# Round-Efficient Secure Computation in Point-to-Point Networks*

Jonathan Katz** and Chiu-Yuen Koo

Dept. of Computer Science, University of Maryland, College Park, USA
{jkatz,cykoo}@cs.umd.edu

**Abstract.** Essentially all work studying the round complexity of secure computation assume broadcast as an atomic primitive. Protocols constructed under this assumption tend to have very poor round complexity when compiled for a point-to-point network due to the high overhead of emulating each invocation of broadcast. This problem is compounded when broadcast is used in more than one round of the original protocol due to the complexity of handling sequential composition (when using round-efficient emulation of broadcast).

We argue that if the goal is to optimize round complexity in point-to-point networks, then it is preferable to design protocols — assuming a broadcast channel — minimizing the *number of rounds in which broadcast is used* rather than minimizing the *total number of rounds*. With this in mind, we present protocols for secure computation in a number of settings that use only a *single* round of broadcast. In all cases, we achieve optimal security threshold for adaptive adversaries, and obtain protocols whose round complexity (in a point-to-point network) improves on prior work.

## 1 Introduction

The round complexity of cryptographic protocols — and, in particular, protocols for secure multi-party computation of general functionalities — has been the subject of intense study. Establishing bounds on the round complexity of various tasks is, of course, of fundamental theoretical importance. Moreover, reducing the round complexity of existing protocols is crucial if we ever hope to use these protocols in the real world. If the best known protocol for a given task requires hundreds of rounds, it will never be used; on the other hand, if we know (in principle) that round-efficient solutions are possible, we can then turn our attention to improving other aspects (such as computation) in an effort to obtain a protocol that can be used in practice.

Previous research investigating the round complexity of protocols for secure multi-party computation (MPC) has almost exclusively focused on optimizing the round complexity *under the assumption that a broadcast channel is available.* (We survey some of this work in Section 1.2.) In most settings where MPC

* Work done in part while the authors were visiting IPAM.

** This research was supported by NSF CAREER award #0447075 and US-Israel Binational Science Foundation grant #2004240.

M. Naor (Ed.): EUROCRYPT 2007, LNCS 4515, pp. 311–328, 2007.

might potentially be used, however, only point-to-point channels are likely to be available and a broadcast channel is not expected to exist. Nevertheless, the use of a broadcast channel is justified in previous work by the fact that the broadcast channel can always be *emulated* by having the parties run a broadcast protocol over the point-to-point network.

We argue that if the ultimate goal is to optimize round complexity for point-to-point networks (i.e., where the protocol will actually be run), then the above may be a poor approach due to the high overhead introduced by the final step of emulating the broadcast channel. Specifically:

- If the broadcast channel is emulated using a deterministic protocol [16,11], then a lower bound due to Fischer and Lynch [13] shows that $\Omega(t+R)$ rounds are needed to emulate $R$ rounds of broadcast in the original protocol (this is true regardless of how many parties broadcast during the same round). Here and in the rest of the paper, $t$ denotes the number of malicious parties and may be linear in the total number of parties $n$. In particular, this will not lead to sub-linear-round protocols with optimal security threshold $t = \Theta(n)$.
- Using randomized protocols, each round of broadcast in the original protocol can be emulated in an expected constant number of rounds [12,14,22]. Nevertheless, the exact constant is rather high. More problematic is that if broadcast is used in *more than one* round of the original protocol, then it is necessary to explicitly handle sequential composition of protocols without simultaneous termination [6,24,22]. (This is not an issue if broadcast is used in only a *single* round.) This leads to a substantial increase in round complexity; we refer the reader to Appendix A for details.

To illustrate the point, consider the protocols of Micali and Rabin [25] and Fitzi, et al. [15] (building on [17]) for verifiable secret sharing (VSS) with $t < n/3$. The Micali-Rabin protocol uses 16 rounds but only a single round of broadcast; the protocol of Fitzi et al. uses three rounds, two of which involve broadcast. Compiling these protocols for a point-to-point network using the most round-efficient randomized broadcast protocol known, the Micali-Rabin protocol runs in an expected 31 rounds while the protocol by Fitzi et al. requires an expected 55 rounds! The conclusion is that optimizing round complexity using broadcast does not, in general, lead to round-optimal protocols in the point-to-point model.

This suggests that if the ultimate goal is a protocol for a point-to-point network, then it is preferable to focus on minimizing the number of rounds *in which broadcast is used* rather than on minimizing the total number of rounds. This raises in particular the following question:

*Is it possible to construct* **constant-round** *(or even sub-linear-round) protocols for secure computation that use only a* **single** *round of broadcast*?

Note that for $t = \Theta(n)$ at least one round of broadcast is necessary if the protocol uses a strict constant number of rounds, since broadcast itself cannot be achieved over point-to-point channels in a strict constant number of rounds.

We resolve the above question in the affirmative in a number of settings. Specifically, we show:

1. A constant-round protocol using a single round of broadcast that is secure for $t < n/3$ and assumes only the existence of one-way functions.
2. A constant-round protocol using a single round of broadcast that is secure for $t < n/2$ and assumes only a public-key infrastructure (PKI) along with secure signatures.
3. A protocol using a single round of broadcast and achieving information-theoretic security for $t < n/3$. Here, the round complexity is linear in the depth of the circuit being computed.

All protocols are secure even for adaptive adversaries.

Of course, the fact that a protocol uses broadcast in only a single round does not necessarily imply that it yields the most round-efficient protocol in a point-to-point setting. For the protocols we construct, however, this is indeed the case (at least given the most round-efficient known techniques for emulating broadcast over point-to-point channels). For example, the first protocol mentioned above requires 41 rounds (in expectation) when compiled for a point-to-point network. In contrast, *any* protocol for $t < n/3$ that uses broadcast in *two* rounds (even if that is all it does!) will require at least 55 rounds (in expectation) when run in a point-to-point network (see Appendix A). Similarly, any protocol for $t < n/2$ that uses broadcast in two rounds will require at least 96 rounds (in expectation) in a point-to-point network. We stress again that the main issue in moving from one broadcast to two (or more) broadcasts is the significant overhead in the latter case needed to deal with sequential composition of protocols that do not terminate in the same round.

