ABSTRACT

Title of dissertation: STUDIES ON FAULT-TOLERANT BROADCAST AND SECURE COMPUTATION

Chiu Yuen Koo
Doctor of Philosophy, 2007

Dissertation directed by: Professor Jonathan Katz
Department of Computer Science

In this dissertation, we consider the design of broadcast and secure multi-party computation (MPC) protocols in the presence of adversarial faults.

Secure multi-party computation is the most generic problem in fault-tolerant distributed computing. In principle, a multi-party computation protocol can be used to solve *any* distributed cryptographic problem. Informally, the problem of multi-party computation is the following: suppose we have n parties $P_1, P_2, \ldots, P_n$ where each party P_i has a private input x_i. Together, the parties want to compute a function of their inputs $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$. However, some parties can be *corrupted* and do not execute a prescribed protocol faithfully. Even worse, they may be controlled by an adversary and attack the protocol in a coordinated manner. Despite the presence of such an adversary, a secure MPC protocol should ensure that each (corrupted) party P_i learn only its output y_i but nothing more.

Broadcast in the presence of adversarial faults is one of the simplest special cases of multi-party computation and important component of larger protocols. In short, broadcast allows a party to send the same message to all parties, and all parties to be assured they have received identical messages.

The contribution of this dissertation is twofold. First, we construct broadcast and secure multi-party computation protocols for honest majority in a point-to-point network whose round complexities improve significantly upon prior work. In particular, we give the first expected constant-round authenticated broadcast protocol for honest majority assuming only public-key infrastructure and signatures. Second, we initiate the study of broadcast in radio networks in the presence of adversarial faults. In radio networks, parties communicate through multicasting messages; a message can only be received by the parties within some radius from the sender. Feasibility and impossibility results are given, and our bounds are tight.

STUDIES ON FAULT-TOLERANT BROADCAST AND SECURE COMPUTATION

by

Chiu Yuen Koo

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

ACKNOWLEDGMENTS

I would like to thank my advisor Jonathan Katz, for his guidance and support. I would also like to thank him for giving me the freedom to work on problems which interest me.

The results in this thesis were obtained in collaboration with my advisor. In addition, part of the results in Chapter 5 are due to joint work with Vartika Bhandari (UIUC) and Nitin Vaidya (UIUC).

I am grateful to Juan Garay for taking me on as a summer intern in Bell Labs in 2006. I would like to thank Jonathan for introducing me to Juan.

I would like to thank the Institute for Pure and Applied Mathematics at UCLA for inviting me to take part in its semester-long program "Securing Cyberspace : Applications and Foundations of Cryptography and Computer Security", as well as providing financial support during the period. I had a wonderful time there.

I thank Tak-Wah Lam (University of Hong Kong) for introducing me to theoretical computer science research, by letting me to work with him when I was an undergraduate.

I thank all my friends for making my graduate life enjoyable.

Last but not least, I would like to thank my parents and my sisters for their unconditional support.

Table of Contents

List of Figures

Chapter 1

Introduction

In this dissertation, we consider the design of *broadcast* and *secure multi-party computation* (MPC) protocols in the *presence of adversarial faults.*

Secure multi-party computation is the most generic problem in fault-tolerant distributed computing. In principle, a multi-party computation protocol can be used to solve *any* distributed cryptographic problem. Informally, the problem of multi-party computation is the following: suppose we have n parties $P_1, P_2, \ldots, P_n$ where each party P_i has an input x_i. Together, the parties want to compute a function of their inputs $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$. However, some parties are *corrupted* and *malicious.* These corrupted parties may not execute a prescribed protocol faithfully. Even worse, they may be controlled by *an adversary* and attack the protocol in a coordinated manner. Despite the presence of such an adversary, a secure MPC protocol should ensure that each (corrupted) party P_i learn only y_i but nothing more.

Broadcast in the presence of adversarial faults is one of the simplest special cases of multi-party computation. In short, broadcast allows a party to send the same message to all parties. It is a useful building block for MPC protocols and other distributed cryptographic protocols. More formally, the broadcast problem can be stated as follows: there is a distinguished party, known as the *dealer*, who holds a message m. However, some parties (including the dealer) can be corrupted by an adversary. Despite this, all honest (i.e., uncorrupted) parties should eventually output a common message m'. In addition, if the dealer is honest, then m' should be equal to m.

