Adaptively Secure Broadcast in Resource-Restricted Cryptography

Ran Cohen[*] Juan Garay[†] Vassilis Zikas[‡]

June 8, 2021

Abstract

The advent of blockchain protocols has reignited the interest in adaptively secure broadcast, as it is by now well known that broadcasting over a diffusion network allows an adaptive adversary to corrupt the sender depending on the message s/he attempts to send and change it. Hirt and Zikas [Eurocrypt '10] proved that this is an inherent limitation of broadcast in the simulation-based setting, i.e., that this task is impossible against an adaptive adversary corrupting a strict majority of the parties.

In this work, we show that, contrary to previous perception, the above limitation is not an artifact of simulation-based security, but that it also applies to the property-based broadcast definition adapted for adaptive adversaries. We then turn to the resource-restricting cryptography (RRC) paradigm, which was proven useful in circumventing strong impossibility results, and ask whether it also allows us to circumvent the above negative result. We answer this question in the affirmative, by showing that time-lock puzzles (TLPs)—which can be viewed as an instance of RRC—indeed allow for achieving the property-based definition and circumvent the impossibility of adaptively secure broadcast.

The natural question is then, do TLPs also allow for simulation-based adaptively secure broadcast against corrupted majorities? It turns out that they do not, which serves as yet another motivation for simulation-based security, especially when dealing with adaptive adversaries. Nonetheless, we show that a positive result can be achieved if we turn to what is essentially a *non-committing* version of TLPs, which uses access to a programmable random oracle.

[*]Northeastern University. E-mail: `rancohen@ccs.neu.edu`. Research supported by NSF grant no. 2055568.

[†]Texas A&M University. E-mail: `garay@cse.tamu.edu`. Research supported by NSF grants no. 2001082 and 2055694.

[‡]Purdue University. E-mail: `vzikas@cs.purdue.edu`. Research supported in part by NSF grant no. 2055599 and by Sunday Group.

Contents

1 Introduction

A physical broadcast channel enables a set of n parties to communicate as if talking via a megaphone: Once a party speaks, all other parties are guaranteed to hear its message. In a *broadcast protocol* (aka Byzantine generals [64, 56]) the parties are asked to realize this megaphone capability over point-to-point channels, even when a subset of them collude and actively disrupt the protocol's execution. The standard formulation of a broadcast protocol requires two core properties: *agreement* (all honest parties output the same value, even if the sender is cheating) and *validity* (if the sender is honest then all honest parties output its message). A broadcast protocol is t-resilient if both properties hold facing any set of (up to) t misbehaving and colluding parties.

Broadcast is one of the most studied problems in the context of fault-tolerant distributed computing and cryptographic protocols, leading to numerous breakthrough results. For example, classical results show that while t-resilient broadcast protocols can be constructed in the plain model for $t < n/3$ [64, 35], a larger corruption threshold cannot be tolerated [56, 33, 11]. Overcoming this lower bound requires working in weaker models. A common approach is to assume a setup assumption in the form of a *public-key infrastructure* (PKI) for digital signatures [27] (where every party generates a pair of signing/verification keys, and publishes its verification key during the setup phase), or more involved *correlated randomness* (where a trusted party generates correlated secrets to the parties before the protocol begins; e.g., an "information-theoretic PKI" [65]); this approach enables broadcast protocols tolerating any number of $t \leq n$ corruptions.[1]

The resource-restricting paradigm. Another, more recent approach to weakening the model without using "private-coin setup assumptions" is the *resource-restricting cryptography* paradigm [40], where instead of considering arbitrary adversaries that run in probabilistic polynomial time (PPT), additional restrictions are assumed on their capabilities. For example, when the computational power of the adversary is smaller than the combined computational power of the honest parties, Nakamoto-style consensus [38, 63] employs proofs of work (PoWs) [28] to overcome the aforementioned lower bound without relying on PKI-like setup assumptions. This is a fruitful promising approach that has led to broadcast protocols [3] and secure multi-party computation protocols [40] that can tolerate any dishonest minority, given only a "public-coin setup."

Another example of resource-restricting cryptography is *time-based hardness*. Here there is no restriction on the overall computational power of the adversary (other than being PPT); instead, there is an assumed bound on the number of parallel steps that the adversary can take within a given time interval. This assumption enables the usage of *time-lock puzzles* (TLPs) [67, 7] and has been used for example by Boneh and Naor [9] to overcome the lower bound of Cleve [23] and construct a fair coin-tossing protocol. This approach has led to several interesting results, such as resource fairness [36], non-interactive non-malleable commitments [57], and round-efficient randomized broadcast [69]. Another use case of time-based hardness which has been shown to be sufficiently strong to overcome Cleve's impossibility is *verifiable delay functions* [10, 66, 71].

Simulation-based vs. property-based definitions. Secure multi-party computation protocols (MPC) [72, 43] enable a set of mutually distrusting parties to compute a function on their private inputs, while guaranteeing various properties such as correctness, privacy, independence of inputs, and more. While the original security definitions had the above property-based flavor, nowadays the standard definition formalizes the above requirements (and others) in a simulation-based

[1]For the related *consensus* problem (aka Byzantine agreement), where all parties have an input, the best achievable bound is $t < n/2$ [34].

manner [15, 16, 42]. Informally, in the simulation paradigm for security, the protocol execution is compared to an ideal world where the parties have access to a trusted party (aka the "ideal functionality") that captures the security properties the protocol is required to achieve. The trusted party takes the parties' inputs and performs the computation on their behalf. A protocol is then regarded secure if for any adversary attacking it, there exists an ideal-world adversary (the "simulator") attacking the execution in the ideal world, such that no external distinguisher (environment) can tell the real and the ideal executions apart.

Simulation-based definitions provide several advantages compared to the property-based approach. First, in a property-based definition, it may be the case that an important property is missed (e.g., one may require privacy of the inputs but neglect to require input independence); this may be subtle to notice since the properties should capture both the guarantees towards the honest parties as well as the influence the adversary may have over the computation. Second, the holistic approach provides a simple and clear definition that can be applied in complex settings, such as adaptive corruptions and concurrent executions. Third, many simulation-based security definitions guarantee security under composition, which enables analyzing a complex task where sub-protocols are modeled as ideal functionalities, and later replaced by protocols securely realizing them.

For the specific case of broadcast, the commonly used ideal functionality (e.g., [19, 44]) mimics an ideal megaphone in a rather simple way: First, the sender provides its message to the ideal functionality, who later hands it out to the adversary and to all the parties.

Adaptively secure broadcast. It is not hard to see that a broadcast protocol that is secure according to the property-based definition also realizes the ideal megaphone functionality when the set of corrupted parties is defined at the onset of the protocol (i.e., when the adversary is *static*). However, as observed by Hirt and Zikas [48], this is no longer true in the adaptive-corruption setting. The issue is that a rushing adversary may be the first to learn the input message of the sender, in which case it can corrupt the sender and replace its message (or simply crash it, in case of fail-stop adversaries). For example, the protocol of Dolev and Strong [27] (and the vast majority of the protocols in the literature) begins by having the sender send its message to all other parties, who then proceed to make sure they all agree on the output value. In case the first party receiving this message is corrupted, the adversary can decide whether to corrupt the sender (thus preventing all other parties from learning it) as a function of that message.

Hirt and Zikas [48] defined a weaker functionality that captures this capability of the adversary to influence the output. In this *unfair broadcast* functionality, once the functionality receives the input from the sender, it first hands it to the adversary, who can now corrupt the sender and replace its input *before* the functionality sends the output to the remaining honest parties. Such a broadcast protocol is *unfair* because the adversary gets a "double dipping" capability to both learn the sender's input before other parties *and* to change it. This is in contrast to the megaphone functionality that allows the adversary to *either* be the first to learn the message *or* to corrupt the sender (without first learning the message) and choose the output, but not both. This difference is best illustrated when each party broadcasts a random bit: if unfair broadcast is used the adversary has the capability to bias the agreed-upon bits towards 0 by corrupting only senders that broadcast 1 and flipping their bit, whereas if the ideal megaphone is used the adversary does not get any advantage over simply randomly guessing which parties broadcast 1.[2]

Hirt and Zikas [48] further showed that the megaphone functionality can be realized for $t \leq n/2$ (i.e., when the adversary cannot corrupt a majority of the parties); the idea is for the sender to

[2]In the context of collective coin tossing, the capability of the adversary to first learn the sender's message and later to corrupt the sender and change its input has been referred to as *strongly adaptive* [45, 50, 54, 46].

"commit" its message into the system using verifiable secret sharing (VSS), and later use unfair broadcast to reconstruct the original message (as observed in [24, 25], robust secret sharing can be used instead of VSS). For the dishonest-majority setting, Hirt and Zikas [48] showed an impossibility result for adaptively secure broadcast protocols that realize the megaphone functionality. This impossibility captures a large class of protocols (that includes all of the known approaches to construct broadcast protocols) where there is an *a-priori*-known round R such that prior to round R it is guaranteed that no set of size $\lfloor n/2 \rfloor - 1$ "knows" the sender's input (in the sense that if this set emulates in its head a continuation of the protocol where all other parties crash, it has a noticeable error probability), and at round R there exists a set of size $\lfloor n/2 \rfloor - 1$ that "knows" the sender's input (i.e., by emulating the continuation, the set errs only with negligible probability). In the sequel, we will denote this class of "step-release" protocols by $\Pi_{\mathsf{step\text{-}rel}}$.

Atomic vs. non-atomic multisend. The attack from [48] applies in the so called *non-atomic multisend* model, where sending multiple messages to the network are considered as separate operations. This is the classical model considered in the distributed-computing literature since the '80s (e.g., [32, 27, 33]), where the adversary could corrupt a party and crash it (or change its input [31]) after the party sends its messages to some of the parties, but before it completed sending to all parties. This is also the standard model for capturing adaptive corruptions in the MPC literature (e.g., [18, 15, 16, 20]). The non-atomic multisend model captures, for example, the setting where the outgoing communication of a party goes through a central router (e.g., an ISP) that may block some (or all) outgoing messages thus enabling the adversary to perform its attack. The ability of the adversary to corrupt a party in such a manner is also referred to as *strongly rushing* [2, 1, 69].

Garay et al. [37] noticed that this attack does not translate to the *atomic multisend* model where the sender is guaranteed not to be corrupted in the time between sending its first message for a given round and the time it completes sending all messages for that round, and, further, a message that has been sent is guaranteed to arrive at its destination. Interestingly, Garay et al. [37] showed another variation of this attack illustrating that the protocol of Dolev and Strong [27] (and all other protocols in the literature) does not realize the megaphone functionality even in the atomic-multisend model. Complementarily, Garay et al. [37] constructed an adaptively secure broadcast protocol tolerating $t < n$ corruptions in this model.

The atomic-multisend model has recently gained popularity in many consensus protocols that seek security against adaptive corruptions (e.g., [22, 1, 21, 70, 8]). However, the non-atomic-multisend model is a preferred model as it requires less assumptions on the underlying communication network. This model is more challenging to work with as it considers more powerful adversaries; indeed, certain impossibility results in the non-atomic multisend model do not translate to the atomic-multisend realm [48, 12, 1].

Thus, the main question we ask in this paper is:

> *Can the impossibility of adaptively secure broadcast [48] be circumvented in the resource-restricted paradigm?*

Intriguingly, the answer to the above seemingly innocent question is different depending on the definition of (adaptively secure) broadcast one adopts (property-based vs. simulation-based) and/or on how strong a setup we are willing to assume. In particular, we answer this question in the affirmative in the case of property-based definition via TLPs, which can be viewed as an instance of RRC. However, in the case of simulation-based security, it turns out that TLPs do not suffice. Nonetheless, we show that a positive result—i.e., simulation-based adaptively secure broadcast against corrupted majorities—can be achieved based on *non-committing* TLPs, which use access to a programmable random oracle.

1.1 Our Contributions

Our work does a thorough investigation of adaptively secure broadcast both in the property-based and in the simulation-based security settings.