### 1.1 Overview of Our Techniques

We give a high-level overview of the main techniques we use in constructing the protocols outlined above. Call $(a, b, c)$, where $a, b$, and $c$ are elements of some field, a *random multiplication triple* if $a$ and $b$ are uniformly distributed, each of $a, b, c$ is shared among the players,[1] and $c = ab$. Beaver [3] shows that if, in a "setup phase," the parties share their inputs along with sufficiently-many multiplication triples — in particular, one multiplication triple for each multiplication gate of the circuit being evaluated — then the parties can evaluate the circuit in a round-efficient manner *without using any further invocations of broadcast*. Our task is thus reduced to showing how to perform the necessary setup using only a single round of broadcast.

To achieve this, we use the concept of *moderated protocols* as introduced in [22]. In such protocols, there is a distinguished party $P_m$ known as the *moderator*. Given a protocol $\Pi$, designed under the assumption of a broadcast channel, the moderated version of $\Pi$ is a protocol $\Pi'$ that runs in a point-to-point network and has the following properties (roughly speaking):

- At the end of $\Pi'$, each party $P_i$ outputs a binary value $\mathsf{trust}_i(m)$.
- If the moderator $P_m$ is honest, then each honest $P_i$ outputs $\mathsf{trust}_i(m) = 1$. This represents the fact that an honest party $P_i$ "trusts" the moderator $P_m$.

[1] For now, we do not specify the exact manner in which sharing is done.

- If any honest party $P_i$ outputs $\mathsf{trust}_i(m) = 1$, then $\Pi'$ achieves the functionality of $\Pi$.

In our prior work [22], we have shown[2] how to compile any protocol $\Pi$ into its moderated version $\Pi'$, while increasing the round complexity of $\Pi$ by at most a constant multiplicative factor (the exact effect on the round complexity depends on the number of invocations of broadcast in $\Pi$). For $t < n/3$, the compilation does not require any assumptions; for $n/3 \leq t < n/2$, the compilation assumes a PKI and digital signatures.

Let $\Pi_i$ denote some protocol, designed assuming a broadcast channel, that shares the input value of party $P_i$ as well as sufficiently-many multiplication triples. Such protocols are constructed in, e.g., [7,29,1,19,9,10]. We compile $\Pi_i$ into a moderated protocol $\Pi_i'$ where $P_i$ itself acts as the moderator. Now consider the following protocol that uses broadcast in only a single round:

1. Run protocols $\{\Pi_i'\}_{i=1}^n$ in parallel.[3] Recall that $P_i$ is the moderator in $\Pi_i'$.
2. Each party $P_i$ broadcasts $\{\mathsf{trust}_i(1), \ldots, \mathsf{trust}_i(n)\}$.
3. A party $P_i$ is disqualified if $|\{j : \mathsf{trust}_j(i) = 1\}| \leq t$; i.e., if $t$ or fewer players broadcast $\mathsf{trust}_j(i) = 1$. If $P_i$ is disqualified, then a default value is used as the input for $P_i$.
4. Let $i^*$ be the minimum value such that $P_{i^*}$ is not disqualified. The set of random multiplication triples that the parties will use is taken to be the set that was generated in $\Pi_{i^*}'$.

Analyzing the above, note that if $P_i$ is honest and there exists an honest majority, then at least $t + 1$ parties broadcast $\mathsf{trust}_j(i) = 1$. Hence an honest $P_i$ is never disqualified. On the other hand, at least one of the parties that broadcast $\mathsf{trust}_j(i^*) = 1$ must be honest. The properties of moderated protocols discussed earlier thus imply that $\Pi_{i^*}'$ achieves the functionality of $\Pi_{i^*}$. Since $\Pi_{i^*}$ is assumed to securely share sufficiently-many multiplication triples, it follows that the above protocol securely shares sufficiently-many multiplication triples. A similar argument shows that the inputs of all non-disqualified parties are shared appropriately. We conclude that the above protocol implements the necessary setup phase using only one round of broadcast.

In a naive compilation of $\Pi_i$ to $\Pi_i'$ (following [22]), each round of broadcast in $\Pi_i$ is replaced by six rounds in $\Pi_i'$ (for the case $t < n/3$). Proceeding directly thus yields secure MPC protocols with relatively high round complexity: after all, existing constructions of protocols $\Pi_i$ achieving the needed functionality do not attempt to minimize the number of rounds of broadcast. We present instead a new set of protocols that minimize their use of broadcast. Furthermore, our implementation of the setup phase deviates from the above simplified approach in order to further optimize the round complexity of the final protocol. Along the way, we construct round-efficient protocols for VSS that use broadcast only

[2] Although our prior work only claims the result when $\Pi$ is a VSS protocol, it is not hard to verify that the proof extends for more general classes of functionalities.

[3] In fact, only protocols $\Pi_1', \ldots, \Pi_{t+1}'$ need to share multiplication triples; the remaining protocols only need to share the input of the appropriate player.

once; these in turn yield the most round-efficient VSS and broadcast protocols for point-to-point networks. For $t < n/3$ we show a 7-round VSS protocol using broadcast once (the best previous VSS protocol using broadcast once, obtained by combining [15,22], requires 14 rounds), and for $t < n/2$ we obtain a 5-round VSS protocol using broadcast once (the best previous protocol required 34 rounds [22]). The latter implies an expected 36-round broadcast protocol for the same threshold (improved from 58 rounds in [22]).

### 1.2 Prior Work

There is a vast amount of work in the cryptographic and distributed computing literature studying the round complexity of various tasks; here, we summarize the work most relevant to our own.

**Broadcast/Byzantine agreement.** For $t < n/2$, broadcast and Byzantine agreement (BA) have essentially the same round complexity (to within one round); therefore, we freely interchange between the two. In a synchronous network with pairwise authenticated channels and no additional setup, BA is achievable iff $t < n/3$ [26,23]. In this setting, a lower bound of $t + 1$ rounds for any deterministic protocol is known [13]. A protocol with this round complexity (but exponential message complexity) was shown by Pease, et al. [26,23]. Following a long sequence of works, Garay and Moses [16] show a fully-polynomial BA protocol with optimal resilience and round complexity.