In this dissertation, we consider two different kinds of communication models: *point-to-point networks* and *radio networks*. In point-to-point networks, all pairs of parties communicate through direct links. On the other hand, in radio networks, parties communicate through a wireless channel. Two parties can communicate directly only if they are within the transmission range of each other. Furthermore, every party in the range hears the message being transmitted.

This dissertation is divided into two parts. In the first part, we study the round efficiency of broadcast and secure multi-party computation protocols in point-to-point networks under the standard model from the literature. In the second part, we propose an adversarial model for corruption of parties in radio networks and study the feasibility of broadcast under this model.

Preliminary version of the work in this dissertation appeared in [KK06, KK07, Koo04, KBKV06].

1.1 Part One: Round-Efficient Protocols in Point-to-Point Networks

The round complexity of cryptographic protocols has been the subject of intense study. Much work has been done on establishing bounds on the round complexity of various tasks such as zero knowledge [BCY89, FS89, GK96, BJY97, HT98, Ros00, CKPR01, PRS02], broadcast (Byzantine agreement) [PSL80, LSP82, FL82, DS83, Bra87, CC85, FM85, DSS90, BG93, MW94, FM97, GM98, BB98, BE03], verifiable secret sharing [GIKR01, FGG$^+$06], and secure multi-party computation [Yao86, BMR90, IK00, GIKR02, KOS03, Lin03, KO04, DI05].

Apart from the fact that these results are of fundamental theoretical importance, 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.

In this part, we focus on constructing round-efficient protocols for broadcast and secure multi-party computation in *synchronous* point-to-point networks in the presence of *honest majority*, i.e., at least half of the parties are honest. In a synchronous network, an execution of protocol takes place in rounds. In each round, parties send messages to each other depending on the messages they have received from the previous rounds; the parties also receive the messages being sent to them within the same round.

1.1.1 Broadcast

When designing cryptographic protocols, it is often convenient to abstract away various details of the underlying communication network. As one noteworthy example, it is often convenient to assume the existence of a *broadcast channel* which allows any party to send the same message to all other parties (and all parties to be assured they have received identical messages) in a single round. In most cases, it is understood that the protocol will be run in a network where only point-to-point communication is available and the parties will have to "emulate" the broadcast channel by running a broadcast protocol. Unfortunately, this "emulation" typically increases the round complexity of the protocol substantially.

Work has been done on reducing the round complexity of protocols for broadcast or the related task of *Byzantine agreement* (BA) [PSL80, LSP82]. In Byzantine agreement, each party has an initial input. Eventually, all parties have to output a *common value*. The requirement is that if all (honest) parties have the same input value, then the parties have to output that particular value. In the case of honest majority, any Byzantine agreement protocol implies a broadcast protocol using one additional round (in the first round the sender sends its message to all parties, who then run a Byzantine agreement protocol on the values they received). Note that Byzantine agreement is not defined if there is an absence of honest majority.

Related Work. The Byzantine agreement problem is introduced by Pease, Shostak and Lamport [PSL80, LSP82] who show that in a synchronous network with pairwise authenticated channels and no additional set-up assumptions, BA among n parties is achievable if and only if the number of corrupted parties t satisfies $t < n/3$. Concerning round complexity, a lower bound of $t + 1$ rounds for any deterministic BA protocol is known in this setting [FL82]. A protocol with this round complexity — but with exponential message complexity — was shown by Pease, et al. [PSL80, LSP82]. Following a long sequence of works, Garay and Moses [GM98] show a fully-polynomial BA protocol with optimal resilience and round complexity.

To circumvent the above-mentioned lower bound, researchers beginning with Rabin [Rab83] and Ben-Or [B83] explored the use of randomization to obtain better round complexity. This line of research [Bra87, CC85, FM85, DSS90] culminated in the work of Feldman and Micali [FM97], who show a randomized BA protocol with optimal resilience $t < n/3$ that runs in an expected *constant* number of rounds. Their protocol requires channels to be both private and authenticated.