Property-based definition of broadcast. Our first contribution towards overcoming the impossibility results of Hirt and Zikas [48] is to come up with a simpler, property-based definition that captures the essence of their attack. Property-based definitions are indeed better suited for understanding lower bounds as they highlight the exact attack surface that is exploited. With the illustrating example of broadcasting a random bit in mind (where the adversary's goal is to corrupt only parties who broadcast 1 and flip their bit), we provide the following definition (see Definition 5 for a formal version).

Definition (Broadcast, property-based definition, informal)**.** *An n-party protocol is an* adaptively secure t-resilient broadcast protocol *according to the property-based definition if, in addition to agreement and validity, it satisfies the following fairness property:*

- ***Fairness:*** *The probability of any PPT adversary to win the following fairness game is bounded by $1/2 + \mathsf{negl}(\kappa)$ (where κ denotes the security parameter). When attacking an execution of the protocol where the sender begins with a random bit $b \leftarrow \{0, 1\}$ as its input, we say that the adversary wins the fairness game if one of the following events occurs:*
  - *$b = 0$ and the sender remained honest at the end of the protocol;*
  - *$b = 1$ and the common output of the honest parties is 0.*

Property-based vs. simulation-based broadcast. It is not hard to verify that any broadcast protocol that is secure according to the simulation-based definition (i.e., realizes the ideal megaphone functionality) is also secure according to the property-based definition of (adaptively secure) broadcast. The intuition is that a simulator that interacts with the megaphone functionality can win the fairness game only with probability $1/2$ (by guessing the input), and therefore any adversary that can win the fairness game with a noticeable probability over $1/2$ can be translated to a distinguisher between the real and ideal computations. We formally prove this result in Lemma 9.

However, one may ask whether the property-based definition is actually weaker than the simulation-based definition, or if it is equivalent. Stated differently, does the property-based definition above capture the attack from [48]? Indeed, the attack from [48] rules out the simulation-based definition but that may be due to another feature of the megaphone functionality.

Our second contribution is extending the impossibility result from [48] to rule out the property-based definition for the same class of protocols $\Pi_{\mathsf{step\text{-}rel}}$. This means that in term of *feasibility*, the simulation-based definition and the property-based definition are *equivalent*, i.e., for $t \leq n/2$ both definitions can be satisfied, and for $t > n/2$ both definitions cannot be satisfied (by protocols from the class $\Pi_{\mathsf{step\text{-}rel}}$). Note that this does not imply that any protocol that satisfies the property-based definition also satisfies the simulation-based definition.

Theorem 1 (Impossibility for property-based broadcast, informal)**.** *Let $t > n/2$. Then, there is no adaptively secure broadcast protocol (from the class $\Pi_{\mathsf{step\text{-}rel}}$) tolerating a fail-stop, PPT t-adversary that satisfies the property-based definition of (adaptively secure) broadcast.*

We note that the impossibility result holds even assuming any correlated-randomness setup and/or secure data erasures.

Overcoming the property-based impossibility via TLPs. Next, we study whether the resource-restricting paradigm can be used to overcome the impossibility of adaptively secure broadcast. We use time-lock puzzles [67, 7] for this task. The idea is quite simple: the sender "hides" its message inside a TLP and uses a protocol for unfair broadcast (e.g., [27, 69]) to send the puzzle to all parties; every recipient can open the puzzle after investing a polynomial amount of computation and obtain the output.

The guarantee provided by a TLP with gap $\varepsilon < 1$ is that when setting the puzzle with difficulty parameter $T^{1/\varepsilon}$, any adversary that can evaluate circuits of polynomial size, but of depth bounded by $T(\kappa)$, cannot solve the puzzle with better than negligible probability. We say that an adversary is (R, T)-*bounded* if the number of parallel steps it can take within R rounds is bounded by $T(\kappa)$. Therefore, if the unfair broadcast protocol takes R rounds, we are guaranteed that any (R, T)-bounded adversary cannot win the fairness game with more than $1/2 + \mathsf{negl}(\kappa)$ probability.

In fact, our protocol does not require a "lightweight" generation of the puzzle, and can use a puzzle generation that is as computationally expensive as solving the puzzle. Therefore, we only require the *weak* variant of time-lock puzzles [59, 7] that allows for parallelizable, yet computationally expensive puzzle generation, and can be based on one-way functions and the existence of non-parallelizing languages [7].

Theorem 2 (Feasibility of property-based broadcast via TLPs, informal)**.** *Let $t \leq n$, let T be a polynomial, assume that weak time-lock puzzles exist, and that unfair broadcast can be computed in R rounds. Then, there is an adaptively secure broadcast protocol tolerating an (R, T)-bounded t-adversary that satisfies the property-based definition of (adaptively secure) broadcast.*

TLP barriers for simulation-based broadcast. Next, we ask whether time-lock puzzles are also sufficient to satisfy the simulation-based definition of broadcast. Somewhat surprisingly, the answer to this question is negative, thus posing a separation between the two definitions. The main reason is illustrated when trying to simulate the protocol that satisfies the property-based definition. When the sender is honest and a simulator tries to simulate the TLP without knowing the message, it gets stuck, since the TLP is a committing object: Once the puzzle is generated it can only be opened to a unique value. Therefore, the simulator success probability is again restricted to correctly guessing the sender's input, which results in a noticeable distinguishing probability between the real and ideal computations.

In Section 5.1 we extend this argument to rule out *any* adaptively secure broadcast protocol (from the class $\Pi_{\mathsf{step\text{-}rel}}$) even facing an (R, T)-bounded adversary. In turn, this impossibility result implies that the TLP assumption is not sufficient for realizing simulation-based broadcast.

Theorem 3 (Impossibility for simulation-based broadcast from TLPs, informal)**.** *Let $t > n/2$, and let R and T be polynomials. Then, there is no adaptively secure broadcast protocol (from the class $\Pi_{\mathsf{step\text{-}rel}}$) tolerating an (R, T)-bounded, fail-stop, PPT t-adversary that satisfies the simulation-based definition of broadcast.*

The impossibility result can be extended to hold even assuming any correlated-randomness setup, secure data erasures, a non-programmable random oracle, and time-lock puzzles.

Overcoming the simulation-based impossibility via a programable random oracle. We note that this "barrier" resembles other barriers in achieving adaptive security of committing cryptographic primitives, such as commitments [17] and public-key encryption [61]. Next, we show that a programmable random oracle can be used to construct a non-committing variant of TLPs, which in turn allows us to overcome the above barrier. Namely, instead of hiding the message m inside the puzzle, the sender samples a random one-time pad key x, hides x inside the puzzle, and unfairly broadcast the puzzle along with $c = m \oplus H(x)$. Now the simulator can simulate a puzzle when the sender is honest, and upon a corruption request of the sender (or after R rounds have elapsed, and it can safely ask the megaphone functionality for the output), the simulator can program the random oracle appropriately.

Theorem 4 (Feasibility of simulation-based broadcast via TLPs in the RO model, informal)**.** *Let $t \leq n$, let T be a polynomial, assume that weak TLPs exist, and that the unfair broadcast can be computed in R rounds. Then, there is an adaptively secure broadcast protocol, according to the simulation-based definition, tolerating an (R,T)-bounded t-adversary in the programmable random-oracle model.*

Summary of our contributions. Taken together, our results distill the essence of the impossibility result from [48]; this emphasizes that the impossibility is not just an artifact of the simulation-based definition, but it also applies to the natural extension of the property-based broadcast definition to the adaptive-corruptions case. Further, we show how the resource-restricting paradigm separates the property-based definition from the simulation-based definition. This serves as yet another motivation for using simulation-based security, especially when designing adaptively secure protocols. Finally, we put forth the notion of *non-committing* time-lock puzzles in the programmable random oracle, and use it to achieve simulation-based adaptively secure broadcast. Our results are summarized in Table 1.

	property-based	**simulation-based**
PKI	✗ **Thm 1**	✗ HZ [48]
PKI+TLP	✓ **Thm 2**	✗ **Thm 3**
PKI+RO+TLP	✓ **Thm 2**	✓ **Thm 4**

Table 1: Feasibility of adaptively secure broadcast with non-atomic multisend synchronous communication. The left column considers the property-based definition and the right one the simulation-based definition. All negative results (lower bounds) are for protocols in the class $\Pi_{\mathsf{step\text{-}rel}}$ and hold for any dishonest majority of fail-stop corruptions and any correlated-randomness setup; all positive results (protocol constructions) tolerate an arbitrary number of malicious corruptions and require a PKI for signatures. TLP stands for a weak time-lock puzzle and RO stands for programmable random-oracle model.

1.2 Additional Related Work

Recently, Wan et al. [69] used time-lock puzzles to construct adaptively secure (unfair) broadcast protocols in the non-atomic multisend model, with the goal of reducing the round complexity of randomized broadcast from linear to poly-logarithmic, facing a constant fraction of corrupted parties. As pointed out by the authors, their goal was not to realize the megaphone functionality, but only to satisfy the property-based definition of (unfair) broadcast. The main idea in [69] is to use TLPs to "hide" the contents of the messages within a round in a way that essentially provides atomic-multisend guarantees. Given this, they run the poly-logarithmic-round protocol of Chan et al. [21], which in turn is based on Dolev and Strong [27]. We note that although the protocol in [69] relies on similar assumptions as the ones in this work, it does not answer the question posed in this paper as it is vulnerable to the attack from [37], showing that the Dolev-Strong protocol [27] is not adaptively secure even in the atomic-multisend model.

Baum et al. [6] study a stronger version of TLPs that provide universal composability. They define an ideal TLP functionality, and prove that realizing it inherently requires a programmable random oracle. Next, they realize the TLP functionality based on generic-group-style formalization of the repeated-squaring technique from [67] as well as a restricted programmable and observable random oracle [14]. In contrast to the weaker, property-based definition of TLP [67, 7] (used in this paper), the reliance on a random oracle enables the TLP functionality to define a known step with the guarantee that the adversary learns nothing about the content of the TLP prior to that step and that once that step is reached, the content of the puzzle is fully revealed.

In more detail, Baum et al. [6] give an elegant argument showing that coin-flipping protocols based on TLP, such as the one of Boneh and Naor [9], cannot be simulated without resorting to a programmable RO, even facing so-called *computationally restricted environments*. Essentially, when simulating a TLP-based coin-flipping protocol, the environment may first get the information needed to learn the output (possibly after the conclusion of the protocol) and then abort. Next, it can check whether the output learned from its view matches the honest party's output; if so it outputs *ideal* and if not *real*. The simulator who receives the honest party's output must simulate the view using this output bit without knowing whether the environment will abort or not; in case of abort, the simulator must equivocate the output obtained from the committed view by the environment to be a random bit—a task that cannot be achieved in the standard model.

Although our proof technique and overall reasoning are very different from that in [6], the source of the impossibility in both cases is the fact that TLPs are non-equivocable. Such equivocality turns out to be essential in both simulation arguments, despite the inherent difference of the primitives and the statements themselves. For example, as the impossibility of [6] relies on Cleve's impossibility [23], the attack applies even with static corruptions; further, when considering the multiparty setting it is oblivious to the underlying network (e.g., it applies even given a broadcast channel). In contrast, in our setting, the attack crucially relies on the adaptive and rushing capabilities of the adversary, and is very sensitive to the underlying network assumptions (e.g., the attack no longer holds in the atomic-multisend model).

Organization of the paper. Section 2 presents the model and the cryptographic primitives that are used in this paper. In Section 3 we present the property-based and simulation-based definitions of broadcast and of unfair broadcast. In Section 4 we analyze the property-based definition, presenting the impossibility result and the protocol construction from time-lock puzzles. Finally, Section 5 treats the simulation-based definition, separating it from the property-based definition, and showing how to realize it in the programmable random-oracle model.