To obtain protocols with sub-linear round complexity, researchers explored the idea of using randomization [28,5]. This culminated in the work of Feldman and Micali [12], who show a randomized BA protocol with optimal resilience running in an expected *constant* number of rounds.

To achieve resilience $t \geq n/3$, additional assumptions are needed; the most common assumptions are digital signatures and a PKI. Under these assumptions, linear-round deterministic broadcast protocols are known for $t < n$ [26,23,11]. For $t < n/2$, randomized protocols with expected constant-round complexity exist [14,22], the latter without any additional computational assumptions.

**VSS.** Gennaro, et al. [17] show a 2-round VSS protocol for $t < n/4$ and a 4-round protocol for $t < n/3$. They also give a 3-round protocol for $t < n/3$ with exponential complexity. Fitzi, et al. [15] determine the exact round complexity of VSS by showing a fully-polynomial 3-round VSS protocol for $t < n/3$. Their work also shows how to run many sequential VSS protocols at an amortized cost of only $(1 + \epsilon)$ rounds.

We stress that the above consider the round complexity of VSS *under the assumption that a broadcast channel is available*. (In particular, the VSS protocol from [15] is only optimal in this setting.) While of theoretical interest, this appears to be a poor approach (as explained in the Introduction) if one is ultimately interested in round-efficient protocols for point-to-point networks.

**General secure MPC.** Unconditionally-secure MPC protocols in point-to-point networks exist for $t < n/3$ (combining [7,8] with [26]), or for $t < n/2$ assuming a broadcast channel is available [2,30]. The broadcast channel can

be removed for $t < n/2$ by relying on a PKI and digital signatures [11] or information-theoretic pseudo-signatures [27].

Beaver, Micali, and Rogaway [4] gave a constant-round (computationally-secure) protocol for secure MPC with $t < n/2$, assuming a broadcast channel and one-way functions. Damgård and Ishai [10] showed a constant-round protocol under the same assumptions that is secure even for adaptive adversaries. These can both be converted to *expected* constant-round protocols in point-to-point networks by using the broadcast protocols mentioned above [12,22]. We stress that the constant is rather high, on the order of hundreds of rounds.

The work of Gennaro et al. [17] mentioned earlier implies a 3-round MPC protocol with resilience $t < n/4$, assuming the existence of one-way functions. The resulting protocol uses broadcast in only a single round, and so yields a very round-efficient protocol in point-to-point networks; the drawback is that the resilience is not optimal. In subsequent work [18], the same authors show that 2-round MPC is not possible (in general) for $t \geq 2$. However, they show that certain functionalities can be securely computed in 2 rounds for $t < n/6$.

Hirt, Nielsen, and Przydatek [21] show a protocol for asynchronous secure MPC that uses only one round of broadcast. Their result is not directly comparable to ours due to differences in the way rounds are counted in the synchronous and asynchronous settings. (In particular, their protocol requires a linear number of rounds when directly adapted to the synchronous setting.) They assume $t < n/3$ and a global setup assumption (stronger than a PKI).

Goldwasser and Lindell [20] show various round-efficient MPC protocols for point-to-point networks; however, the point of their work is to consider weaker security definitions in which fairness and output delivery are not guaranteed (even when an honest majority exists).

### 1.3 Outline of the Paper

We review and formalize some standard notions in Section 2. In Section 3 we focus on the case $t < n/3$ in both the computational and information-theoretic settings. Section 4 discusses the case of $t < n/2$ (with computational security). Due to lack of space, we defer some of the details to the full version.

## 2 Model and Preliminaries

We use the standard synchronous communication model where parties communicate using pairwise private/authenticated channels. In addition, we assume a broadcast channel with the understanding that it will be emulated using a round-efficient broadcast sub-routine. As a convenient shorthand, we say that a protocol has round complexity $(r, r')$ if it uses $r$ rounds in total and $r' \leq r$ of these rounds invoke broadcast (possibly by all parties). We emphasize that since our aim is to minimize the eventual round complexity in point-to-point networks, we will construct protocols that access the broadcast channel in only a single round (i.e., $(\star, 1)$-round protocols).

When we say a protocol tolerates $t$ malicious parties, we always mean that it is secure against a *rushing* adversary who may *adaptively* corrupt up to $t$ parties during execution of the protocol and coordinate the actions of these parties as they deviate from the protocol in an arbitrary manner. Parties not corrupted by the adversary are called *honest*. In our protocol descriptions, we implicitly assume that parties send a properly-formatted message at all times; this is without loss of generality, as an improper or missing message can always be interpreted as some default message.

For $t < n/3$ we do not assume any setup, but for $t < n/2$ we assume a PKI. Note that, since we are assuming a broadcast channel, the additional assumption of a PKI may not be necessary; nevertheless, we see no harm in assuming it since a PKI will be needed anyway once we compile our protocols to run in a point-to-point network. We leave open the question of constructing an $(O(1), 1)$-round secure MPC protocol for $t < n/2$ that uses a broadcast channel but no PKI.

### 2.1 Gradecast

Gradecast was introduced by Feldman and Micali [12].

**Definition 1.** *(Gradecast): A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished* dealer *$P^* \in \mathcal{P}$ holds initial input $M$, is a* gradecast protocol tolerating $t$ malicious parties *if the following conditions hold for any adversary controlling at most $t$ parties:*

- *Each honest party $P_i$ outputs a* message *$m_i$ and a* grade *$g_i \in \{0, 1, 2\}$.*
- *If the dealer is honest, then the output of every honest party $P_i$ satisfies $m_i = M$ and $g_i = 2$.*
- *If there exists an honest party $P_i$ who outputs message $m_i$ and grade $g_i = 2$, then the output of every honest party $P_j$ satisfies $m_j = m_i$ and $g_j \geq 1$.*

**Lemma 1 ([12,22]).** *There exists a $(3, 0)$-round gradecast protocol tolerating $t < n/3$ malicious parties and, assuming a PKI, a $(4, 0)$-round gradecast protocol tolerating $t < n/2$ malicious parties.*

### 2.2 Generalized Secret Sharing and VSS

Throughout, we assume a finite field $\mathbb{F}$ whose order is a power of 2 and which contains $[n]$ as a subset.