To achieve resilience $t \geq n/3$, additional assumptions are needed even if randomization is used [KY]. The most widely-used assumptions are the existence of digital signatures and a public-key infrastructure (PKI); protocols in this setting are termed *authenticated*. Implicit in this setting is that the adversary cannot forge signatures. Pease, et al. [PSL80, LSP82] show an authenticated broadcast protocol for $t < n$, and a fully-polynomial protocol achieving this resilience was given by Dolev and Strong [DS83]. These works rely only on the existence of digital signature schemes and a PKI, and do not require private channels. Digital signature schemes can be constructed from one-way functions [NY89, Rom90]; however, the schemes will only be secure against a computationally bounded adversary; alternatively, if information-theoretic "pseudo-signatures" [PW96] are used, security can be obtained even against an unbounded adversary.

The $(t+1)$-round lower bound for deterministic protocols holds in the authenticated setting as well [DS83], and the protocols of [PSL80, LSP82, DS83] meet this bound. Some randomized protocols beating this bound for the case of $n/3 \leq t < n/2$ are known [Tou84, Bra87, Wai91], but these are only partial results:

- Toueg [Tou84] gives an expected $O(1)$-round protocol, but assumes a trusted dealer. After the dealing phase the parties can only run the BA protocol a *bounded* number of times.

- A protocol by Bracha [Bra87] implicitly requires a trusted dealer to ensure that parties agree on a "Bracha assignment" in advance (see [FM85]). Furthermore, the protocol only achieves expected round complexity $\Theta(\log n)$ and tolerates (slightly sub-optimal) $t \leq n/(2+\epsilon)$ for any $\epsilon > 0$.

- Waidner [Wai91], building on [Bra87, FM85], shows that the dealer in Bracha's protocol can be replaced by an $\Omega(t)$-round pre-processing phase during which a broadcast channel is assumed. The expected round complexity (after the pre-processing) is also improved from $\Theta(\log n)$ to $\Theta(1)$.

The latter two results assume private channels.

Fitzi and Garay [FG03], building on [Tou84, CKS00, Nie02], give the first full solution to this problem: that is, they show the first authenticated BA protocol with optimal resilience $t < n/2$ and expected constant round complexity that does not require any trusted dealer or pre-processing (other than a PKI). Even assuming private channels, however, their protocol requires specific number-theoretic assumptions (essentially, some appropriately-homomorphic public-key encryption scheme) and cannot be based on signatures alone. Because of its reliance on additional assumptions, the Fitzi-Garay protocol cannot be adapted to the information-theoretic setting using pseudo-signatures.

Our Contributions

As our main result, in Chapter 3, we extend the work of Feldman and Micali and show that:

Theorem 1.1.1 *Assuming a public-key infrastructure and the existence of signature schemes, there exists a broadcast protocol tolerating $t < n/2$ malicious parties running in expected constant rounds.*

For those unfamiliar with the specifics of the Feldman-Micali protocol, we stress that their approach does *not* readily extend to the case of $t < n/2$. In particular, they rely on a primitive termed *graded VSS* and construct this primitive using in an essential way the fact that $t < n/3$. We take a different approach: we introduce a new primitive called *moderated VSS* (mVSS) and use this to give an entirely self-contained proof of our result.

We suggest that mVSS is a useful alternative to graded VSS *in general*, even when $t < n/3$. For one, mVSS seems easier to construct: we show a generic construction of mVSS *in the point-to-point model* from any VSS protocol that relies on a broadcast channel, while a generic construction of this sort for graded VSS seems unlikely. Perhaps more importantly, mVSS provides what we believe to be a conceptually-simpler approach to the problem at hand: in addition to our authenticated broadcast protocol for $t < n/2$, our techniques give a broadcast protocol (in the plain model) for $t < n/3$ that is both more round-efficient than the Feldman-Micali protocol and also admits a self-contained proof that is, in our opinion, significantly simpler than that of [FM97]. Moreover, the concept of *moderated* protocols proved to be an useful technique in developing round-efficient secure multi-party computation protocols (see discussion in the next section).