2 Preliminaries

2.1 The Model

An n-party protocol $\pi = (P_1, \ldots, P_n)$ is an n-tuple of interactive Turing machines (ITMs). The term *party* P_i refers to the i^{th} ITM; we denote the set of parties by $\mathcal{P} = \{P_1, \ldots, P_n\}$. Each party P_i starts with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. Without loss of generality, the input length of each party is assumed to be the security parameter κ. We consider protocols that additionally have a setup phase (used, e.g., to model a public-key infrastructure (PKI)) where a trusted dealer samples (possibly correlated) secret values $(\mathtt{r}_1, \ldots, \mathtt{r}_n) \leftarrow D_\pi$ from some efficiently sampleable distribution D_π, and hands party P_i the secret string $\mathtt{r}_i$ (referred to as the correlated randomness of P_i). While our lower bounds hold with respect to any distribution for correlated randomness, our upper bounds rely on a weaker setup assumption of a PKI for digital signatures, where each party generates a pair of signing/verification keys and publishes its verification key.

An *adversary* $\mathcal{A}$ is another ITM describing the behavior of the corrupted parties. It starts the execution with input that contains the security parameter (in unary) and an additional auxiliary input. At any time during the execution of the protocol the adversary can corrupt one of the honest parties, in which case the adversary can read its internal state (containing its input, random coins, correlated randomness, and incoming messages) and gains control over it. A t-adversary is limited to corrupt up to t parties.

The parties execute the protocol over a fully connected synchronous network of point-to-point channels. That is, the execution proceeds in rounds: Each round consists of a *send phase* (where parties send their messages from this round) followed by a *receive phase* (where they receive messages from other parties). The adversary is assumed to be *rushing*, which means that it can see the messages the honest parties send in a round before determining the messages that the corrupted parties send in that round. The communication lines between the parties are assumed to be ideally authenticated (and thus the adversary cannot modify messages sent between two honest parties but can read them).

Throughout the execution of the protocol, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. In our positive results, the adversary is considered to be actively malicious, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. Our lower bounds, however, only rely on fail-stop adversaries that can crash parties, but not cheat in any other way. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties do not output anything and the adversary outputs an (arbitrary) function of its view of the computation (containing the views (internal states) of the corrupted parties).

Atomic multisend. A subtle point that is central to this work is the capabilities of the adversary when corrupting a party that has just sent its messages for the round. Two central models are considered in the literature:

- In the *atomic multisend* model [37] a message that has been sent to the network is guaranteed to be delivered to its recipients even if the sender becomes corrupted shortly after sending; further, the messages are sent to the network as an atomic operation in the sense that once the sender begins sending its messages for the round it cannot become corrupted until it has finished sending all of its messages for the round. This model has gained popularity in many recent consensus protocols (e.g., [22, 1, 21, 70, 8]).
- In the standard (*non-atomic multisend*) model, the operation of sending messages to the channel is not atomic, and the adversary may corrupt a sender *after* it sent its message to some party P_i and *before* it has sent its message to another party P_j; further, the adversary can drop the message the newly corrupted sender sent to P_i and replace it with another. This is the model that has been used in classical models of distributed computation (e.g., [32, 27, 33, 31]) and cryptographic protocols [18, 15, 16, 20]. This models has also been referred to as *strongly adaptive* [45, 50, 54, 46] and *strongly rushing* [2, 1, 69].

In this work we consider the non-atomic-multisend model. Clearly, this is the preferred one as it requires less assumptions on the underlying communication network. However, this model is more challenging to work with as it considers more powerful adversaries; indeed, certain impossibility results in the non-atomic multisend model do not translate to the atomic-multisend realm [48, 12, 1].

In fact, as proven by Katz et al. [51], atomic multisend is a strictly weaker model facing dishonest-majority as it cannot be realized from the basic ingredients needed for synchronous communication (bounded-delay channels and a synchronizing clock).

Secure data erasures. Two models are normally considered in the adaptive-corruption setting, depending on the ability of honest parties to securely erase certain parts of their memory (i.e., from their internal state) without leaving any trace; see [18, 15] for a discussion. While some impossibility results of adaptively secure cryptographic protocols crucially rely on parties *not* being able to erase any information, and completely break otherwise (e.g., [61, 41, 47, 39]), other impossibility results are stronger and do not rely on the absence of secure erasures (e.g., [48, 52, 12, 26]).

In this work we do not assume secure erasures for our protocol constructions; however, our impossibility results hold even in the secure-erasures model. This makes for the strongest statements; to avoid confusion we will state the model explicitly in each section.

2.2 Simulation-Based Security

Some of the results in this work consider a simulation-based definition of broadcast, where security is defined via the real vs. ideal paradigm. Namely, a protocol is considered secure if every attack that can be executed by a PPT adversary in the real-world execution, can be simulated by a PPT simulator in an ideal world, where an incorruptible trusted third party (aka, the *ideal functionality*) receives inputs from the parties and carries out the computation on their behalf. For the specific task of broadcast, the trusted party receives the input from the broadcaster and delivers it to all other parties (see Section 3.2).

We present our results in a *synchronous* model with an online distinguisher (aka, the *environment*); this is the prevalent model in many frameworks for cryptographic protocols; see, e.g., [16, 62, 49, 55, 4, 58, 5, 6]. Such a model requires the simulator to report its view to the distinguisher in every round.[3] We do not rely on any other specific properties of the model, but for concreteness, we state our results in the synchronous model of the UC framework as defined in [51, 55, 5].

Loosely speaking, we consider protocols that run in a hybrid model where parties have access to a simple "clock" functionality $\mathcal{G}_{\mathsf{clock}}$. This functionality keeps a counter, which is incremented once *all honest parties* request the functionality to do so, i.e., once all honest parties have completed their operations for the current round. In addition, all communication is done over bounded-delay channels, where each party requests the channel to fetch messages that are sent to him, such that the adversary is allowed to delay the message delivery by a bounded and *a priori* known number of fetch requests. Stated differently, once the sender has sent some message, it is guaranteed that the message will be delivered within a known number of activations of the receiver. For simplicity, we assume that every message is delivered within a single fetch request.

We note that when considering online distinguishers, a resource-restricted adversary may bypass its limitations by delegating some of its computation to the environment. It is therefore standard to restrict the resources of the environment as well, see e.g., [36]. In this work, when considering a resource-restricted adversary in the simulation-based setting, we will consider the pair of an adversary and an environment as resource restricted, in the sense the their joint resource is bounded.

To simplify the presentation we describe the functionalities and protocols in a less technical way than standard UC formulations (e.g., we do not explicitly mention the session id and party id in every message, and somewhat abuse the activation policy by batching several operations together).

[3] Note that the stand-alone version of synchronous protocols (e.g., [18, 15, 42]) considers offline distinguishers and only requires the simulator to provide an indistinguishable view from the environment *after* the completion of the protocol.

2.3 Time-Lock Puzzles

Time-lock puzzles [67] enable a sender to "lock" its message in a way that "unlocking" requires an inherently sequential computation. This is a powerful primitive that has led to many results, and has been extensively studied; see, e.g., [9, 36, 59, 7, 57, 29, 60, 13, 68, 53, 69, 30, 6]. While the standard definition requires the puzzle generation to be "lightweight" compared to solving the puzzle, our feasibility results can be based on the weaker notion in which puzzle generation is as computationally expensive as solving the puzzle (yet, as opposed to puzzle solving, the puzzle generation is parallelizable). Such weak time-lock puzzles are known from the minimal assumption of one-way functions [59, 7] and the existence of non-parallelizing languages. In this paper we follow the formulation of Bitansky et al. [7].

Puzzles. A puzzle is associated with a pair of parameters: A security parameter κ determining the cryptographic security of the puzzle, as well as a difficulty parameter T that determines how difficult it is to solve the puzzle.

Definition 1 (Puzzle). *A* puzzle *is a pair of algorithms* $(\mathsf{PGen}, \mathsf{PSol})$ *satisfying the following requirements.*

- Syntax:
  - $Z \leftarrow \mathsf{PGen}(T, s)$ *is a probabilistic algorithm that takes as input a difficulty parameter* T *and a solution* $s \in \{0,1\}^\kappa$*, where* κ *is a security parameter, and outputs a puzzle* Z*.*
  - $s = \mathsf{PSol}(Z)$ *is a deterministic algorithm that takes as input a puzzle* Z *and outputs a solution* s*.*
- Completeness: *For every security parameter* κ*, difficulty parameter* T*, solution* $s \in \{0,1\}^\kappa$ *and puzzle* Z *in the support of* $\mathsf{PGen}(T, s)$*,* $\mathsf{PSol}(Z)$ *outputs* s*.*
- Efficiency:
  - $Z \leftarrow \mathsf{PGen}(T, s)$ *can be computed in time* $\mathsf{poly}(\log T, \kappa)$*.*
  - $\mathsf{PSol}(Z)$ *can be computed in time* $T \cdot \mathsf{poly}(\kappa)$*.*

Time-lock puzzles. In a time-lock puzzle, we require that the parallel time required to solve a puzzle is proportional to the time it takes to solve the puzzle honestly, up to some fixed polynomial loss.

Definition 2 (Time-lock puzzle). *A puzzle* $(\mathsf{PGen}, \mathsf{PSol})$ *is a* time-lock puzzle with gap $\varepsilon < 1$ *if there exists a polynomial* $T_1(\cdot)$*, such that for every polynomial* $T(\cdot) \geq T_1(\cdot)$ *and every polysize adversary* $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa \in \mathbb{N}}$ *of depth* $\mathsf{depth}(\mathcal{A}_\kappa) \leq T^\varepsilon(\kappa)$*, there exists a negligible function* μ*, such that for every* $\kappa \in \mathbb{N}$*, and every pair of solutions* $s_0, s_1 \in \{0,1\}^\kappa$*:*

$$\Pr\Big[b \leftarrow \mathcal{A}_\kappa(Z) \;\Big|\; b \leftarrow \{0,1\}, Z \leftarrow \mathsf{PGen}(T, s_b)\Big] \leq 1/2 + \mu(\kappa).$$

Definition 3 (Weak puzzle). *A* weak puzzle *is a pair of algorithms* $(\mathsf{PGen}, \mathsf{PSol})$ *satisfying the* Syntax *and* Completeness *requirements as per Definition 1, and the following* weak efficiency *requirement.*

- Weak Efficiency:

- $Z \leftarrow \mathsf{PGen}(T, s)$ *can be computed by a uniform circuit of size* $\mathsf{poly}(T, \kappa)$ *and depth* $\mathsf{poly}(\log T, \kappa)$.
- $\mathsf{PSol}(Z)$ *can be computed in time* $T \cdot \mathsf{poly}(\kappa)$.

Mahmoody et al. [59] showed how to construct a weak time-lock puzzle in the random-oracle model while Bitansky et al. [7] showed how to construct it from any one-way function and non-parallelizing language.

Definition 4 (**Non-parallelizing language**). *A language* $\mathcal{L} \in \mathsf{DTime}(T(\cdot))$ *is* non-parallelizing with gap $\varepsilon < 1$ *if for every family of non-uniform polysize circuits* $\mathcal{B} = \{\mathcal{B}_\kappa\}_{\kappa \in \mathbb{N}}$ *where* $\mathsf{depth}(\mathcal{B}_\kappa) \leq T^\varepsilon(\kappa)$ *and every large enough* κ, $\mathcal{B}_\kappa$ *fails to decide* $\mathcal{L}_\kappa = \mathcal{L} \cap \{0,1\}^\kappa$.

Theorem 5. *[7] Let* $\varepsilon < 1$. *Assume that one-way functions exist, and that for every polynomially bounded function* $T(\cdot)$ *there exists a non-parallelizing language* $\mathcal{L} \in \mathsf{DTime}(T(\cdot))$ *with gap* ε. *Then, for any* $\varepsilon_1 < \varepsilon$ *there exists a weak time-lock puzzle with gap* ε_1.

3 Broadcast Protocols: Definitions

Intuitively, a broadcast protocol should emulate a "megaphone" functionality in the sense that when the sender speaks, all recipients receive the sender's message. This is traditionally captured via the *agreement* and *validity* properties. However, as observed in Hirt and Zikas [48], such a property-based definition falls short of capturing the ideal megaphone functionality when facing adaptive corruptions. Namely, the ideal megaphone functionality does not allow the adversary to corrupt the sender after learning its input message, and change it retrospectively. Hirt and Zikas [48] further showed that the ideal megaphone functionality cannot be realized in the dishonest-majority setting in the standard (non-atomic-multisend) communication model.