**Definition 2.** *(1-level sharing): We say a value $s \in \mathbb{F}$ has been* 1-level shared *if there exists a degree-$t$ polynomial $F_s(x)$ such that (1) $F_s(0) = s$ and (2) player $P_i$ holds the share $s_i \stackrel{\text{def}}{=} F_s(i)$. In this case, we say that $F_s(x)$* shares *$s$.*

When $t < n/3$ the parties can reconstruct $s$ by having all parties send their shares to all other parties, and then having each party use Reed-Solomon decoding to recover $s$. If $s, s'$ are 1-level shared then for any publicly-known $\alpha, \beta \in \mathbb{F}$ the value $\alpha s + \beta s'$ has been 1-level shared as well. We recall the following technical lemma concerning multiplication of shares [19]:

**Lemma 2.** *Let $A, B$ be degree-$t$ polynomials over $\mathbb{F}$, and $\alpha_1, \ldots, \alpha_{2t+1} \in \mathbb{F}$ distinct elements. Then $A(0) \cdot B(0) = \sum_{i=1}^{2t+1} \beta_i \cdot A(\alpha_i) \cdot B(\alpha_i)$ for some constants $\beta_1, \ldots, \beta_{2t+1} \in \mathbb{F}$.*

**Definition 3.** *(2-level sharing): We say a value $s \in \mathbb{F}$ has been* 2-level shared *if (1) there exists a degree-$t$ polynomial $F_s(x)$ that shares $s$ and (2) for $i \in [n]$, there exists a degree-$t$ polynomial $F_{s_i}(x)$, known to $P_i$, that shares $s_i \stackrel{\text{def}}{=} F_s(i)$ (i.e., each party $P_j$ holds a share $s_{j,i}$ of $s_i$).*

**Definition 4.** *(3-level sharing): We say a value $s \in \mathbb{F}$ has been* 3-level shared *if (1) there exists a degree-$t$ polynomial $F_s(x)$ that shares $s$; (2) for $i \in [n]$, the value $s_i \stackrel{\text{def}}{=} F_s(i)$ has been 2-level shared; and (3) each party $P_i$ knows the polynomial $F_{s_i}(x)$ that shares $s_i$.*

Note that if $s$ is 3-level (resp., 2-level) shared, then it is 2-level (resp., 1-level) shared as well.

**Definition 5.** *(VSS with 2-level (resp., 3-level) sharing): A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished* dealer *$P^* \in \mathcal{P}$ holds an initial input $s$, is a* VSS protocol with 2-level (resp., 3-level) sharing tolerating $t$ malicious parties *if the following conditions hold for any adversary controlling at most $t$ parties by the end of the protocol:*

**Secrecy:** *If the dealer is honest, then the joint view of the malicious parties is independent of the dealer's input $s$.*

**Commitment:** *At the end of the protocol, some value $s'$ is 2-level (resp., 3-level) shared. Moreover, if the dealer is honest then $s' = s$. On the other hand, if the dealer is dishonest, then $s'$ can be efficiently computed from the messages sent from the malicious parties to the honest parties during the protocol execution. We refer to this latter property as* **extraction.**

## 3 Secure Multiparty Computation for $t < n/3$

### 3.1 Outline of the Construction

We first construct a $(7, 1)$-round VSS protocol with 2-level sharing; this protocol will be used by parties to share their inputs. This VSS protocol is based on the $(4, 3)$-round VSS protocol due to Gennaro et al. [17].

Based on the above VSS protocol with 2-level sharing, we can construct an $(8, 1)$-round VSS protocol with *3-level* sharing. We sketch the protocol below:

1. The dealer shares the secret $s$ using the VSS protocol with 2-level sharing. In parallel, the dealer shares $g_1 \stackrel{\text{def}}{=} F_s(1), \ldots, g_n \stackrel{\text{def}}{=} F_s(n)$ using $n$ invocations of the 2-level VSS protocol.
2. The parties reconstruct $g_1 - F_s(1), \ldots, g_n - F_s(n)$ and check if all the values are equal to 0. If this condition does not hold, the dealer is disqualified.

Using the above as a building block, we construct a $(17, 3)$-round protocol for sharing a random multiple triple $(a, b, c = ab)$. On a high level, our protocol consists of the following steps:

1. Each party $P_i$ shares two random values $a^{(i)}$ and $b^{(i)}$ using VSS with 3-level sharing.
2. Set $a = \sum a^{(i)}$ and $b = \sum b^{(i)}$. Note that $a$ and $b$ have been 3-level shared. Let $F_a(x)$ and $F_b(x)$ be the polynomials sharing $a$ and $b$ respectively. Using the sharing of product of shares protocol from [7], each party $P_i$ shares $F_a(i) \cdot F_b(i)$ (using VSS with 2-level sharing) and proves that the right value is being shared; all parties can identify the set of parties that are not sharing the correct value.
3. Since $t < n/3$, there exist $2t + 1$ parties $P_i$ that correctly share $F_a(i) \cdot F_b(i)$ in step 2. Following Lemma 2, each party can compute its share of $c$ non-interactively.

The above protocol runs VSS twice sequentially. Using the amortization technique from [15], the round complexity can be reduced to $(11, 3)$. The idea is as follows: Suppose a party $P_i$ needs to share two values $a$ and $b$ using VSS in two consecutive steps. $P_i$ can do the following instead:

1. $P_i$ picks a random value $r$ and shares $a$ and $r$ using VSS.
2. $P_i$ broadcasts the value $b - r$. Since the value $r$ has been shared and $b - r$ has been made public, each party can compute its share of $b$ non-interactively.

By running the above protocols in parallel, we obtain a $(11, 3)$-round protocol $\Pi_i$ that allows party $P_i$ to both share its input and generate sufficiently-many random multiplication triples (cf. the overview in Section 1.1). In Section 3.3, we show how to use the ideas described in Section 1.1 (in particular, the idea of using moderated protocols) to implement the needed setup for all parties via a (21,1)-round protocol. (Our implementation of this protocol does not exactly follow the description in Section 1.1 for the reason described there). Based on this setup, we then show MPC protocols using only one round of broadcast in both the information-theoretic and computational settings (based on [7] and [10], respectively).