As mentioned earlier, cryptographic protocols are often designed under the assumption that a broadcast channel is available; when run in a point-to-point network, these protocols must "emulate" the broadcast channel by running a broadcast protocol as a sub-routine. If the original protocol uses multiple invocations of the broadcast channel, and these invocations are each emulated using a *probabilistic* broadcast protocol, subtle issues related to the parallel and sequential *composition* of the various broadcast sub-protocols arise; see the detailed discussion in Section 3.3. Parallel composition can be dealt with using existing techniques [BE03, FG03]. There are also techniques available for handling sequential composition of protocols without simultaneous termination [BE03, LLR02]; however, it applies only to the case $t < n/3$ [BE03] or else is rather complex [LLR02]. As an additional contribution, we show how to extend previous work [BE03] so as to enable sequential composition when $t < n/2$ (assuming digital signatures and a PKI) in a simpler and more round-efficient manner than [LLR02].

The above results, in combination with prior work [BMR90, DI05], yield the following result:

Theorem 1.1.2 *Assuming a public-key infrastructure and the existence of signature schemes, there exists a secure multi-party computation protocol tolerating $t < n/2$ malicious parties running in expected constant rounds.*

A preliminary version of this work appeared in [KK06].

1.1.2 Secure Multi-Party Computation

Secure multi-party computation (MPC) enables a group of parties to evaluate a function f with the guarantee that each honest party will learn its output and each corrupted party will *only* learn its own output but nothing more.

In below, we present the definition of secure computation *without abort* (the definition is adapted from [GL05]), which is the standard definition used for the case of honest majority. There are other (relaxed) definitions of secure computation (see [GL05]), but we will not consider those here.

Multi-party function evaluation A multi-party computation problem for n parties is cast by specifying a random process that maps vectors of inputs to vectors of outputs (one input and one output for each party). We denote such process $f : (\{0,1\}^\star)^n \to (\{0,1\}^\star)^n$, where $f = (f_1, \ldots, f_n)$. That is, for a vector of inputs $\bar{x} = (x_1, \ldots, x_n)$, the output vector is a random variable $(f_1(\bar{x}), \ldots, f_n(\bar{x}))$ ranging over vectors of strings. The output for the i^{th} party (with input x_i) is defined to be $f_i(\bar{x})$.

The security of a multi-party computation protocol is analyzed by comparing what an adversary (who controls the corrupted parties) can do in the protocol to what it can do in an *ideal* scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible *trusted party* as follows:

Execution in the Ideal Model

1. Inputs: Each party P_i obtains its respective input x_i.

2. Send inputs to trusted party: An honest party P_i always sends its input x_i to the trusted party. A corrupted party, on the other hand, may send modified value x'_i to the trusted party; x'_i can be dependent on the inputs of other corrupted parties. Let the sequence of inputs obtained by the trusted party be $\bar{x}'$.

3. Trusted party answers the parties: The trusted party computes $f(\bar{x}')$ and sends $f_i(\bar{x}')$ to party P_i for every i.

4. Outputs: An honest party always outputs the message that it received from the trusted party and the corrupted parties output nothing. The adversary outputs an arbitrary function of the initial inputs of the corrupted parties and the messages that the corrupted parties received from the trusted party.

Execution in the Real Model: In the real model, the parties execute a protocol Π to evaluate f. Throughout the execution, the honest parties follow the instructions of the prescribed protocol, while the corrupted parties may deviate from the protocol in an arbitrary manner, subject to the choice of the adversary. At the end of the protocol execution, the honest parties output their prescribed output from Π, the corrupted parties output nothing and the adversary outputs its view of the computation.

Secure MPC: A protocol Π is said to be a secure MPC protocol for the computation of f if for every adversary for the real model, there exists an ideal-world adversary such that the following two distributions are indistinguishable:

- The joint output of the honest parties and the real-world adversary in the real model.
- The joint output of the honest parties and the ideal-world adversary in the ideal model.

As mentioned earlier, we will focus on studying the round complexity of protocols for secure multi-party computation. Previous research investigating this aspect has almost exclusively focused on optimizing the round complexity *under the assumption that a broadcast channel is available.* (We survey some of this work later in this section.) In most settings where MPC 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, as mentioned in the last section, a 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 [GM98, DS83], then a lower bound due to Fischer and Lynch [FL82] 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) where t is the number of malicious parties. 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 (see Chapter 3). 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 one must explicitly handle sequential composition of protocols without simultaneous termination. (This is not an issue if broadcast is used in only a *single* round.) Unfortunately, this leads to a substantial increase in round complexity. See Section 3.3 for details.