3.1 Property-Based Broadcast

With the goal of distilling the essence of the impossibility result in [48], we provide a weaker, property-based definition that is complete in the presence of adaptive corruptions. Recall that when broadcasting a random bit via an "unfair" broadcast (following [48] terminology), the adversary gets to learn the input bit *before* deciding whether to corrupt the sender and change its input; for example, the adversary may corrupt the sender when the input is 1 and flip it to 0, but when the input is 0 the adversary may continue without corrupting the sender. Informally, a broadcast protocol should not concede this capability to the adversary.

Without loss of generality, we consider the message space to be $\{0,1\}^\kappa$. Looking ahead, our lower bounds hold even in the simpler, Boolean case where the message space is $\{0,1\}$, while our upper bounds hold for any polynomial-length messages.

Definition 5 (**Broadcast, property-based definition**). *An* n*-party protocol* π, *where a distinguished sender holds an initial input message* $m \in \{0,1\}^\kappa$, *is a* broadcast protocol *(according to the property-based definition) tolerating an adaptive PPT* t*-adversary, if the following conditions are satisfied:*

- ***Termination:*** *There exists an a-priori-known round* R *such that the protocol is guaranteed to complete (i.e., every so-far honest party produces an output value) within* R *rounds.*
- ***Agreement:*** *All honest parties (at the end of the protocol) output the same value, with all but negligible probability.*

- ***Validity:*** *If the sender is honest (at the end of the protocol) then all honest parties (at the end of the protocol) output m, with all but negligible probability.*
- ***Fairness:*** *For every PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}}(\kappa) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\kappa),$$

where the experiment $\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}}(\kappa)$ is defined in Figure 1.

Experiment $\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}}(\kappa)$

1. The challenger samples a uniformly random bit $b \leftarrow \{0, 1\}$ and invokes $\mathcal{A}$ on input 1^κ.
2. The challenger samples correlated randomness $(\mathtt{r}_1, \ldots, \mathtt{r}_n) \leftarrow D_\pi$ and simulates the protocol π on sender-input b^κ toward $\mathcal{A}$, who can adaptively corrupt parties throughout the execution. (The challenger simulates all honest parties, and upon a corruption request reveals the internal state of the corrupted party to the adversary, as well as the control over that party.)
3. The output of the experiment is set to 1 if:
   - $b = 0$ and the sender is honest at the end of the protocol;
   - $b = 1$ and the output value of an arbitrary honest party is 0^κ.

   Otherwise, the output of the experiment is set to 0. (If all parties are corrupted, the output is set to be 0.)

Figure 1: The fairness-property experiment for adaptively secure broadcast

Note that, as observed in [48], the protocol of Dolev and Strong [27] (as well as most broadcast protocols in the literature) allows an adversary to first learn the sender's input message m, and later change the common output as a function of m. Therefore, this protocol does not satisfy the *fairness* property (even in the atomic-multisend model [37]). The broadcast protocols from [48, 24, 25] satisfy this property for $t \leq n/2$ in the standard (non-atomic-multisend) model, and similarly, the protocol from [37] for $t < n$ in the atomic-multisend model.

We shall refer to the commonly used property-based definition of broadcast as *unfair broadcast*.

Definition 6 (Unfair broadcast, property-based definition). *An n-party protocol π tolerating an adaptive PPT t-adversary, is an unfair broadcast protocol if* agreement, validity *and* termination *hold, but* fairness *does not necessarily hold.*

3.2 Simulation-Based Broadcast

While the property-based definitions provide the core requirements of broadcast, they are weaker than simulation-based definitions and are therefore more suitable for lower bounds. We next present the stronger simulation-based definitions which are better suited for proving the security of protocol constructions.

Definition 7 (Broadcast, simulation-based definition). *An n-party protocol π, is a broadcast protocol (according to the simulation-based definition) tolerating an adaptive PPT t-adversary, if π securely realizes the broadcast functionality, defined in Figure 2.*

The functionality $\mathcal{F}_{\mathsf{bc}}$

- **Initialization:** The functionality initializes the output message $m_{\mathsf{out}} := \perp$ and a Boolean flag $\mathsf{isOutputLocked} := \mathsf{false}$.
- **Input:** The sender sends an input message $m \in \{0,1\}^\kappa$. The functionality sets the output message $m_{\mathsf{out}} := m$.
- **Output request:** If the adversary asks to receive the output value and there exists at least one corrupted party, the functionality hands the adversary the message m_{out} and sets $\mathsf{isOutputLocked} := \mathsf{true}$. If all parties are honest, the functionality ignores this request.
- **Corruption request:** If the adversary corrupts the sender, the functionality hands the adversary the message m_{out}. The adversary can provide the functionality a message m' and if $\mathsf{isOutputLocked} = \mathsf{false}$, the functionality sets the output message to be $m_{\mathsf{out}} := m'$.
- **Output:** The functionality sends m_{out} as output to all parties and sets $\mathsf{isOutputLocked} := \mathsf{true}$.

Figure 2: The broadcast functionality

Next, we provide the simulation-based definition of unfair broadcast, where the adversary can first learn the message and later corrupt the sender and replace its message.

Definition 8 (Unfair broadcast, simulation-based definition). *An n-party protocol π, is an unfair broadcast protocol (according to the simulation-based definition) tolerating an adaptive PPT t-adversary, if π securely realizes the unfair broadcast functionality, defined in Figure 3.*

The functionality $\mathcal{F}_{\mathsf{ubc}}$

- **Input:** The sender sends an input message $m \in \{0,1\}^\kappa$. The functionality sets the output value $m_{\mathsf{out}} := m$ and sends m to the adversary.
- **Corruption request:** If the adversary corrupts the sender, the adversary can provide the functionality a message m' and if no honest party received the output yet, the functionality sets the output message to be $m_{\mathsf{out}} := m'$.
- **Output:** The functionality sends m_{out} as output to all parties.

Figure 3: The unfair-broadcast functionality

As a sanity check, we prove that a protocol that satisfies the simulation-based definition (Definition 7) also satisfies the property-based definition (Definition 5).

Lemma 9. *If an n-party protocol π is a broadcast protocol according to the simulation-based definition tolerating an adaptive PPT t-adversary, then π is a broadcast protocol according to the property-based definition tolerating an adaptive PPT t-adversary.*

Proof. Assume that π satisfies Definition 7 but does not satisfy Definition 5. If *termination*, *agreement*, or *validity* are not satisfied, then we immediately derive a contradiction; therefore, we assume that the *fairness* property is not satisfied. This means that there exists a PPT adversary $\mathcal{A}'$ that can win $\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}'}(\kappa)$ (i.e., bias the output to 0^κ while keeping the sender honest given 0^κ as input) with probability $1/2 + \mu(\kappa)$ for a non-negligible μ.

Consider the environment $\mathcal{Z}$ and adversary $\mathcal{A}$ defined as follows: First, $\mathcal{Z}$ chooses a random bit $b \leftarrow \{0,1\}$ and activates the sender with input b^κ. Next, the $\mathcal{A}$ attacks the protocol execution by proceeding according to the operations of $\mathcal{A}'$. Finally, the environment checks whether when $b = 0$ the sender remained honest, and when $b = 1$ the output of an arbitrary honest party is 0^κ;

if so, the environment outputs 1 (real) and otherwise 0 (ideal). By construction, the environment outputs 1 when interacting with the real-world protocol with probability $1/2 + \mu(\kappa)$.

By Definition 7, there exists a PPT simulator $\mathcal{S}$ that can simulate this attack. However, by definition of $\mathcal{F}_{\mathsf{bc}}$, the simulator can exactly simulate one of the two actions below, but not both.

- Learn the input without corrupting the sender; in this case if the input is 1^κ the adversary can no longer influence the output to be 0^k.
- Corrupt the sender without first learning the input value and set the output to be 0^κ; in this case if the input is 0^κ the sender does not remain honest.

It follows that for any PPT simulator, the environment will output 1 when interacting with the ideal-world computation with probability $1/2 + \mathsf{negl}(\kappa)$. This leads to a contradiction to the assumption that π satisfies Definition 7. $\square$

4 Property-Based Adaptively Secure Broadcast

In this section we analyze the property-based definition of adaptively secure broadcast. In Section 4.1 we extend the impossibility result of Hirt and Zikas [48] to this regime, and in Section 4.2 we show how to overcome this impossibility using resource-restricted cryptography; namely, via time-lock puzzles.

First we observe that the impossibility proof of Hirt and Zikas [48, Lemma 8] makes an implicit assumption that for an invocation of broadcast with sender P_s, the adversary, is aware of the first subset of $\mathcal{P} \setminus \{P_s\}$ of size $t-1$, which receives information about the input that P_s is attempting to broadcast, and the actual round in which this occurs. An analogue of this property can also be defined for computationally secure protocols, where information might be available to a set but *computationally inaccessible*. In fact, it is easy to verify that all known broadcast protocols have such a "release" round, which is not only defined, but also publicly known by the protocol structure. For instance, in common protocols where the sender sends its input to everyone in the first round (e.g., [27, 35]), the first round is actually this public round. We will denote the class of protocols with such as step-release structure as $\Pi_{\mathsf{step\text{-}rel}}$. In the following, we formally specify this class and prove our impossibility results for all protocols that satisfy it.

For any given protocol π in the correlated-randomness model, any subset of parties $\hat{\mathcal{P}} \subseteq \mathcal{P}$, and any round ρ, let $\textsc{view}^{\rho}_{\pi,\hat{\mathcal{P}}}(x,\kappa)$ denote the joint view of the parties in $\hat{\mathcal{P}}$ at the beginning of round ρ in an honest execution (i.e., without the adversary corrupting anyone) on sender-input x, where κ is the security parameter. In particular, $\textsc{view}^{1}_{\pi,\hat{\mathcal{P}}}(\cdot)$ consists of the inputs and the setup (including randomness) of all parties in $\hat{\mathcal{P}}$ at the beginning of the protocol (before any message is exchanged). For simplicity—in order to capture also randomized protocols with non-simultaneous termination—we will allow the view to be defined even after a party terminates: if for some $P \in \hat{\mathcal{P}}$, party P terminates in some round $\rho \leq R$ (where R is the upper bound of the protocol's round complexity guaranteed to exist by the *termination* property of Definition 5), then $\textsc{view}^{R}_{\pi,\hat{\mathcal{P}}}(\cdot)$ includes the view of this party up to termination (round R). We will also assume for simplicity (again without loss of generality) that for any such party, its view includes the party's output.

The definition of the class $\Pi_{\mathsf{step\text{-}rel}}$ ensures that a round $\hat{r}_\pi$ and a set $\hat{\mathcal{P}}_\pi \subseteq \mathcal{P} \setminus \{P_s\}$ of size $|\hat{\mathcal{P}}_\pi| < \lfloor n/2 \rfloor$ are defined by the protocol, such that the set $\hat{\mathcal{P}}_\pi$ is the first set of parties that are able to learn the actual input and this happens in round $\hat{r}_\pi$; i.e., no other set of parties (of the same size) is able to output the input of the sender based on its view from rounds $1, \ldots, \hat{r}_\pi - 1$. Formally:

Definition 10 (The protocol class $\Pi_{\mathsf{step\text{-}rel}}$). *For any protocol π in the class $\Pi_{\mathsf{step\text{-}rel}}$, there exists some round number $\hat{r}_\pi$, a set $\hat{\mathcal{P}}_\pi \subseteq \mathcal{P} \setminus \{P_s\}$ of size $|\hat{\mathcal{P}}_\pi| < \lfloor n/2 \rfloor$, and a PPT algorithm $\hat{B}_\pi$ such that the following properties hold:*

1. *There exists a negligible function ν such that for any input x it holds that*
$$\Pr\left[\hat{B}_\pi\left(\text{VIEW}^{\hat{r}_\pi}_{\pi,\hat{\mathcal{P}}_\pi}(x,\kappa)\right) = x\right] \geq 1 - \nu(\kappa).$$

2. *Let D be the input domain of the broadcast protocol (the set of possible inputs). If the input x is chosen uniformly at random from D, then the output of the honest parties in the following experiment is $y \neq x$ with noticeable probability:*
   (a) *Initiate the protocol π with sender P_s receiving a uniformly distributed input $x \leftarrow D$, and sample and distribute the correlated randomness according to π.*
   (b) *Consider a fail-stop adversary that corrupts the parties in $\hat{\mathcal{P}}_\pi \cup \{P_s\}$ in round $\hat{r}_\pi$ and crashes them before sending their round-$\hat{r}_\pi$ messages.*
   (c) *Have the honest parties complete their protocol and set y to the output of any honest party (e.g., the one with the smallest index.)*

4.1 Impossibility of Property-Based Adaptively Secure Broadcast

We start by adapting the impossibility result of Hirt and Zikas [48] to work with the property-based definition. In particular, we present a simpler argument than [48] that extends the impossibility to: (1) capture a smaller, Boolean input domain (as opposed to exponential-size domain in [48]), and (2) we show the impossibility with respect to a property-based definition (as opposed to the simulation-based definition in [48]). We also observe that this proof strategy works both for deterministic and randomized protocols assuming any correlated-randomness setup and/or secure data erasures. We note that by Lemma 9 an impossibility of a broadcast protocol according to the property-based definition also rules out such protocols secure according to the simulation-based definition.