### 3.2 A $(7, 1)$-Round VSS Protocol with 2-Level Sharing

Our protocol is based on the $(4, 3)$-round VSS protocol of Gennaro et al. [17]. For readers who are already familiar with their protocol, we describe the two main modifications we make:

- Instead of using a "random pad" technique to detect inconsistent shares and resolve the inconsistencies in the next round — which requires two rounds of broadcast — we use a different method that requires only one round of broadcast. This gives us a $(6, 2)$-round protocol.
- After the above, two rounds of broadcast still remain in the protocol of [17]. We devise a way for parties to postpone the first broadcast (and then combine it with the second) without affecting the progress of the protocol. This gives the $(7, 1)$-round protocol as claimed.

We start by describing a $(6, 2)$-round protocol. When we say the dealer $P^*$ is *disqualified* we mean that execution of the protocol halts, and a default value $s'$ is 2-level shared (using some default polynomials).

**Round 1.** The dealer chooses a random bivariate polynomial $F \in \mathbb{F}[x, y]$ of degree $t$ in each variable with $F(0, 0) = s$. The dealer sends to $P_i$ the polynomials $g_i(x) \stackrel{\text{def}}{=} F(x, i)$ and $h_i(y) \stackrel{\text{def}}{=} F(i, y)$.

**Round 2.** $P_i$ sends $h_i(j)$ to $P_j$.

**Round 3.** Let $h'_{j,i}$ be the value $P_i$ received from $P_j$. If $h'_{j,i} \neq g_i(j)$, then $P_i$ sends "$\mathsf{complain}_i(j)$" to the dealer.

**Round 4.** If the dealer receives "$\mathsf{complain}_i(j)$" from $P_i$ in the last round, then the dealer sends "$\mathsf{complain}_i(j)$" to $P_j$.

**Round 5.** For every ordered pair $(i, j)$, parties $P_i, P_j$, and the dealer do the following:

- If $P_i$ sent "$\mathsf{complain}_i(j)$" to the dealer in round 3, then $P_i$ broadcasts "$(P_i, i, j) : g_i(j)$" else $P_i$ broadcasts "$(P_i, i, j)$: no complaint".
- If $P_j$ received "$\mathsf{complain}_i(j)$" from the dealer in round 4, then $P_j$ broadcasts "$(P_j, i, j) : h_j(i)$" else $P_j$ broadcasts "$(P_j, i, j)$: no complaint".
- If the dealer received "$\mathsf{complain}_i(j)$" from $P_i$ in round 3, then the dealer broadcasts "$(P^*, i, j) : F(j, i)$" else the dealer broadcasts "$(P^*, i, j)$: no complaint".

We say party $P_i$ is *unhappy* if $P_i$ broadcasted a message of the form "$(P_i, i, j) : Y$," the dealer broadcasted a message of the form "$(P^*, i, j) : X$," and[4] $X \neq Y$. Similarly, $P_i$ is unhappy if $P_i$ broadcasted a message of the form "$(P_i, j, i) : Y$," the dealer broadcasted a message of the form "$(P^*, j, i) : X$," and $X \neq Y$.

**Round 6.** For each unhappy party $P_j$, the dealer broadcasts the polynomials $g_j(x)$ and $h_j(y)$, and each party $P_i$ who is not unhappy broadcasts $b'_{i,j} \stackrel{\text{def}}{=} h_i(j)$ and $c'_{i,j} \stackrel{\text{def}}{=} g_i(j)$.

A party $P_i$ that is not unhappy becomes *accusatory* if, in round 6, for some unhappy party $P_j$, the dealer broadcasts polynomial $g_j(x)$ and $h_j(y)$ but $b'_{i,j} \neq g_j(i)$ and $c'_{i,j} \neq h_j(i)$.

A party that is neither unhappy nor accusatory is said to be *happy*. The dealer is disqualified if the number of happy parties is less than $n - t$.

**Output determination.** If the dealer has not been disqualified, then a happy party $P_i$ keeps the polynomials $g_i(x)$ and $h_i(y)$ it received from the dealer in the first round. An unhappy party $P_i$ takes the polynomials broadcasted by the dealer in the final round as $g_i(x)$ and $h_i(y)$. (We do not define what accusatory players do, since if the dealer is not disqualified then all such parties are malicious.) The share $P_i$ holds with respect to $s$ is $s_i \stackrel{\text{def}}{=} g_i(0)$, and the share $P_i$ holds with respect to $s_i$ is $s_{i,j} \stackrel{\text{def}}{=} h_i(j)$.

We briefly argue that the requirements of Definition 5 hold. Consider an honest dealer $P^*$. For any pair of honest parties $(P_i, P_j)$, the parties $P^*, P_i$, and $P_j$

[4] Note that $X$ or $Y$ can be field elements or the string "no complaint."

will always broadcast "no complaint" with respect to the ordered pair $(i, j)$ in round 5. Hence secrecy will not be violated. It is easy to see that an honest party $P_i$ will never become unhappy or accusatory. Therefore $P^*$ will not be disqualified as dealer.

Next consider a malicious dealer $P^*$. Suppose two honest parties $P_i$ and $P_j$ are holding inconsistent shares (i.e., $h'_{j,i} \neq g_i(j)$ or $h'_{i,j} \neq g_j(i)$). In round 5, they will broadcast different messages with respect to the ordered pair $(i, j)$ (or the ordered pair $(j, i)$). Hence the inconsistency is made known to all parties. The commitment property then follows the argument in [17].

Call the above protocol $\Pi_{(6,2)}$. We now show how to transform $\Pi_{(6,2)}$ into a $(7, 1)$-round protocol $\Pi_{(7,1)}$. The first four rounds of $\Pi_{(7,1)}$ are the same as $\Pi_{(6,2)}$. Then the parties carry out the following instructions:

**Round 5.** If $P_i$ is instructed to broadcast a message $m$ in round 5 of $\Pi_{(6,2)}$, then $P_i$ sends the message $m$ to all parties (via point-to-point links).