To illustrate the second point, consider the protocols of Micali and Rabin [MR90] and Fitzi, et al. [FGG$^+$06] (building on [GIKR01]) 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 result from Chapter 3, 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** *protocols for secure computation that use only a* **single** *round of broadcast*?

Note that this is optimal in terms of the usage of the broadcast channel for a constant-round secure MPC protocol, since secure computation of the broadcast functionality requires a single invocation of the broadcast channel.

Our Contributions

As our main results, in Chapter 4, we show that:

Theorem 1.1.3 *Assuming the existence of one-way functions, there is a constant-round secure multi-party computation protocol tolerating $t < n/3$ malicious parties that uses a single round of broadcast.*

Theorem 1.1.4 *Assuming the existence of one-way functions and a public-key infrastructure, there is a constant-round secure multi-party computation protocol tolerating $t < n/2$ malicious parties that uses a single round of broadcast.*

We obtain the above results by using the concept of moderated protocols (see previous section). Along with the construction of our protocols, we also develop techniques to minimize the *exact* round complexity. 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 results in Chapter 3, which contain the most round-efficient known techniques for emulating broadcast over point-to-point channels). For example, using the results from Chapter 3, 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. 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.

A preliminary version of this work appeared in [KK07].

Prior Work. Initial feasibility results showed the existence of unconditionally-secure MPC protocols in point-to-point networks for $t < n/3$ (combining [BGW88, CCD88] with [PSL80]), or for $t < n/2$ assuming a broadcast channel is available [Bea91b, RB89, Rab94].

Beaver, Micali, and Rogaway [BMR90] gave the first constant-round protocol for secure MPC with $t < n/2$, assuming a broadcast channel and one-way functions. Damgård and Ishai [DI05] showed a constant-round protocol under the same assumptions that is secure even for adaptive adversaries. As mentioned in the previous section, using our results, these can both be converted to *expected* constant-round protocols in point-to-point networks. We stress that the constant obtained in this way is rather high, on the order of hundreds of rounds.

The work of Gennaro, et al. [GIKR01] implies a 3-round MPC protocol with resilience $t < n/4$, assuming the existence of one-way functions. We remark that the resulting protocol only uses broadcast in a single round, and so would yield a very round-efficient protocol in a point-to-point network; the drawback is that the resilience is not optimal. In subsequent work [GIKR02], 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 $t < n/6$.

Goldwasser and Lindell [GL05] show various round-efficient secure MPC protocols for point-to-point networks that do not use broadcast; however, their work considers weakened security definitions in which fairness and output delivery are not guaranteed (even when an honest majority exists).

1.2 Part Two: Feasibility of Broadcast in Radio Networks

Much work has focused on the broadcast problem in a fully connected point-to-point network. (We have surveyed these work in Section 1.1.1). Some research has also explored the problem when pairwise channels exist only between selected pairs of parties, or under the assumption of "k-cast channels" shared by all subsets of parties of size k [FM00, ASS$^+$03, CFF$^+$05].

However, none of these models are appropriate for radio networks in which a party can communicate only by multicasting a message which is then received by all parties within some radius r (i.e., the neighbors of the transmitting party). With recent advancements in wireless technology, deployment of large-scale networks in which the sole means of communication is via wireless (radio) transmission is now possible. Since broadcast can serve as a building block for many applications in these enviroments, it is of interest to establish the conditions under which it can be achieved. Yet, as far as we are aware, prior to our work, obtaining broadcast in radio networks in the presence of malicious parties has not been studied before. Our work corrects this omission, and provides the first analysis of broadcast in radio networks.

Our Contributions We study feasibility of broadcast in the following network model: parties are located on an infinite grid (each grid unit is a 1×1 square). In the absence of *collisions*, if a party locates at (x, y), $P(x, y)$, *multicasts* a message m, then all parties within distance r will receive the message. The parties within the distance r from (x, y) are known as the *neighbors* of $P(x, y)$. A collision at (x, y) occurs when two neighbors P_1, P_2 of $P(x, y)$ multicast at the same time. In this case, there is no guarantee as to what message(s) $P(x, y)$ will receive. (The square grid model has been considered in [KKP01], in the context of minimizing broadcast latency with crash failures.) We assume there exists a pre-determined time division multiple access (TDMA) schedule such that if all parties follow the schedule in carrying out local broadcast, then no collisions will occur.

