}_{\text{RBC}}$-hybrid model, or whether it is interacting with $\mathcal{S}$ and (dummy) parties communicating directly with $\mathcal{F}_{\text{BC}}$. The simulator $\mathcal{S}$ starts by internally invoking the adversary $\mathcal{A}$, and forwarding all messages between $\mathcal{A}$ and $\mathcal{Z}$ in the usual way. The simulator will simulate both the ideal functionality $\mathcal{F}_{\text{RBC}}$ for $\mathcal{A}$, as well as an execution of protocol π_{BC}.

In our description of $\mathcal{S}$, we distinguish two cases depending on whether or not the sender P_i is corrupted at the outset.

Case 1: We first treat the easier case where P_i is corrupted at the outset. Here, $\mathcal{A}$ corrupts P_i (in the hybrid world) and so $\mathcal{S}$ corrupts P_i (in the ideal world). Any additional corruptions that $\mathcal{A}$ requests during its execution can be easily simulated by $\mathcal{S}$, so we do not mention them.

Protocol π_{BC}

The protocol is carried out among a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of parties. For notational convenience, we let P_i denote the sender (though in fact any party can act as sender). We let (S, R) be a non-interactive commitment scheme.

- **Stage 1:** Upon receiving input $(\mathsf{Bcast}, \mathsf{sid}, m)$ from the environment $\mathcal{Z}$, the sender P_i chooses random $\omega \leftarrow \{0,1\}^*$, computes $\mathsf{com} := S(1^k, m; \omega)$, and sends $(\mathsf{Bcast}, \mathsf{sid}, \mathsf{com})$ to $\mathcal{F}_{\mathrm{RBC}}$. Let com^* denote the value received by the honest parties in this stage (note that this value is the same for all honest parties).
- **Stage 2:** Upon receiving $(\mathsf{Bcast}, \mathsf{sid}, P_i, \mathsf{com})$ from $\mathcal{F}_{\mathrm{RBC}}$, the sender P_i sends (m, ω) to every other party over point-to-point channels.
- **Stage 3:** The following is done by each party $P_j \in \mathcal{P}$: Let (m_j, ω_j) denote the value that P_j received from P_i in stage 2. (If P_j receives nothing, it takes (m_j, ω_j) as some default values.) P_j sends $(\mathsf{Bcast}, \mathsf{sid}, (m_j, \omega_j))$ to $\mathcal{F}_{\mathrm{RBC}}$.
- **Stage 4:** Each party P_j receives messages $\{(\mathsf{Bcast}, \mathsf{sid}, P_k, (m_k, \omega_k))\}_{P_k \in \mathcal{P}}$ from $\mathcal{F}_{\mathrm{RBC}}$, taking (m_k, ω_k) as some default values if nothing is received (note that the (m_k, ω_k) values are the same for all honest parties). Each party P_j then decides on its output as follows: Let $\mathsf{valid} = \{k \in \{1, \ldots, n\} \mid R(m_k, \mathsf{com}^*, \omega_k) = 1\}$. If valid is empty, then output some default value. Otherwise, let k^* be the smallest value in valid and output m_{k^*}.

Figure 3: A protocol realizing $\mathcal{F}_{\mathrm{BC}}$ in the $\mathcal{F}_{\mathrm{RBC}}$-hybrid model.

When $\mathcal{Z}$ provides input to P_i, this input is read by $\mathcal{S}$ who forwards it to $\mathcal{A}$. Then $\mathcal{A}$ begins running the first stage of π_{BC} (on behalf of the corrupted P_i) by specifying some message $(\mathsf{Bcast}, \mathsf{sid}, \mathsf{com}^*)$ to send to $\mathcal{F}_{\mathrm{RBC}}$. The simulator $\mathcal{S}$ stores com^*, and simulates the response of $\mathcal{F}_{\mathrm{RBC}}$ by giving $(\mathsf{Bcast}, \mathsf{sid}, P_i, \mathsf{com}^*)$ to $\mathcal{A}$ (and all corrupted parties). Next, $\mathcal{A}$ (now executing the second stage of π_{BC}) decides on messages (m_j, ω_j) to send to each honest party P_j on behalf of P_i. In response, $\mathcal{S}$ simulates the third stage of π_{BC} by giving $(\mathsf{Bcast}, \mathsf{sid}, P_j, (m_j, \omega_j))$ to $\mathcal{A}$ for every honest party P_j. For each such P_j, the adversary $\mathcal{A}$ may then choose to (corrupt P_j and) replace (m_j, ω_j) by some other message (m'_j, ω'_j). Once $\mathcal{A}$ has sent some (m'_j, ω'_j) to the appropriate instance of $\mathcal{F}_{\mathrm{RBC}}$ for all P_j, the simulator simulates the output of $\mathcal{F}_{\mathrm{RBC}}$ for all corrupted parties in the obvious way. Finally $\mathcal{A}$, executing the third stage of π_{BC} on behalf of the remaining corrupted parties, specifies messages $(\mathsf{Bcast}, \mathsf{sid}, (m'_j, \omega'_j))$ that each such party P_j should send to $\mathcal{F}_{\mathrm{RBC}}$.