**Round 6.** Parties forward all the messages received in last round to all parties.

**Round 7.** The dealer does the following:

- For every ordered pair $(i, j)$, if in round 6 the dealer received messages of the form "$(P_i, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, each from $t + 1$ different parties, then the dealer broadcasts the polynomials $g_i(x)$ and $h_i(y)$.
- Similarly, for every ordered pair $(i, j)$, if in round 6 the dealer received messages of the form "$(P_j, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, each from $t + 1$ different parties, then the dealer broadcasts the polynomials $g_j(x)$ and $h_j(y)$.

In parallel with the above, all parties $P_k$ do the following:

- A party broadcasts all the messages it received in round 5 (note that the round-5 messages include the identity of the sender).
- For every ordered pair $(i, j)$, if in round 6 party $P_k$ received messages of the form "$(P_i, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, each from $t+1$ different parties, then $P_k$ broadcasts $b'_{k,i} \stackrel{\text{def}}{=} h_k(i)$ and $c'_{k,i} \stackrel{\text{def}}{=} g_k(i)$.
- For every ordered pair $(i, j)$, if in round 6 party $P_k$ received messages of the form "$(P_j, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, each from $t+1$ different parties, then $P_k$ broadcasts $b'_{k,j} \stackrel{\text{def}}{=} h_k(j)$ and $c'_{k,j} \stackrel{\text{def}}{=} g_k(j)$.

**Output determination.** Parties decide on their output as follows:

1. A party $P_i$ is said to *announce* a message $m$ if, in round 7, at least $n - t$ parties broadcast that they received $m$ from $P_i$ in round 5.
2. A party $P_i$ is *unhappy* if $P_i$ announced a message of the form "$(P_i, i, j) : Y$," the dealer announced a message of the form "$(P^*, i, j) : X$," and $X \neq Y$. Similarly, $P_i$ is unhappy if $P_i$ announced a message of the form "$(P_i, j, i) : Y$," the dealer announced a message of the form "$(P^*, j, i) : X$," and $X \neq Y$.
3. A party $P_i$ that is not unhappy becomes *accusatory* if, in round 7, for some unhappy party $P_j$, the dealer broadcasts polynomials $g_j(x)$ and $h_j(y)$, and $P_i$ broadcasts $b'_{i,j}$ and $c'_{i,j}$ with $g_j(i) \neq b'_{i,j}$ or $h_j(i) \neq c'_{i,j}$.

We remark that because broadcast is used in round 7, all parties agree on which parties are unhappy or accusatory.

4. The dealer is disqualified if any of the following conditions hold:
   (DQ.1) There exists an ordered pair $(i, j)$ such that the dealer does not announce a message of the form "$(P^*, i, j) : X$."
   (DQ.2) There exists an unhappy party $P_i$ such that the dealer does not broadcast $g_i(x)$ or $h_i(y)$ in round 7.
   (DQ.3) The number of unhappy and accusatory parties exceeds $t$.
5. If the dealer has not been disqualified, then the parties determine their output the same way as in $\Pi_{(6,2)}$.

We now make two observations regarding the protocol $\Pi_{(7,1)}$:

- If an honest party $P_i$ sends a message $m$ to all parties in round 5, then $P_i$ will be considered as announcing $m$ by the end of round 7.
- If a (possibly malicious) party $P_i$ announces a message $m$, then every honest party received $m$ from at least $t + 1$ different parties in round 6.

If an honest party $P_i$ sends a message $m$ to all parties in round 5, then all honest parties receive it. Since all honest parties broadcast this information in round 7 and there are at least $n - t$ of them, the first condition above holds. If a party $P_i$ announces a message $m$ in round 5, then, by definition, in round 7 at least $n - t$ parties broadcast that they received $m$ from $P_i$ in round 5. At least $n - t - t \geq t + 1$ of them are honest. These honest parties will forward $m$ to all parties in round 6. Hence the second condition above holds.

With the above observations, it is not hard to see that $\Pi_{(7,1)}$ will preserve the commitment property of $\Pi_{(6,2)}$. Now we argue that secrecy is preserved as well. The only issue is that, if $P_i$ is malicious, then in round 7 an honest party $P_k$ may broadcast $h_k(i)$ and $g_k(i)$ (or an honest dealer may broadcast $g_i(x)$ and $h_i(y)$) even if $P_i$ is not considered unhappy by the end of the protocol. However, this does not affect secrecy since the malicious $P_i$ already knows these values.

We defer the full description of the protocol $\Pi_{(7,1)}$ and the proof of correctness to the full version.

### 3.3 Secure Multiparty Computation Using One Round of Broadcast

In this section, we describe how parties can share their inputs and generate random multiplication triples using only one round of broadcast. As discussed in Section 1.1, following such a "setup" phase the parties can then compute their respective outputs *without using any additional invocations of broadcast* using the techniques of Beaver [3]. For completeness, this too is discussed below.

As stated in Section 3.1, we can construct an $(11, 3)$-round protocol $\Pi_i$ that simultaneously allows a party $P_i$ to share its input and generate sufficiently-many random multiplication triples. In the resulting protocol, broadcast is invoked in the $7^{\text{th}}, 9^{\text{th}}$, and $10^{\text{th}}$ rounds. We now show how to transform $\Pi_i$ into a $(21, 1)$-round protocol $\Pi_i'$ with the following properties: (1) By the end of the

protocol, all honest parties output a common bit $\mathsf{trust}(i)$; (2) if $P_i$ is honest, then $\mathsf{trust}(i) = 1$. Moreover, the view of the adversary remains independent of $P_i$'s input; (3) if $\mathsf{trust}(i) = 1$, then $P_i$'s input as well as all the random multiplication triples have been 2-level shared. Furthermore, given the view of the adversary, the first two components of each multiplication triple $(a, b, c)$ are uniformly distributed in the field $\mathbb{F}$.

$\Pi_i'$ proceeds as follows: Each party $P_j$ initializes a binary flag $f_j$ to 1. Roughly speaking, the flag $f_j$ indicates whether $P_j$ "trusts" $P_i$ or not. The parties then run an execution of $\Pi_i$. When a party $P$ is directed by $\Pi_i$ to send message $m$ to another party over a point-to-point channel, it simply sends this message. When a party $P$ is directed to broadcast a message $m$ in the $7^{\mathsf{th}}$ or $9^{\mathsf{th}}$ round of $\Pi_i$, all parties run the following "simulated broadcast" sub-routine:

- $P$ gradecasts the message $m$.
- Each party $P_i$ gradecasts the message it output in the previous step.
- Let $(m_j, g_j)$ and $(m_j', g_j')$ be the output of party $P_j$ in steps 1 and 2, respectively. Within the underlying execution of $\Pi_i$, party $P_j$ will use $m_j'$ as the message "broadcast" by $P$. Furthermore, $P_j$ sets $f_j := 0$ if either (or both) of the following conditions hold: (1) $g_j' \neq 2$, or (2) $m_j' \neq m_j$ and $g_j = 2$.