We introduce the locally-bound fault model – which we believe is a natural model for the distribution of corruption in radio networks. Under this fault model, an adversary can corrupt up to t neighbors of any party.

A party corrupted by the adversary is allowed to deviate from the TDMA schedule, cause *message collision* and send out *spoofed address* for a bounded number of times. We assume the bound are known in advance by all parties.

As our main result, in Chapter 4, we show the following:

Theorem 1.2.1 *In the L_∞ metric, if $t < \frac{1}{2}r(2r+1)$, then there exists a protocol achieves broadcast as long as there is a bound on the number of collisions caused and spoofed messages sent by each corrupted party.*

The above result is tight in two ways. First, it is easy to see that there does not exist any broadcast protocol in the presence of an adversary capable of causing unbounded number of collisions. (An adversary can prevent an honest party from receiving any message by continuously causing collisions.) Second, we show that it is not possible to tolerate a larger value of t:

Theorem 1.2.2 *In the L_∞ metric, if $t \geq \frac{1}{2}r(2r+1)$, broadcast is impossible even if the adversary cannot cause collisions nor send out spoofed address.*

A preliminary version of this work appeared in [Koo04] and [KBKV06]. In the first paper, we study broadcast assuming the adversary cannot cause collisions nor send out spoofed address. Feasibility and impossibility results are shown. The feasibility results are subsequently improved by Bhandari and Vaidya [BV05a], they give an upper bound that matches the lower bound we gave in [Koo04] (as in [Koo04], they assume the adversary cannot cause collisions nor send out spoofed address). In [KBKV06], we show how to achieve the same upper bound as in [BV05a] even if the adversary is allowed to cause a bounded number of collisions or sending out a bounded number of spoofed address.

Related and Subsequent Work. Prior work on broadcast in radio networks mostly focus on minimizing the broadcast latency assuming the radio networks is *fault-free*. As noted in [KKP01], not many results are known about broadcast in radio networks in the presence of faults. Kranakis, Krizanc and Pelc [KKP01] consider the effect of a passive adversary (a corrupted party will not send any message) on the broadcast latency.

The initial results in [Koo04] are subsequently improved in [BV05a, Vai05]. In the above mentioned work, it was assumed that the adversary *cannot* cause collisions nor carry out address spoofing. Under this assumption, [BV05a] gave a protocol that achieves broadcast when $t < \frac{1}{2}r(2r+1)$ for the L_∞ metric. An approximate threshold was also established for the L_2 metric (the threshold is shown to be tight asymptotically). In [BV05b], a sufficient condition for broadcast in general graphs under the locally bounded adversarial model was described and simpler broadcast protocols for a grid network (compared to [BV05a]) was presented.

Broadcast in an arbitrary graph was considered in [PP05a]. Upper and lower bounds for achievability of broadcast were presented based on graph-theoretic parameters. However, no exact thresholds were established. It was also shown that there exist certain graphs in which algorithms that work with knowledge of topology succeed in achieving broadcast, while those lacking this knowledge fail.

Random transient failures were considered in [PP05b]. At each step, each party may fail with some constant probability p. Tight bounds on p were obtained concerning the feasibility of broadcast. Random permanent failures in a grid network have been considered in [BV07], and necessary and sufficient conditions on the required transmission range r have been derived.

Achieving consensus in wireless network was studied in [CDG$^+$05], but in a slightly different model. A single-hop wireless broadcast network consisting of fixed but *a priori* unknown collection of parties was considered. Parties can suffer from crash-stop failures and messages can be lost. Necessary and sufficient conditions were derived. Recently, Gilbert, et al.[GGN06] extend the result of [CDG$^+$05] to the case where a party can suffer from Byzantine failure and cause message collisions.

Part I

Round-Efficient Protocols in Point-to-Point Networks

Chapter 2

Model, Technical Preliminaries, and Basic Primitives

2.1 Model and Technical Preliminaries

Here, we describe the model that we consider in Part I and state the technical preliminaries.