Theorem 1. *Let $t > n/2$. Then, there exists no broadcast protocol in the class $\Pi_{\mathsf{step\text{-}rel}}$ (secure according to the property-based definition) tolerating an adaptive, fail-stop PPT t-adversary. The theorem holds both for deterministic and randomized protocols assuming any correlated-randomness setup and/or secure erasures.*

Proof. Without loss of generality, we prove this statement for Boolean broadcast. Assume towards a contradiction that π is a Boolean broadcast protocol (according to the property-based definition) with sender P_s tolerating an adaptive PPT t-adversary. By classical impossibility results [56, 33, 11], if $t \geq n/3$ then π cannot be defined in the plain model (even assuming standard cryptographic hardness assumptions), and some form of setup is required. That is, we consider a trusted dealer that samples correlated randomness $(\mathbf{r}_1, \ldots, \mathbf{r}_n) \leftarrow D_\pi$ from some efficiently sampleable distribution D_π, and privately hands each P_i the string $\mathbf{r}_i$. Without loss of generality, assume that the random coins used by each party are defined within $\mathbf{r}_i$, so the transcript and the view of each party are random variables over the probability space defined by the random coins used for sampling from D_π and by the random choice of the input bit $x \leftarrow \{0,1\}$.

By Definition 10 there exist a round number $\hat{r}_\pi$, a set $\hat{\mathcal{P}}_\pi$, and an algorithm $\hat{B}_\pi$ for the protocol π. The adversary $\mathcal{A}$ that breaks the fairness property of Definition 5 is defined as follows: Let $T(\cdot)$ denote the polynomial that bounds the running time of the protocol.

1. For rounds $1, \ldots, \hat{r}_\pi - 1$ the adversary corrupts no party.

2. At the beginning of round $\hat{r}_\pi$: $\mathcal{A}$ corrupts all the parties in $\hat{\mathcal{P}}_\pi$, and uses its rushing ability to deliver all the messages that parties outside of $\hat{\mathcal{P}}_\pi$ send to the corrupted parties. This way, $\textsc{view}^{\hat{r}_\pi}_{\pi,\hat{\mathcal{P}}_\pi}$ includes all messages that are in those parties' view of an honest execution at round $\hat{r}_\pi$.
3. $\mathcal{A}$ computes $\hat{y} \leftarrow \hat{B}_\pi(\textsc{view}^{\hat{r}_\pi}_{\pi,\hat{\mathcal{P}}_\pi})$ and proceeds as follows:
   - **if** $\hat{y} = 1$, then $\mathcal{A}$ corrupts also the sender P_s and crashes all corrupted parties—so that the last messages received by parties in $\mathcal{P} \setminus (\hat{\mathcal{P}}_\pi \cup \{P_s\})$ were the ones received at round $\hat{r}_\pi - 1$.
   - **else** (i.e., if $\hat{y} = 0$) the adversary does not corrupt P_s and allows all parties to continue playing their protocol.

Next, we show that adversary $\mathcal{A}$ violates the fairness property of Definition 5. In slight abuse of notation, but without loss of generality, we will use $\mathcal{P}$ to denote the broadcast-protocol parties simulated by the Challenger in $\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}}(\kappa)$, and denote the sender by P_s. First we observe that the input b that the Challenger uses in its simulation of broadcast towards $\mathcal{A}$ is distributed uniformly. Hence, the views that $\mathcal{A}$ and Challenger witness in the fairness experiment up to round $\hat{r}_\pi$ are distributed identically as in an honest execution of π, and, therefore, the properties of the class $\Pi_{\mathsf{step\text{-}rel}}$ hold for the interaction between them.

We consider the following events in the simulated execution of π between the Challenger and $\mathcal{A}$:

- $\mathcal{E}_{\mathsf{b=0}}$ occurs when the Challenger chooses $b = 0$.
- $\mathcal{E}_{\mathsf{b=1}}$ occurs when the Challenger chooses $b = 1$.
- $\mathcal{E}_{\mathtt{h}}$ occurs if the sender P_s is honest at the end of the simulated protocol execution.
- $\mathcal{E}_{\mathsf{c}}$ occurs if the sender P_s gets corrupted before the end of the simulated protocol execution.
- $\mathcal{E}_{\mathtt{flip}}$ occurs if on sender input x, some party that is honest until it terminates, outputs $y = 1-x$ in the simulated execution. (Note that by *agreement* this implies that all honest parties will output y except with negligible probability.)

Note that the events $\mathcal{E}_{\mathsf{b=0}} \wedge \mathcal{E}_{\mathtt{h}}$ and $\mathcal{E}_{\mathsf{b=1}} \wedge \mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}}$ are disjoint, therefore:

$$\begin{aligned}\Pr\left[\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}} = 1\right] &= \Pr\Big[(\mathcal{E}_{\mathsf{b=0}} \wedge \mathcal{E}_{\mathtt{h}}) \vee (\mathcal{E}_{\mathsf{b=1}} \wedge \mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}})\Big] \\ &= \Pr\Big[(\mathcal{E}_{\mathsf{b=0}} \wedge \mathcal{E}_{\mathtt{h}})\Big] + \Pr\Big[(\mathcal{E}_{\mathsf{b=1}} \wedge \mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}})\Big]. \qquad (1)\end{aligned}$$

Next, we observe that when the input bit is $b = 0$, then the adversary $\mathcal{A}$ corrupts the sender and flips the output bit only with negligible probability (i.e., the probability that $\hat{B}_\pi$ outputs the wrong value 1, which is negligible for the protocols in the class we are considering). Hence:

$$\Pr\Big[(\mathcal{E}_{\mathsf{b=0}} \wedge \mathcal{E}_{\mathtt{h}})\Big] = \Pr\Big[\mathcal{E}_{\mathtt{h}} \mid \mathcal{E}_{\mathsf{b=0}}\Big] \cdot \Pr\Big[\mathcal{E}_{\mathsf{b=0}}\Big] = \frac{1}{2} \cdot \Pr\Big[\mathcal{E}_{\mathtt{h}} \mid \mathcal{E}_{\mathsf{b=0}}\Big] \geq \frac{1}{2} \cdot (1 - \nu(\kappa)), \qquad (2)$$

for some negligible function $\nu(\cdot)$.

Similarly, by the second property of the protocol class $\Pi_{\mathsf{step\text{-}rel}}$, the above adversary makes the honest parties output $1 - b$ with noticeable probability. Hence:

$$\begin{aligned}\Pr\Big[\mathcal{E}_{\mathsf{b=1}} \wedge \mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}}\Big] &= \Pr\Big[\mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}} \mid \mathcal{E}_{\mathsf{b=1}}\Big] \cdot \Pr\Big[\mathcal{E}_{\mathsf{b=1}}\Big] \\ &= \frac{1}{2} \cdot \Pr\Big[\mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}}\Big] \\ &\geq \frac{1}{2} \cdot \Big(1 - \mu(\kappa)\Big) \cdot q(\kappa), \qquad (3)\end{aligned}$$

for some noticeable function $q : \mathbb{N} \to [0, 1]$ and some negligible function $\mu(\cdot)$.

Putting the above together we get that:

$$\begin{aligned}\Pr\left[\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}} = 1\right] &= \Pr\Big[(\mathcal{E}_{\mathsf{b=0}} \wedge \mathcal{E}_{\mathtt{h}})\Big] + \Pr\Big[(\mathcal{E}_{\mathsf{b=1}} \wedge \mathcal{E}_{\mathsf{c}} \wedge \mathcal{E}_{\mathtt{flip}})\Big] \\ &\geq \frac{1}{2} \cdot (1 - \nu(\kappa)) + \frac{1}{2} \cdot \Big(1 - \mu(\kappa)\Big) \cdot q(\kappa) \\ &= \frac{1}{2} + \frac{q(\kappa)}{2} - \frac{\nu(\kappa) + \mu(\kappa) \cdot q(\kappa)}{2}. \end{aligned} \tag{4}$$

Since $q(\kappa)$ is not negligible, nor is $q(\kappa)/2$. Furthermore, since $\mu(\kappa)$ and $\nu(\kappa)$ are negligible and $q(\cdot) \leq 1$, it holds that $\mu(\kappa) \cdot q(\kappa)$ is also negligible; hence, so is $\nu(\kappa) + \mu(\kappa) \cdot q(\kappa)$. Therefore,

$$\frac{q(\kappa)}{2} - \frac{\nu(\kappa) + \mu(\kappa) \cdot q(\kappa)}{2}$$

is a noticeable function, which means that Equation 4 contradicts the fairness property of the broadcast definition. □

4.2 Property-Based Adaptively Secure Broadcast Protocol

Next, we proceed to show that the property-based definition of broadcast can be realized assuming a time-lock puzzle. The high-level idea is quite simple. The sender hides its message inside a (weak) time-lock puzzle, and uses an unfair broadcast protocol (e.g., Dolev and Strong [27]) to deliver the puzzle to all parties. The TLP parameters should guarantee that the adversary cannot solve the puzzle before the unfair broadcast completes.

In the spirit of resource-restricting cryptography, we will not consider arbitrary PPT adversaries, since otherwise the impossibility results from Section 4.1 will kick in. Instead we will assume an upper bound on the number of parallel steps an adversary can perform during the protocol's execution.

Definition 11 (**(R, T)-bounded adversary**)**.** *A PPT adversary $\mathcal{A}$ is (R, T)-bounded if for every $\kappa \in \mathbb{N}$, the maximal depth of a circuit that $\mathcal{A}$ can evaluate within R communication rounds is bounded by $T(\kappa)$.*

Theorem 2. *Let $t \leq n$ and let $T(\cdot)$ be a polynomial. Assume that weak time-lock puzzles with gap $\varepsilon < 1$ exist and that unfair broadcast can be computed in R rounds against an adaptive PPT t-adversary. Then, Protocol $\pi_{\mathsf{bc\text{-}prop}}$ (Figure 4) is a broadcast protocol (according to Definition 5) that is secure against an (R, T)-bounded adaptive PPT t-adversary.*

Protocol $\pi_{\mathsf{bc\text{-}prop}}(T, \kappa)$

- **Hybrid model:** The protocol is defined in the unfair broadcast $\mathcal{F}_{\mathsf{ubc}}$-hybrid model, where $\mathcal{F}_{\mathsf{ubc}}$ produces an output within R rounds.
- **Public parameters:** A puzzle $(\mathsf{PGen}, \mathsf{PSol})$ with gap $\varepsilon < 1$, a difficulty parameter T, and the security parameter κ.
- **Private input:** The sender has a private input $m \in \{0, 1\}^\kappa$.
- **The protocol:**
  - **Lock:** The sender computes $Z \leftarrow \mathsf{PGen}(T^{1/\varepsilon}, m)$.
  - **Unfair broadcast:** The sender broadcasts Z via $\mathcal{F}_{\mathsf{ubc}}$.
  - **Recover the output:** Upon receiving Z, each party computes $m = \mathsf{PSol}(Z)$ and outputs m.