$\mathcal{S}$ now has values (m_k, ω_k) for every $P_k \in \mathcal{P}$, defined by the output of each appropriate (simulated) instance of $\mathcal{F}_{\mathrm{RBC}}$ in the (simulated) third stage of the protocol. $\mathcal{S}$ defines a set valid and determines k^*, m_{k^*} as prescribed by the protocol. It then sends $(\mathsf{Bcast}, \mathsf{sid}, m_{k^*})$ (on behalf of P_i) to its own ideal functionality $\mathcal{F}_{\mathrm{BC}}$.

It is not hard to see that $\mathcal{S}$ provides a perfect simulation. The view of $\mathcal{A}$ is clearly identical whether it is running in the $\mathcal{F}_{\mathrm{RBC}}$-hybrid model or whether it is being run as a sub-routine by $\mathcal{S}$ in the ideal world with access to $\mathcal{F}_{\mathrm{BC}}$. As for the outputs of the honest parties (i.e., those that

are honest by the end of the protocol execution), note that if $\mathcal{A}$ were running in the $\mathcal{F}_{\text{RBC}}$-hybrid model then every honest party P_j would receive com^* in the first stage and $\{(m_k, \omega_k)\}_{P_k \in \mathcal{P}}$ in the third stage, and would thus decide on output m_{k^*} exactly as $\mathcal{S}$ does. Since $\mathcal{S}$ sends m_{k^*} to $\mathcal{F}_{\text{BC}}$, the output of each honest party in the ideal world is also m_{k^*}. We remark that the fact that the commitment scheme is not binding (for a malicious sender) is irrelevant here.

Case 2: We now turn to the more difficult case where P_i is not corrupted at the outset. As before, adaptive corruptions of parties other than P_i can be handled easily, so we do not mention it. Corruption of P_i will, however, be explicitly mentioned.

$\mathcal{S}$ begins by computing $(\mathsf{com}, \mathsf{st}) \leftarrow \tilde{S}_1(1^k)$. It then simulates the first stage of π_{BC} (on behalf of the honest P_i) by giving to $\mathcal{A}$ the message $(\mathsf{Bcast}, \mathsf{sid}, P_i, \mathsf{com})$ on behalf of $\mathcal{F}_{\text{RBC}}$. At this point, $\mathcal{A}$ can choose whether to corrupt P_i or not, and we further divide our description of $\mathcal{S}$ depending on which is the case.

If $\mathcal{A}$ requests to corrupt P_i, then $\mathcal{S}$ corrupts P_i and waits until it receives input $(\mathsf{Bcast}, \mathsf{sid}, m)$ from $\mathcal{Z}$. At that point, $\mathcal{S}$ computes $\omega \leftarrow \tilde{S}_2(\mathsf{st}, m)$ and gives m and ω to $\mathcal{A}$ as the state of P_i. The remainder of the simulation then proceeds exactly as in the case when P_i was corrupted at the outset. (Note in particular that $\mathcal{A}$ may choose to change com to some other value com^*.)

If $\mathcal{A}$ does not corrupt P_i, then $\mathcal{S}$ waits until it receives a message $(\mathsf{Bcast}, \mathsf{sid}, P_i, m)$ from its ideal functionality $\mathcal{F}_{\text{BC}}$. (Note that at this point, the output of every honest party in the ideal world is m.) $\mathcal{S}$ then computes $\omega \leftarrow \tilde{S}_2(\mathsf{st}, m)$, and simulates the second phase of the protocol by sending (m, ω) to every corrupted party. The remainder of the protocol is simulated in the obvious way, essentially the same as before (with the only difference being that it provides state m, ω to $\mathcal{A}$ if P_i is ever corrupted).

In this case, $\mathcal{S}$ provides a computationally indistinguishable simulation for $\mathcal{Z}$. The only difference between the view of $\mathcal{A}$ in the above simulation and the view of $\mathcal{A}$ when it is running in the $\mathcal{F}_{\text{RBC}}$-hybrid model is with regard to (com, ω): in the former case these are produced using $(\tilde{S}_1, \tilde{S}_2)$, whereas in the latter case these are produced using the honest sender algorithm. Definition 3.2 guarantees that these distributions are computationally indistinguishable. As for the outputs of the honest parties, if P_i is corrupted during stage 1 then the argument is as given previously. If P_i is not corrupted during stage 1, then we need to argue that with all but negligible probability every honest party would output m in that case in the $\mathcal{F}_{\text{RBC}}$-hybrid world (since, as noted above, every honest party outputs m in that case in the ideal world). This follows from the honest-binding property of Definition 3.2. ■