We consider the standard synchronous communication model where parties communicate in synchronous rounds using pairwise *private* and *authenticated* channels. Authenticated channels can be realized using signature schemes if one is willing to assume a *public-key infrastructure* (PKI). By a PKI in a network of n parties, we mean that prior to any protocol execution all parties hold the same vector $(pk_1, \ldots, pk_n)$ of public keys for a digital signature scheme, and each honest party P_i holds the honestly-generated secret key sk_i associated with pk_i. Malicious parties may generate their keys arbitrarily, even dependent on keys of honest parties. For static adversaries, private channels can be realized using one additional round by having each party P_i send to each party P_j a public key $PK_{i,j}$ for a semantically-secure public-key encryption scheme (using a different key for each sender avoids issues of malleability). For adaptive adversaries, more complicated solutions are available [BH92, CFGN96] but we do not discuss these further. For simplicity, we assume *unconditional* private/authenticated channels with the understanding that these guarantees hold only computationally if the above techniques are used.

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*. For $t < n/3$ we do not assume any setup, but for $t < n/2$ we assume a PKI and secure signature schemes (this is the authenticated case). Protocols designed under this assumption are termed *authenticated* protocols.

In the protocol descriptions, we implicitly assume that all parties send a properly-formatted message at all times (this is without loss of generality, as we may interpret an improper or missing message as some default message).

When we describe signature computation in authenticated protocols we often omit for simplicity additional information that should be signed along with the message. That is, when we say that party P_i signs message m and sends it to P_j, we implicitly mean that P_i signs the concatenation of m with additional information including: (1) the identity of the recipient P_j, (2) the current round number, (3) an identifier for the message (in case multiple messages are sent to P_j in the same round); and (4) an identifier for the (sub-)protocol (in case multiple sub-protocols are being run; cf. [LLR06]). This information is also verified, as appropriate, when the signature is verified.

In some of our protocol constructions we assume a broadcast channel. A broadcast channel allows any party to send the same message to all other parties (and all parties to be assured they have received identical messages) in a single round. 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). Notice that if a protocol has round complexity $(x, 0)$, then the protocol does not require a broadcast channel. When no broadcast is used sometimes we will just say the protocol uses x rounds.

We use "=" to denote a test for equality, and ":=" to denote variable assignment. We use $[n]$ to denote the set $\{1, \ldots, n\}$.

The standard definition of broadcast follows.

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

Agreement All honest parties output the same value.

Validity If the dealer is honest, then all honest parties output M. $\diamondsuit$

2.2 Basic Primitives

We now define and construct some basic primitives for constructing the protocols in Chapter 3 and Chapter 4.

2.2.1 Gradecast

Gradecast, a relaxed version of broadcast, was introduced by Feldman and Micali [FM97, Def. 11]; we provide a definition which is slightly weaker than theirs but suffices for our purposes.

Definition 2 (Gradecast):

A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $P^* \in \mathcal{P}$ holds an 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 a message m_i and the grade $g_i = 2$, then the output of every honest party P_j satisfies $m_j = m_i$ and $g_j \geq 1$.

$\diamondsuit$

The following result is due to [FM97] and proved for completeness below:

Lemma 2.2.1 *There exists a $(3, 0)$-round gradecast protocol tolerating $t < n/3$ malicious parties.*

Proof The protocol proceeds as follows:

Round 1 The dealer sends M to all other parties.

Round 2 Let M_i denote the message received by P_i (from the dealer) in the previous round. P_i sends M_i to all the parties.

Round 3 Let $M_{j,i}$ denote the message received by P_i from P_j in the previous round. Each party P_i does the following: if there exists an M_i^* such that $|\{j : M_{j,i} =$

$M_i^*\}| \geq 2n/3$ then P_i sends this M_i^* to all the parties. Otherwise, P_i sends nothing.

Output determination Let $M_{j,i}^*$ denote the message (if any) received by P_i from P_j in the previous round. Each party P_i determines its output as follows: if there exists an M_i^{**} such that $|\{j : M_{j,i}^* = M_i^{**}\}| \geq 2n/3$, then P_i outputs $m_i := M_i^{**}$ and $g_i := 2$. Otherwise, if there exists[1] an M_i^{**} such that $|\{j : M_{j,i}^* = M_i^{**}\}| \geq n/3$, then P_i outputs $m_i := M_i^{**}$ and $g_i := 1$. Otherwise, P_i outputs $m_i := \perp$ and $g_i := 0$.