Figure 4: Adaptively secure, property-based, broadcast protocol

Proof. We prove each property separately.

Termination. Every honest party is guaranteed to receive a puzzle from $\mathcal{F}_{\mathsf{ubc}}$ within R rounds, and therefore is guaranteed to produce an output and terminate after a polynomial number of steps needed to open the puzzle.

Agreement. By definition of $\mathcal{F}_{\mathsf{ubc}}$, all honest parties are guaranteed to receive the same Z, and therefore, by the correctness of the time-lock puzzle, are guaranteed to output the same message with overwhelming probability.

Validity. Assume the sender is honest at the end of the execution and is set with input m. By definition of $\mathcal{F}_{\mathsf{ubc}}$, all honest parties are guaranteed to receive Z. Therefore, by the correctness of the time-lock puzzle, all honest parties will compute $m = \mathsf{PSol}(Z)$ with overwhelming probability.

Fairness. Let $\mathcal{A}$ be an (R, T)-bounded PPT adversary that wins $\mathsf{Expt}^{\mathsf{fair\text{-}bcast}}_{\pi,\mathcal{A}}(\kappa)$ with probability $1/2 + \mu(\kappa)$ for a non-negligible μ. Note that the only information that $\mathcal{A}$ receives during the execution of the protocol is the puzzle Z (since the views of non-sender parties that get corrupted contain no information about the input bit), and based on this information $\mathcal{A}$ has to decide whether to corrupt the sender and flip its bit, i.e., decide whether the puzzle Z contains 1^κ. Since $\mathcal{A}$ is (R, T)-bounded and has to decide whether to corrupt the sender within R rounds, the algorithm that $\mathcal{A}$ computes can be represented as a depth $T(\kappa)$ circuit. However, since the difficulty parameter of the puzzle is set to $T^{1/\varepsilon}$, it follows that $\mathcal{A}$ can be used to break the security of the time-lock puzzle, according to Definition 2. $\square$

By instantiating $\mathcal{F}_{\mathsf{ubc}}$ with the protocol of Dolev and Strong [27] we derive the following corollary.

Corollary 12. *Assume that weak time-lock puzzles with gap $\varepsilon < 1$ exist, let $t \leq n$, and let $T(\cdot)$ be a polynomial. Then, there exists a broadcast protocol (according to the property-based definition) that is secure against an (n, T)-bounded adaptive PPT t-adversary, given a PKI for digital signatures.*

5 Simulation-Based Adaptively Secure Broadcast

In this section we analyze the simulation-based definition of broadcast. In Section 5.1 we show that the assumptions used in Section 4.2 that satisfy the property-based definition are not sufficient to realize the simulation-based definition, and in Section 5.2 we show how to overcome the new impossibility via the new notion of non-committing time-lock puzzles.

5.1 Impossibility of Simulation-Based Adaptively Secure Broadcast

We demonstrate that for the class of protocols $\Pi_{\mathsf{step\text{-}rel}}$ (which in particular includes our protocol from the previous section proved secure according to the property-based definition), assuming time-lock puzzles does not help in realizing the simulation-based definition. This demonstrates not only the separation between the two definitions, property-based and simulation-based, but also the fact that time-lock puzzles are less effective in a simulation-based setting. Intuitively, the reason is that the puzzle is a non-interactive object which has a *binding* property (once handed over, its solution cannot be changed) and a *temporary hiding* property (while the solver works to solve the puzzle, they cannot distinguish it from a puzzle with another solution). In fact, once one observes these properties, the limits of the strength of TLPs for simulation-based adaptive security becomes less of a surprise, as it resembles analogous issues displayed by primitives with similar properties, such as commitments [17] and public-key encryption [61].

Before stating our results we first extend the notion of (R,T)-bounded adversaries to the simulation-based setting, where the adversary can use the computational resources of the environment. We consider the pair of environment $\mathcal{Z}$ and adversary $\mathcal{A}$ to be (R,T)-bounded, meaning that for every $\kappa \in \mathbb{N}$, the maximal depth of a circuit that $\mathcal{Z}$ and $\mathcal{A}$ can jointly evaluate within R communication rounds, is bounded by $T(\kappa)$.

We note that by restricting the joint resources of the environment and the adversary, we actually obtain a stronger impossibility result, since even a weaker distinguisher can distinguish between the real execution and the simulated one. Moreover, the result is in fact even stronger since we do not restrict the simulator to be (R,T)-bounded.

We are now ready to state the impossibility result, which states that even TLPs cannot help circumvent the impossibility of adaptively secure broadcast under simulation-based security. Informally, this is proven by using the fact that, by definition of $\Pi_{\mathsf{step\text{-}rel}}$, in round $\hat{r}_\pi$ the adversary attacking π and corrupting $\hat{\mathcal{P}}_\pi$ has all the information it needs to recover the output (even when the sender is honest). This means that, in order to simulate, the simulator needs to give its adversary this information. But the only way the simulator can ensure this is by asking the functionality $\mathcal{F}_{\mathsf{bc}}$ for the sender's input. This gives rise to the following distinguishing strategy for the environment: Once the environment gets its round $\hat{r}_\pi$ messages, it attempts to flip the output by corrupting the sender and all parties in the set $\hat{\mathcal{P}}_\pi$ defined by class $\Pi_{\mathsf{step\text{-}rel}}$. What complicates things is that, unlike the proof of Theorem 1, the environment cannot set a trap for the simulator by making its choice to corrupt the sender depend on the output of $\hat{B}_\pi$. The reason is that the input (to $\hat{B}_\pi$) view of round $\hat{r}_\pi$ might include TLPs, which the environment cannot quickly solve (within round $\hat{r}_\pi$) by the time it decides whether or not to corrupt the sender and try to flip the output.

Instead the environment does the following: It always, optimistically, corrupts the sender and tries to flip the output; it then uses *input-dependent* check-events to distinguish as follows. If the input is 0 the environment checks that the simulator gave it consistent $\hat{r}_\pi$-round messages by running algorithm $\hat{B}_\pi$;[4] otherwise, if the input is 1 then it checks if the simulator managed to flip the bit by looking at the output of $\mathcal{F}_{\mathsf{bc}}$. As discussed above, the only way the simulator can ensure that the first check succeeds is by asking the functionality $\mathcal{F}_{\mathsf{bc}}$ for the input; however, when this happens, the output of $\mathcal{F}_{\mathsf{bc}}$ gets locked which will make it impossible for the simulator to flip the output. Hence, one of the two check events will occur noticeably more frequently in the real than in the ideal world, rendering the protocol insecure. We proceed with formal statement and proof.

Theorem 3. *Let $t > n/2$ and let R, T be polynomials in κ. Then, there exists no broadcast protocol in $\Pi_{\mathsf{step\text{-}rel}}$, which is secure according to the simulated-based definition and tolerates an adaptive, fail-stop, PPT, t-adversary. The theorem holds both for deterministic and randomized protocols assuming any (even inefficient[5]) correlated-randomness setup and/or secure data erasures and holds even for (R,T)-bounded environments and adversaries and assuming time-lock puzzles.*

Proof. Let π be a broadcast protocol from the class $\Pi_{\mathsf{step\text{-}rel}}$. As in the proof of Theorem 1 we consider a (trusted dealer that samples a) correlated-randomness setup $(\mathbf{r}_1, \ldots, \mathbf{r}_n) \leftarrow D_\pi$ from some distribution D_π, and privately hands each P_i the string $\mathbf{r}_i$. Without loss of generality, assume that the random coins used by each party are defined within $\mathbf{r}_i$, so the transcript and the view of

[4] The environment can take its time running $\hat{B}_\pi$ after the protocol terminates.

[5] Classical correlated randomness setup assumes efficient sampling and distribution mechanisms. By removing such restrictions here we can even capture non-programmable random oracle, as an exponential-space correlated randomness functionality that samples the entire random table of the RO.

each party are random variables over the probability space defined by the random coins used for sampling from D_π and by the random choice of the input bit $x \leftarrow \{0,1\}$.

By Definition 10 there exist a round index $\hat{r}_\pi$, a set $\hat{\mathcal{P}}_\pi$, and an algorithm $\hat{B}_\pi$ for the protocol π. One might be tempted to play the same strategy as in the proof of Theorem 1—i.e., at round $\hat{r}_\pi$, the adversary who corrupts the parties in $\hat{\mathcal{P}}_\pi$ evaluates $\hat{B}_\pi$ on their view, and depending on the output of $\hat{B}_\pi$, either corrupts the sender and crashes all corrupted parties, or lets the protocol complete. Unfortunately, this attack might not work in this setting, as the definition of $\hat{B}_\pi$ makes no restriction on the (parallel) time that it takes to compute its output other than that this time is polynomial. Thus, an (R,T)-bounded adversary/environment pair might not be able to evaluate $\hat{B}_\pi$ before round $\hat{r}_\pi$ finishes. This is, for example, the case if the view of the parties in $\hat{\mathcal{P}}_\pi$ includes a freshly generated time-lock puzzle, as in the protocol $\pi_{\textsf{bc-prop}}$ (Figure 4), that requires multiple rounds to be solved by an (R,T)-bounded adversary.

Thus, we need a different strategy for the environment. Consider the following two (R,T)-bounded environments $\mathcal{Z}_0$ and $\mathcal{Z}_1$: $\mathcal{Z}_b$ gives the sender input b with probability 1 and works as follows: It instantiates an execution with a *dummy adversary* [16], namely, an adversary that simply follows the environment's instructions. It then proceeds is as follows:

1. For rounds $1, \dots, \hat{r}_\pi - 1$ it (instructs the adversary to) corrupts no party.
2. At the beginning of round $\hat{r}_\pi$: The environment tells its dummy adversary $\mathcal{A}$ to corrupt all the parties in $\hat{\mathcal{P}}_\pi$, and use its rushing ability to deliver all the messages that parties outside of $\hat{\mathcal{P}}_\pi$ send to the corrupted parties. This way, $\textsc{view}^{\hat{r}_\pi}_{\pi,\hat{\mathcal{P}}_\pi}$ includes all messages that are in those parties' view of an honest execution at round $\hat{r}_\pi$.
3. After receiving all $\hat{r}_\pi$-round messages from all honest parties running the protocol (or from the simulator in the ideal world) $\mathcal{Z}_b$ instructs $\mathcal{A}$ to corrupt also the sender P_s and crash all corrupted parties, including the sender, so that the last messages received by parties in $\mathcal{P} \setminus (\hat{\mathcal{P}}_\pi \cup \{P_s\})$ were the ones received at round $\hat{r}_\pi - 1$.
4. $\mathcal{Z}_b$ allows the protocol to terminate and records the output y of any honest party.
5. Finally, $\mathcal{Z}_b$ takes its time after the protocol has outputted to evaluate $\hat{b} = \hat{B}_\pi(\textsc{view}^{\hat{r}_\pi}_{\pi,\hat{\mathcal{P}}_\pi})$.[6]
6. The output of $\mathcal{Z}_b$ is then computed as follows:
   - If $b = 0$: output 0 (real) if $\hat{b} = b$ and output 1 (ideal) otherwise;
   - if $b = 1$: output 0 (real) if $y = 0$ and output 1 (ideal) otherwise.

Next, We show that any simulator that successfully simulates against $\mathcal{Z}_0$ will fail to simulate against $\mathcal{Z}_1$. To make the statement even stronger, we do not even restrict the simulator to be (R,T)-bounded. Indeed, let $\mathcal{S}$ be any simulator in the $\mathcal{F}_{\textsf{bc}}$-ideal experiment for the dummy adversary.