4 Adaptively Secure MPC

In the previous section we showed a protocol (call it bc) that securely realizes the broadcast functionality $\mathcal{F}_{\text{BC}}$ in the presence of an adaptive adversary corrupting any number of parties. Given any protocol π (e.g., the one of [7]) for securely computing some function f in the presence of an adaptive adversary corrupting any number of parties in the $\mathcal{F}_{\text{BC}}$-hybrid model (i.e., protocol π assumes an ideal broadcast channel), the composed protocol π^{bc} securely computes f in the presence of an adaptive adversary corrupting any number of parties in the $\mathcal{F}_{\text{CERT}}$-hybrid model, using point-to-point communication only. The above is stated in the UC framework, but an analogous composition theorem could be stated with respect to "stand-alone" notions of security as well [2]. (We refer the reader to [10] for detailed definitions of security for MPC with dishonest majority.)

Even given the above, it is interesting to explore whether adaptively secure MPC can be achieved in the weaker $\mathcal{F}_{\text{RBC}}$-hybrid model, for at least two reasons:

- If we take as our communication model the non-atomic, point-to-point model of Hirt-Zikas, it is impossible to realize $\mathcal{F}_{\text{BC}}$ when $t > n/2$. Thus, if we want to realize adaptively secure MPC for $t > n/2$ in this communication model, some other approach is needed.
- Even in the atomic communication model, one may prefer to base adaptively secure MPC on relaxed broadcast rather than broadcast since protocols for the former may be more efficient than protocols for the latter.

Note that, in the case of dishonest majority, adaptively secure MPC does not imply adaptively secure broadcast because the usual notions of security for MPC do not guarantee output delivery or fairness (see [10] for a more extensive treatment) — these properties are, in general, not achievable [8] — whereas definitions of security for broadcast do require guaranteed output delivery. In particular, the Hirt-Zikas impossibility result for adaptively secure broadcast in the non-atomic communication model says nothing about the feasibility of adaptively secure MPC in that setting.

Although we cannot claim that all adaptively secure MPC protocols using broadcast remain secure when broadcast is replaced with relaxed broadcast, specific protocols from the literature do remain secure in that case. Once again, we focus on protocols in the UC framework, though we expect these results would extend to protocols in the "stand-alone" setting as well.

Specifically, consider the adaptively secure MPC protocol π of Canetti, Lindell, Ostrovsky, and Sahai [7], which relies on a broadcast channel. We first observe that the protocol remains secure even in the non-atomic communication model. In either communication model, the protocol also remains secure if the broadcast channel is replaced with relaxed broadcast. At a high level, the reason is that the messages that are broadcast are always commitments to some values, except in the last round where the broadcast messages reveal the output. The ability to corrupt a sender based on the message being broadcast is "useless" in the former case; in the latter case such an attack corresponds to preventing output delivery/violating fairness, something which is permitted by the definitions of security when there is a dishonest majority. We remark that the advantage of using relaxed broadcast as opposed to the "echo broadcast" protocol from [10] is that the former ensures agreement on abort.

Even given the above, there are several reasons to securely realize $\mathcal{F}_{\text{BC}}$ rather than be contended with $\mathcal{F}_{\text{RBC}}$. First, one may be interested in broadcast itself, rather than as a sub-protocol for some larger task. Furthermore, there is an advantage to working with $\mathcal{F}_{\text{BC}}$ in that it can be safely used to instantiate the broadcast channel in *arbitrary* protocols, so one can avoid having to examine protocols on a case-by-case basis to determine whether $\mathcal{F}_{\text{RBC}}$ suffices.

Note: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes.

References

[1] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 186–195. IEEE, 2004.

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

[3] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001. Full version at `http://eprint.iacr.org/2000/067/`.

[4] R. Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop*, pages 219–235. IEEE Computer Society, 2004. Full version at `http://eprint.iacr.org/2003/239/`.

[5] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648. ACM Press, 1996.

[6] R. Canetti and M. Fischlin. Universally composable commitments. In *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, 2001.

[7] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, 2002.

[8] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369. ACM Press, 1986.

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

[10] S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *J. Cryptology*, 18(3):247–287, 2005.

[11] M. Hirt and V. Zikas. Adaptively secure broadcast. In *Advances in Cryptology — Eurocrypt 2010*, volume 6110 of *LNCS*, pages 466–485. Springer, 2010.

[12] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Trans. Programming Language Systems*, 4(3):382–401, 1982.

[13] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

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

[15] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — Crypto '91*, volume 576 of *LNCS*, pages 129–140. Springer, 1992.

[16] B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *9th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *LNCS*, pages 339–350. Springer, 1992.