In the $10^{\mathsf{th}}$ round of $\Pi_i$, when a party $P_j$ is directed to broadcast a message $m$, it simply broadcasts this message along with the flag $f_j$. If fewer than $2t + 1$ parties broadcast $f_j = 1$, then all parties set $\mathsf{trust}(i) = 0$; otherwise, all parties set $\mathsf{trust}(i) = 1$.

The transformation from $\Pi_i$ to $\Pi_i'$ is similar to the compilation of VSS to *moderated* VSS in [22], except that we retain the last invocation of broadcast in $\Pi_i$. The proof of correctness is similar and is omitted due to space limitations.

We now describe how to use the above to obtain a secure MPC protocol.

**The information-theoretic setting.** Suppose the given circuit has $K$ multiplication gates. Following the approach in [7], we can construct the following error-free multiparty computation protocol:

1. For $1 \leq i \leq n$, protocol $\Pi_i'$ is executed in parallel (i.e., $P_i$ shares its input values and generates[5] $K$ random multiplication triples). At the end of this step, all parties agree on values $\mathsf{trust}(i)$ for $i \in [n]$.
2. For each $i$, if $\mathsf{trust}(i) = 0$, then $P_i$ is disqualified and default values are used as the input values of $P_i$. Let $i^*$ be the minimum value such that $\mathsf{trust}(i^*) = 1$. In the next step, parties use the multiplication triples generated in $\Pi_{i^*}'$.
3. The parties evaluate the circuit gate by gate. Suppose that values $x$ and $y$, representing the values on the two input-wires of some gate in the circuit, have been 1-level shared. The value of the output wire can be 1-level shared as follows:

   **Addition gate:** This is easy to do non-interactively.

[5] An optimization is to have each party generate $K/(n - t)$ random multiplication triples, and then use (in step 3, below) the multiplication triples generated by the first $n - t$ non-disqualified parties.

**Multiplication gate:** Using the method suggested in [3], this can be reduced to one round of value reconstruction while consuming one random multiplication triple $(a, b, c)$. Specifically: the parties publicly reconstruct $d_x = x - a$ and $d_y = y - b$. Then, parties non-interactively compute shares of $d_x d_y + d_x b + d_y a + c$ (using their shares of $a, b, c$). Note that if $c = ab$, then $d_x d_y + d_x b + d_y a + c = xy$ (recall that calculations are performed in a field of characteristic 2).

3. Output values are reconstructed by having all parties send appropriate shares to the appropriate parties and using error correction.

The above protocol invokes one round of broadcast. The total number of rounds required is equal to the depth of the circuit being computed plus 22.

**The computational setting.** Assuming the existence of one-way functions, Damgård and Ishai [10] give a multiparty computation protocol with round complexity $(O(1), O(1))$. Roughly speaking, they transform evaluations of a given circuit into evaluations of degree-3 polynomials. Using the approach described in the last section, we can obtain an $(O(1), 1)$-round MPC protocol. (Details are omitted due to space constraints.) The end result is that we obtain a $(26, 1)$-round MPC protocol. Using the expected constant-round broadcast protocol from [22], the (expected) round complexity of the MPC protocol becomes 41.[6]

## 4 Secure Multiparty Computation for $t < n/2$

For $t < n/2$, we assume a PKI and a secure digital signature scheme. Our protocol is based on the protocols in [9,10]. On a high level, the construction is similar to the case of $t < n/3$ with the following differences:

1. Since $t$ may be greater than $n/3$, we can no longer apply Reed-Solomon decoding to reconstruct shared values. Instead, we use the linear information checking tool from [9]. Unfortunately, their protocol as described requires additional invocations of broadcast. We show how to eliminate this usage of broadcast by utilizing the PKI.
2. The presence of a PKI enables us to "catch" a malicious party who cheats more easily. For instance, if a malicious party $P_i$ sends two contradicting messages (with valid signatures) to $P_j$ and $P_k$, then the latter two parties can conclude that $P_i$ is cheating upon exchange of messages. This allows us to construct more round-efficient protocols.
3. As in the case of $t < n/3$ (see Section 3.1), in the protocol for sharing a random multiplication triple $(a, b, c)$, the parties first share two random field elements $a$ and $b$ and then each party shares $a_i b_i$ (where $a_i$ and $b_i$ are the shares held by $P_i$ with respect to $a$ and $b$) and proves that the correct value has been shared. In order for the parties to compute their shares of $c$ (by

[6] Even though broadcast is not used in the final round of the resulting protocol, we do not need to introduce special techniques to deal with issues of non-simultaneous termination since only secrets are reconstructed after broadcast is invoked.

applying Lemma 2), there should exist a set of $2t+1$ parties $P_i$ that have correctly shared $a_i b_i$. For $t < n/3$, this condition is always satisfied since there are at least $2t+1$ honest parties. However, for $t < n/2$, we can only guarantee that $t+1$ (honest) parties will correctly share the product of their shares. Hence the protocol needs to be designed in such a way that if a party does not share the product of its shares correctly, then its shares will be made public.

Due to lack of space, we defer the actual details of the construction to the full version. The round complexity of the final MPC protocol we construct is $(34, 1)$. When the protocol is compiled for a point-to-point network, the round complexity is 64 (in expectation).

## 5 Conclusion

Previous work on round complexity has (for the most part) aimed to minimize *the total number of rounds* for a given task, but under the assumption of a broadcast channel "for free". In fact, broadcast is not for free since emulating broadcast over point-to-point channels is rather expensive. We argue here that if one is ultimately interested in round-efficient protocols for point-to-point networks (which is where most protocols would eventually be run), then it is more productive to focus on minimizing *the number of rounds in which broadcast is used*. With this motivation, we have shown here protocols for secure multi-party computation in a number of settings that use broadcast in a *single* round.