We consider the following events in the real execution of π with this environment $\mathcal{Z}$ and the dummy adversary$\mathcal{A}$ (recall that b stands for the input bit, $\hat{b}$ for the output of $\hat{B}_\pi$, and y for the common output bit):

- $\mathcal{E}^{\mathcal{Z},\mathcal{A},\pi}_{b=\hat{b}}$ occurs when in the real experiment $b = \hat{b}$.
- $\mathcal{E}^{\mathcal{Z},\mathcal{A},\pi}_{b=1}$ occurs when in the real experiment $b = 1$.
- $\mathcal{E}^{\mathcal{Z},\mathcal{A},\pi}_{b=0}$ occurs when in the real experiment $b = 0$.
- $\mathcal{E}^{\mathcal{Z},\mathcal{A},\pi}_{b=1\neq y}$ occurs when in the real experiment $b = 1$ and $y = 0$.

[6]Observe that the fact that the environment is (R,T)-bounded restricts how fast it can compute the output of $\hat{B}_\pi$ but since $\hat{B}_\pi$ is a polynomial-time algorithm, a polynomial-time environment will be able to compute it within its runtime.

By definition of the events, for the environments $\mathcal{Z}_0$ and $\mathcal{Z}_1$ we have:

$$\Pr\left[\mathcal{E}_{b=0}^{\mathcal{Z}_0,\mathcal{A},\pi}\right] = \Pr\left[\mathcal{E}_{b=1}^{\mathcal{Z}_1,\mathcal{A},\pi}\right] = 1. \tag{5}$$

Additionally, by correctness of the protocol π and the definition of the class $\Pi_{\mathsf{step\text{-}rel}}$, there exists a negligible function μ such that:

$$\Pr\left[\mathcal{E}_{b=\hat{b}}^{\mathcal{Z}_0,\mathcal{A},\pi}\right] = 1 - \mu(\kappa). \tag{6}$$

Since $\mathcal{Z}_0$ always uses input bit $b = 0$, it always uses the first condition to determine the output, i.e., output 0 (real) if and only if $b = \hat{b}$. Hence, for the output $\textsc{real}_{\mathcal{Z}_0,\mathcal{A},\pi}(\kappa)$ of the environment $\mathcal{Z}_0$ in the real experiment with (dummy) adversary $\mathcal{A}$, the output is 1 (ideal) with negligible probability:

$$\Pr\left[\textsc{real}_{\mathcal{Z}_0,\mathcal{A},\pi}(\kappa) = 1\right] = \Pr\left[\overline{\mathcal{E}_{b=\hat{b}}^{\mathcal{Z}_0,\mathcal{A},\pi}}\right] = \mu(\kappa). \tag{7}$$

Additionally, for the environment $\mathcal{Z}_1$, by definition of the class $\Pi_{\mathsf{step\text{-}rel}}$ when every party in $\hat{\mathcal{P}}_\pi$ crashes before sending their $\hat{r}_\pi$-round message, then the output of the honest parties is flipped with noticeable probability. That is, there exists a noticeable function q such that.

$$\Pr\left[\mathcal{E}_{b=1\neq y}^{\mathcal{Z}_1,\mathcal{A},\pi}\right] = q(\kappa). \tag{8}$$

Since $\mathcal{Z}_1$ always uses input bit $b = 1$, it always uses the second condition to determine the output, i.e., output 0 (real) if and only if $y = 0$. Hence:

$$\Pr\left[\textsc{real}_{\mathcal{Z}_1,\mathcal{A},\pi}(\kappa) = 0\right] = \Pr\left[\mathcal{E}_{b=1\neq y}^{\mathcal{Z}_1,\mathcal{A},\pi}\right] = q(\kappa). \tag{9}$$

Let us now turn to the ideal experiment. Consider the following events in the ideal execution of π with environment $\mathcal{Z}$ and the simulator $\mathcal{S}$:

- $\mathcal{E}_{b=\hat{b}}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ occurs when in the ideal experiment $b = \hat{b}$.
- $\mathcal{E}_{b=0}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ occurs when in the ideal experiment $b = 0$.
- $\mathcal{E}_{b=1}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ occurs when in the ideal experiment $b = 1$.
- $\mathcal{E}_{b=1\neq y}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ occurs when in the ideal experiment $b = 1$ and $y = 0$.
- $\mathcal{E}_{\mathsf{rush}}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ which occurs when the simulator asks $\mathcal{F}_{\mathsf{bc}}$ for the output while the sender is still honest and before sending the $\hat{r}_\pi$-round simulated messages to the environment.

As above, we have:

$$\Pr\left[\mathcal{E}_{b=0}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}\right] = \Pr\left[\mathcal{E}_{b=1}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}\right] = 1. \tag{10}$$

We next observe that the event $\mathcal{E}_{\mathsf{rush}}^{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathsf{bc}}}$ occurs with the same probability for $\mathcal{Z}_0$ and $\mathcal{Z}_1$. The reason is that this event is triggered by the simulator based on its view of the (ideal) execution before it gets any information from $\mathcal{F}_{\mathsf{bc}}$ and while $\mathcal{Z}_b$ (asks its adversary to) behave according to the protocol. Indeed, up to round $\hat{r}_\pi$, the environment $\mathcal{Z}_b$ behaves identically to $\mathcal{Z}_{\bar{b}}$ and independently of b. Thus, for some function $f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa)$ it holds that:

$$\Pr\left[\overline{\mathcal{E}_{\mathsf{rush}}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}}\right] = \Pr\left[\overline{\mathcal{E}_{\mathsf{rush}}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}}\right] = f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa). \tag{11}$$

We next observe that when the simulator asks for the value before corrupting the sender (and hence before learning any information about the input), by definition of $\mathcal{F}_{\mathsf{bc}}$ the output gets locked to this (input) value and cannot be flipped even if $\mathcal{S}$ later corrupts the sender. Hence:

$$\Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=1\neq y} \,\middle|\, \mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] = 0. \tag{12}$$

Therefore, for the output $\text{IDEAL}_{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa)$ of the environment $\mathcal{Z}_1$ in the ideal experiment with simulator $\mathcal{S}$ we have:

$$\begin{aligned}
&\Pr\left[\text{IDEAL}_{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 0\right] = \Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=1\neq y}\right]\\
&= \Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=1\neq y} \mid \mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] \cdot \Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] + \Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=1\neq y} \mid \overline{\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right] \cdot \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right]\\
&\overset{\text{Eq.12}}{=} \Pr\left[\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=1\neq y} \mid \overline{\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right] \cdot \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right]\\
&\overset{\text{Eq.11}}{\leq} f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa).
\end{aligned} \tag{13}$$

Moreover, when the simulator does not ask for the output value before corrupting the sender, the $\hat{r}_\pi$-round messages are independent of the actual input value b. Since b is chosen randomly and $\hat{b}$ is the outcome of algorithm $\hat{B}_\pi$ on a view independent of b, it holds that:

$$\Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=\hat{b}}} \mid \overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right] = \frac{1}{2}. \tag{14}$$

Thus,

$$\begin{aligned}
&\Pr\left[\text{IDEAL}_{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 1\right] = \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=\hat{b}}}\right]\\
&= \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=\hat{b}}} \mid \mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] \cdot \Pr\left[\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] + \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=\hat{b}}} \mid \overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right] \cdot \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}}\right]\\
&\overset{\text{Eqs.14,11}}{=} \Pr\left[\overline{\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{b=\hat{b}}} \mid \mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] \cdot \Pr\left[\mathcal{E}^{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}_{\mathsf{rush}}\right] + \frac{1}{2} \cdot f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa)\\
&\geq \frac{1}{2} \cdot f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa).
\end{aligned} \tag{15}$$

Putting things together, the distinguishing advantage of $\mathcal{Z}_0$ in distinguishing a real execution of π with dummy adversary $\mathcal{A}$ from and an $\mathcal{F}_{\mathsf{bc}}$-ideal execution with simulator $\mathcal{S}$ is:

$$\left|\Pr\left[\text{REAL}_{\mathcal{Z}_0,\mathcal{A},\pi}(\kappa) = 1\right] - \Pr\left[\text{IDEAL}_{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 1\right]\right|. \tag{16}$$

Since $\Pr[\text{REAL}_{\mathcal{Z}_0,\mathcal{A},\pi}(\kappa) = 1]$ is negligible (by Equation 7), the assumed security of π (which demands that the above distinguishing advantage be negligible) implies that $\Pr[\text{IDEAL}_{\mathcal{Z}_0,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 1]$ should also be negligible, which, by Equation 15, means that $f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa)$ is negligible.

However, in this case the distinguishing advantage of $\mathcal{Z}_1$, defined as

$$\left|\Pr\left[\text{REAL}_{\mathcal{Z}_1,\mathcal{A},\pi}(\kappa) = 0\right] - \Pr\left[\text{IDEAL}_{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 0\right]\right|, \tag{17}$$

will be noticeable since by Equations 9 and 13 it holds that $\Pr[\text{REAL}_{\mathcal{Z}_1,\mathcal{A},\pi}(\kappa) = 0]$ is noticeable and $\Pr[\text{IDEAL}_{\mathcal{Z}_1,\mathcal{S},\mathcal{F}_{\mathsf{bc}}}(\kappa) = 0]$ is at most $f_{S,\pi,\mathcal{F}_{\mathsf{bc}}}(\kappa)$, hence negligible. This contradicts the assumed security of π. □

5.2 Simulation-Based Adaptively Secure Broadcast Protocol

The main reason why the protocol from Section 4.2 does not realize the simulation-based definition is that once the simulator simulates an honest sender broadcasting the puzzle Z (without knowing the real input value), it cannot equivocate the content of the puzzle upon corruption of the sender, or when the protocol completes and the output is revealed. We now proceed to construct an adaptively secure broadcast protocol that satisfies the simulation-based definition in the random-oracle model. First off, we introduce the notion of time-lock puzzles that are *non-committing*.

Non-committing time-lock puzzles. Standard constructions of time-lock puzzles are committing in the sense that once a puzzle is generated, it can be opened into a unique message with all but negligible probability. In contrast, a *non-committing* time-lock puzzle enables a simulator to initially simulate a puzzle, and later, given an arbitrary message m, to "explain" the puzzle as containing m (and, in particular, ensure that the puzzle is opened for m). We show how to achieve this notion given a standard time-lock puzzle and a programmable random oracle, by generating $Z = \mathsf{PGen}(T, x)$ for a random $x \leftarrow \{0,1\}^\kappa$ and attaching $c = H(x) \oplus m$ to the puzzle. Once the simulator is asked to equivocate the new puzzle $(Z.c)$ to the message m, it can program the random oracle to return $H(x) = c \oplus m$.

We proceed to state the theorem.

Theorem 4. *Assume that weak time-lock puzzle with gap $\varepsilon < 1$ exist and that unfair broadcast can be computed in R rounds against an adaptive, PPT t-adversary, for $t \leq n$. Let $T(\cdot)$ be a polynomial. Then, Protocol $\pi_{\mathsf{bc\text{-}sim}}$ (Figure 5) is a broadcast protocol according to the simulation-based definition (Definition 7) that is secure against an adaptive t-adversary in the programmable random-oracle model, where the adversary and the environment are PPT and (R, T)-bounded.*

Protocol $\pi_{\mathsf{bc\text{-}sim}}(T, \kappa)$

- **Hybrid model:** The protocol is defined in the unfair broadcast $\mathcal{F}_{\mathsf{ubc}}$-hybrid model, requiring R rounds. The parties have access to a random oracle $H : \{0,1\}^\kappa \to \{0,1\}^\kappa$.
- **Public parameters:** A puzzle $(\mathsf{PGen}, \mathsf{PSol})$ with gap $\varepsilon < 1$, a difficulty parameter T, and the security parameter κ.
- **Private input:** The sender has a private input $m \in \{0,1\}^\kappa$.
- **The protocol:**
  - **Lock:** The sender samples a random $x \leftarrow \{0,1\}^\kappa$ and computes $Z \leftarrow \mathsf{PGen}(T^{1/\varepsilon}, x)$.
  - **Unfair broadcast:** The sender sets $c = m \oplus H(x)$ and broadcasts (Z, c) via $\mathcal{F}_{\mathsf{ubc}}$.
  - **Recover the output:** Upon receiving (Z, c), each party computes $x = \mathsf{PSol}(Z)$ and outputs $c \oplus H(x)$.

Figure 5: Adaptively secure, simulation-based broadcast protocol

Proof. Correctness of the protocol follows from the correctness of the time-lock puzzle and of $\mathcal{F}_{\mathsf{ubc}}$. Without loss of generality, we will prove security against the dummy adversary (that simply acts as a relay that delivers everything that it sees to the environment and sends whatever the environment instructs it to send[7]). We will construct a PPT simulator interacting with the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ and with ideal (dummy) parties $\tilde{P}_1, \ldots, \tilde{P}_n$, such that no (R, T)-bounded environment can distinguish between interacting with the real protocol and the dummy adversary or with the ideal computation and the simulator, except with negligible probability.

The simulator $\mathcal{S}$ initially constructs virtual parties $P_1, \ldots, P_n$, and emulates the unfair broadcast functionality $\mathcal{F}_{\mathsf{ubc}}$ and the random oracle H towards the environment $\mathcal{Z}$. To simulate the protocol, $\mathcal{S}$ proceeds as follows:

- The simulator stores a list L of the random oracle queries made by the environment. Whenever $\mathcal{Z}$ queries the random oracle with a value z, the simulator checks to see if a pair (z, w) (for some w) appears in L, and if so returns w to $\mathcal{Z}$; otherwise, $\mathcal{S}$ samples a random $w \leftarrow \{0,1\}^\kappa$, adds (z, w) to L and returns w to $\mathcal{Z}$.

[7]The dummy adversary is known to be the "worst-case adversary" in the sense that if the protocol is proven secure in the face of the dummy adversary then the protocol is secure in the face of every adversary [16].

- If $\mathcal{Z}$ requests to corrupt a party before the first round, $\mathcal{S}$ corrupts the dummy party in the ideal world. In case of corrupting the sender, $\mathcal{S}$ continues the rest of the simulation by honestly running all the remaining honest parties.
- In case the sender is honest in the first round, $\mathcal{S}$ samples random $c, x \leftarrow \{0,1\}^\kappa$ and computes $Z \leftarrow \mathsf{PGen}(T, x)$. Next, $\mathcal{S}$ sends (Z, c) to $\mathcal{Z}$ as the leakage from $\mathcal{F}_{\mathsf{ubc}}$.
- If $\mathcal{Z}$ requests to corrupt a party during the first R rounds of the protocol, $\mathcal{S}$ corrupts the dummy party in the ideal world. In case of corrupting the sender, $\mathcal{S}$ learns the input message m, and adds (x, w) to L for $w = m \oplus c$ (in case $(x, \cdot)$ already appears in L the simulator aborts).
- If after R rounds the sender is still honest, $\mathcal{S}$ requests the output from $\mathcal{F}_{\mathsf{bc}}$, receives back m, and adds (x, w) to L for $w = m \oplus c$ (in case $(x, \cdot)$ already appears in L the simulator aborts).
- If $\mathcal{Z}$ requests to corrupt a party after the first R rounds of the protocol, $\mathcal{S}$ corrupts the dummy party in the ideal world.

Let $\mathcal{Z}$ be an (R, T)-bounded environment. Note that $\mathcal{Z}$ can distinguish if and only if it succeeds in querying the random oracle on x during the first R rounds, since in this case the simulator must decide on $H(x)$ before knowing m. Denote by $q(\kappa)$ an upper bound on the number of queries made by $\mathcal{Z}$, then it holds that this event occurs with at most $q(\kappa)/2^\kappa + \mu(\kappa)$ probability, where $\mu(\kappa)$ is the negligible probability in which the environment $\mathcal{Z}$ can learn x from the puzzle Z. In case this event does not occur, the simulation is perfect and induces identically distributed views between the real and the ideal computations. □

By instantiating $\mathcal{F}_{\mathsf{ubc}}$ with the protocol of Dolev and Strong [27] we derive the following corollary.

Corollary 13. *Assume that weak time-lock puzzles with gap $\varepsilon < 1$ exist, let $t \leq n$, and let $T(\cdot)$ be a polynomial. Then, there exists a broadcast protocol (according to Definition 7) in the programmable random-oracle model and given a PKI for digital signatures, that is secure against an adaptive t-adversary, where the adversary and the environment are PPT and (R, T)-bounded.*

Bibliography

[1] I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. Communication complexity of Byzantine agreement, revisited. In P. Robinson and F. Ellen, editors, *38th ACM PODC*, pages 317–326. ACM, 2019.

[2] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Synchronous Byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In I. Goldberg and T. Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 320–334. Springer, Heidelberg, 2019.

[3] M. Andrychowicz and S. Dziembowski. PoW-based distributed cryptography with no trusted setup. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 379–399. Springer, Heidelberg, 2015.

[4] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, 2018.

[5] C. Badertscher, R. Canetti, J. Hesse, B. Tackmann, and V. Zikas. Universal composition with global subroutines: Capturing global setup within plain UC. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 1–30. Springer, Heidelberg, 2020.

[6] C. Baum, B. David, R. Dowsley, J. B. Nielsen, and S. Oechsner. TARDIS: Time and relative delays in simulation. Cryptology ePrint Archive, Report 2020/537, 2020. https://eprint.iacr.org/2020/537 (to appear in Eurocrypt 2021).

[7] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In M. Sudan, editor, *ITCS 2016*, pages 345–356. ACM, 2016.

[8] E. Blum, J. Katz, C.-D. Liu-Zhang, and J. Loss. Asynchronous Byzantine agreement with subquadratic communication. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 353–380. Springer, Heidelberg, 2020.

[9] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, 2000.

[10] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, 2018.

[11] M. Borderding. Levels of authentication in distributed agreement. In *10th International Workshop on Distributed Algorithms WDAG*, pages 40–55, 1996.

[12] E. Boyle, R. Cohen, D. Data, and P. Hubácek. Must the communication graph of MPC protocols be an expander? In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 243–272. Springer, Heidelberg, 2018.

[13] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437. Springer, Heidelberg, 2019.

[14] J. Camenisch, M. Drijvers, T. Gagliardoni, A. Lehmann, and G. Neven. The wonderful world of global random oracles. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, 2018.

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

[16] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, 2001.

[17] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, 2001.

[18] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, 1996.

[19] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, 2002.

[20] R. Canetti, A. Cohen, and Y. Lindell. A simpler variant of universally composable security for standard multiparty computation. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, 2015.

[21] T.-H. H. Chan, R. Pass, and E. Shi. Sublinear-round Byzantine agreement under corrupt majority. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 246–265. Springer, Heidelberg, 2020.

[22] J. Chen and S. Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

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

[24] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Probabilistic termination and composability of cryptographic protocols. In *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, Heidelberg, 2016.

[25] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. In *ICALP 2017*, volume 80 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl, 2017.

[26] R. Cohen, a. shelat, and D. Wichs. Adaptively secure MPC with sublinear communication complexity. In *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 30–60. Springer, Heidelberg, 2019.

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

[28] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 139–147. Springer, Heidelberg, 1993.

[29] L. Eckey, S. Faust, and J. Loss. Efficient algorithms for broadcast and consensus based on proofs of work. Cryptology ePrint Archive, Report 2017/915, 2017. http://eprint.iacr.org/2017/915.

[30] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass. Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Report 2020/779, 2020. https://eprint.iacr.org/2020/779.

[31] P. Feldman. *Optimal Algorithms for Byzantine Agreement*. PhD thesis, Stanford University, 1988. https://dspace.mit.edu/handle/1721.1/14368.

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

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

[34] M. Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2003. URL http://d-nb.info/967397375.

[35] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement in t+1 rounds. In *25th ACM STOC*, pages 31–41. ACM Press, 1993.

[36] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 404–428. Springer, Heidelberg, 2006.

[37] J. A. Garay, J. Katz, R. Kumaresan, and H.-S. Zhou. Adaptively secure broadcast, revisited. In C. Gavoille and P. Fraigniaud, editors, *30th ACM PODC*, pages 179–186. ACM, 2011.

[38] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, 2015.

[39] J. A. Garay, Y. Ishai, R. Ostrovsky, and V. Zikas. The price of low communication in secure multi-party computation. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 420–446. Springer, Heidelberg, 2017.

[40] J. A. Garay, A. Kiayias, R. M. Ostrovsky, G. Panagiotakos, and V. Zikas. Resource-restricted cryptography: Revisiting MPC bounds in the proof-of-work era. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 129–158. Springer, Heidelberg, 2020.

[41] S. Garg and A. Sahai. Adaptively secure multi-party computation with dishonest majority. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 105–123. Springer, Heidelberg, 2012.

[42] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. ISBN ISBN 0-521-83084-2 (hardback).

[43] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, 1987.

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

[45] S. Goldwasser, Y. T. Kalai, and S. Park. Adaptively secure coin-flipping, revisited. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *ICALP 2015, Part II*, volume 9135 of *LNCS*, pages 663–674. Springer, Heidelberg, 2015.

[46] I. Haitner and Y. Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. In *61st FOCS*, pages 1268–1276. IEEE Computer Society Press, 2020.

[47] C. Hazay, Y. Lindell, and A. Patra. Adaptively secure computation with partial erasures. In C. Georgiou and P. G. Spirakis, editors, *34th ACM PODC*, pages 291–300. ACM, 2015.

[48] M. Hirt and V. Zikas. Adaptively secure broadcast. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 466–485. Springer, Heidelberg, 2010.

[49] D. Hofheinz and J. Müller-Quade. A synchronous model for multi-party computation and the incompleteness of oblivious transfer. Cryptology ePrint Archive, Report 2004/016, 2004. http://eprint.iacr.org/2004/016.

[50] Y. T. Kalai, I. Komargodski, and R. Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *DISC*, pages 34:1–34:16, 2018.

[51] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, Heidelberg, 2013.

[52] J. Katz, A. Thiruvengadam, and H.-S. Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 14–31. Springer, Heidelberg, 2013.

[53] J. Katz, J. Loss, and J. Xu. On the security of time-lock puzzles and timed commitments. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 390–413. Springer, Heidelberg, 2020.

[54] H. A. Khorasgani, H. K. Maji, and T. Mukherjee. Estimating gaps in martingales and applications to coin-tossing: Constructions and hardness. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 333–355. Springer, Heidelberg, 2019.

[55] A. Kiayias, H.-S. Zhou, and V. Zikas. Fair and robust multi-party computation using a global transaction ledger. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Heidelberg, 2016.

[56] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[57] H. Lin, R. Pass, and P. Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In C. Umans, editor, *58th FOCS*, pages 576–587. IEEE Computer Society Press, 2017.

[58] C.-D. Liu-Zhang and U. Maurer. Synchronous constructive cryptography. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 439–472, 2020.

[59] M. Mahmoody, T. Moran, and S. P. Vadhan. Time-lock puzzles in the random oracle model. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 39–50. Springer, Heidelberg, 2011.

[60] G. Malavolta and S. A. K. Thyagarajan. Homomorphic time-lock puzzles and applications. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, 2019.

[61] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, 2002.

[62] J. B. Nielsen. *On Protocol Security in the Cryptographic Model.* PhD thesis, University of Aarhus, 2003. https://www.brics.dk/DS/03/8/BRICS-DS-03-8.pdf.

[63] R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, 2017.

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

[65] B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *STACS*, volume 577 of *LNCS*, pages 339–350. Springer, 1992.

[66] K. Pietrzak. Simple verifiable delay functions. In A. Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, 2019.

[67] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.

[68] L. Rotem and G. Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 481–509. Springer, Heidelberg, 2020.

[69] J. Wan, H. Xiao, S. Devadas, and E. Shi. Round-efficient Byzantine broadcast under strongly adaptive and majority corruptions. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 412–456. Springer, Heidelberg, 2020.

[70] J. Wan, H. Xiao, E. Shi, and S. Devadas. Expected constant round Byzantine broadcast under dishonest majority. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411. Springer, Heidelberg, 2020.

[71] B. Wesolowski. Efficient verifiable delay functions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, 2019.

[72] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, 1982.