A number of interesting open questions are suggested by our work:

1. The work of [17,15] characterizes the round complexity of VSS (for $t < n/3$) when a broadcast channel is available. Can we obtain a similar characterization of the round complexity of VSS in a point-to-point network? As a step toward this goal, one might start by establishing the optimal round complexity of VSS when the broadcast channel is used only *once*.
2. Our $(O(1), 1)$-round MPC protocol for $t < n/2$ assumes the existence of a PKI. Does there exist a constant-round MPC protocol using a single round of broadcast that does *not* rely on a PKI? Although a PKI will anyway be needed to implement broadcast, the question is of theoretical interest. Furthermore, such a protocol may be useful in settings (such as a small-scale wireless network) when a broadcast channel *is* available, or when it is desirable to minimize the usage of digital signatures for reasons of efficiency.

## References

1. D. Beaver. Multiparty protocols tolerating half faulty processors. In *Advances in Cryptology — Crypto '89*, pages 560–572. Springer-Verlag, 1989.
2. D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

3. D. Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology — Crypto '91*, pages 420–432. Springer-Verlag, 1992.
4. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, 1990.
5. M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *2nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 1983.
6. M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.
8. D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, 1988.
9. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology — Eurocrypt '99*, volume 1592 of *LNCS*, pages 311–326. Springer-Verlag, 1999.
10. I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Adv. in Cryptology — Crypto 2005*, pages 378–394. Springer-Verlag, 2005.
11. D. Dolev and H. Strong. Authenticated algorithms for Byzantine agreement. *SIAM J. Computing*, 12(4):656–666, 1983.
12. P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.
13. M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Info. Proc. Lett.*, 14(4):183–186, 1982.
14. M. Fitzi and J. A. Garay. Efficient player-optimal protocols for strong and differential consensus. In *22nd Annual ACM Symposium on Principles of Distributed Computing*, pages 211–220, 2003.
15. M. Fitzi, J. A. Garay, S. Gollakota, C. P. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *3rd Theory of Cryptography Conference*, pages 329–342, 2006.
16. J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.
17. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd Annual ACM Symposium on Theory of Computing*, pages 580–589, 2001.
18. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology — Crypto 2002*, pages 178–193. Springer-Verlag, 2002.
19. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computation with applications to threshold cryptography. In *Proc. 17th Annual ACM Symposium on Principles of Distributed Computing*, pages 101–111. ACM Press, 1998.
20. S. Goldwasser and Y. Lindell. Secure computation without agreement. In *16th Intl. Conf. on Distributed Computing (DISC)*, pages 17–32. Springer-Verlag, 2002.

21. M. Hirt, J. B. Nielsen, and B. Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience. In *Adv. in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 322–340. Springer-Verlag, 2005.
22. J. Katz and C.-Y. Koo. On expected constant-round protocols for Byzantine agreement. In *Adv. in Cryptology — Crypto 2006*. Full version available at http://eccc.hpi-web.de/eccc-reports/2006/TR06-028/index.html.
23. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
24. Y. Lindell, A. Lysyanskaya, and T. Rabin. Sequential composition of protocols without simultaneous termination. In *Proc. 21st Annual ACM Symposium on Principles of Distributed Computing*, pages 203–212, 2002.
25. S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *Adv. in Cryptology — Crypto '90*, pages 253–266. Springer-Verlag, 1991.
26. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
27. B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.
28. M. Rabin. Randomized Byzantine generals. In *Proc. 24th IEEE Symposium on Foundations of Computer Science*, pages 403–409, 1983.
29. T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *J. ACM*, 41(6):1089–1109, 1994.
30. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st Annual ACM Symposium on Theory of Computing*, pages 73–85. ACM Press, 1989.

## A Round Complexity of Emulating Broadcast

In this section, we discuss the round complexity of emulating broadcast by the most round-efficient randomized protocol known [22], The randomized broadcast protocols of [22] allow all parties to broadcast a message simultaneously.[7] Roughly speaking, the protocols consist of two phases. The first phase is a "setup" phase that is independent of the messages being broadcast, and (only) consists of parallel executions of moderated VSS. Using the VSS protocol developed in this work, for $t < n/3$, this initial phase can be implemented in (strict) 12 rounds.

The execution of the second phase depends on the messages being broadcast, and terminates in 16 rounds (in expectation) assuming $t < n/3$. For a single invocation of broadcast, the first 5 rounds of the second phase can be executed in parallel with the last 5 rounds of the first phase, and hence the round complexity of the entire broadcast protocol (in expectation) is 23. (See [22, Appendix C] for

[7] Note, however, that the protocols do not achieve the "simultaneous broadcast" functionality (i.e., with all broadcast messages being independent of each other). Instead, they simply emulate a round of broadcast with rushing.

further details. Note that the numbers computed there do not take into account the more efficient VSS protocol constructed here.)

However, if broadcast is invoked multiple times sequentially the round complexity does not simply scale linearly. The reason is that the second phase does not guarantee simultaneous termination and so sequential executions do not compose directly; instead, additional steps (which increase the round complexity) are needed. Without going into the details (see [22, Appendix C]), it is possible to show using the techniques of [6,24,22] that emulation of multiple rounds of broadcast requires 32 rounds (in expectation) per additional broadcast, in addition to an initial 23 rounds.

As stated in the Introduction, the Micali-Rabin VSS protocol uses 16 rounds but only a single round of broadcast. In fact, the broadcast is used in the final round. If the first 15 rounds of the Micali-Rabin protocol are executed in parallel to the first phase of the broadcast protocol, then the Micali-Rabin protocol takes $31 = 15+16$ rounds (in expectation) when compiled for a point-to-point network. On the other hand, the protocol by Fitzi et al. requires $55 = 32 + 23$ rounds since they use two rounds of broadcast. (Messages sent during the one round of their protocol that does not use broadcast can be "piggy-backed" on messages sent as part of the broadcast protocols.)

For the case of $t < n/2$, the numbers are even worse: a single invocation of broadcast takes 36 rounds (in expectation), while 58 rounds are needed per additional broadcast when broadcast is used in two or more rounds.