Let us now prove that the above protocol satisfies Definition 2. Assume first that the dealer is honest. Then each honest party P_i receives $M_i = M$ in round 1 and sends this to all parties in round 2. So in round 3, for each honest P_i it holds that $M_i^* = M$ and so P_i sends this value to all the parties. It follows that any honest party P_i outputs $m_i = M$ and $g_i = 2$.

Before proving the second required property, we show that if any two honest parties P_i, P_j send a message in round 3 then they in fact send the *same* message. To see this, say P_i sends M_i^* in round 3. Then P_i must have received M_i^* from at least $2n/3$ parties in round 2, and so strictly more than $n/3$ honest parties must have sent M_i^* in round 2. But this means that P_j receives any value $M_j^* \neq M_i^*$ from strictly fewer than $n - n/3 = 2n/3$ parties in round 2, and so P_j either sends M_i^* or nothing in round 3.

[1]It will follow from the proof below that at most one such M_i^{**} exists in this case.

Now assume there is an honest party P_i who outputs a message m_i and grade $g_i = 2$, and let P_j be any other honest party. P_i must have received m_i from at least $2n/3$ parties in round 3, and so more than $n/3$ honest parties sent m_i as their round-3 message. It follows that $|\{k : M^*_{k,j} = m_i\}| \geq n/3$ and so P_j outputs grade $g_j \geq 1$. Say there was an $m_j \neq m_i$ for which $|\{k : M^*_{k,j} = m_j\}| \geq n/3$. Then at least one honest party sent $m_j \neq m_i$ as its round-3 message, contradicting what we have shown in the previous paragraph. So, P_j outputs message m_i as required.

■

Next, we prove an analogue of the above for the case of *authenticated* gradecast.

Lemma 2.2.2 *There exists a* $(4, 0)$*-round authenticated gradecast protocol tolerating* $t < n/2$ *malicious parties.*

Proof The protocol proceeds as follows:

Round 1 The dealer computes a signature σ of M and sends (M, σ) to all parties.

Round 2 Let (M_i, σ_i) be the message received by party P_i (from the dealer) in the previous round. If σ_i is a valid signature of M_i (with respect to the dealer's public key), then P_i sends (M_i, σ_i) to all other parties; otherwise P_i sets $M_i :=\perp$ and sends nothing.

Round 3 Let $(M_{j,i}, \sigma_{j,i})$ be the message received by P_i from P_j in the previous round. If there exists a j such that $M_{j,i} \neq M_i$ but $\sigma_{j,i}$ is a valid signature of $M_{j,i}$ (with respect to the dealer's public key), then P_i sets $M_i :=\perp$.

If $M_i \neq \perp$, then P_i computes a signature σ'_i of M_i and sends (M_i, σ'_i) to all parties. (If $M_i = \perp$, then P_i sends nothing.)

Round 4 Let $(M'_{j,i}, \sigma'_{j,i})$ be the message received by P_i from P_j in the previous round. If there exist $\ell \geq n/2$ distinct indices $j_1, \ldots, j_\ell$ and a message M^* such that $M'_{j_1,i} = \cdots = M'_{j_\ell,i} = M^*$ and $\sigma'_{j_k,i}$ is a valid signature of M^* (with respect to the public key of P_{j_k}) for $1 \leq k \leq \ell$, then P_i sends $(M^*, j_1, \sigma'_{j_1,i}, \ldots, j_\ell, \sigma'_{j_\ell,i})$ to all other parties and outputs $m_i := M^*$, $g_i := 2$.

Output determination Assuming P_i has not decided on its output, it proceeds as follows: If in the previous round P_i received any message $(M^*, j_1, \sigma'_1, \ldots, j_\ell, \sigma'_\ell)$ for which $\ell \geq n/2$, the $\{j_k\}_{k=1}^{\ell}$ are distinct, and σ'_k is a valid signature of M^* with respect to the public key of party P_{j_k} for $1 \leq k \leq \ell$, then P_i outputs $m_i := M^*$, $g_i := 1$. Otherwise, P_i outputs $m_i := \perp$, $g_i := 0$.

We show the above protocol satisfies Definition 2. If the dealer is honest, then in round 3 every honest party P_i computes a signature σ'_i of the dealer's message M and sends (M, σ'_i) to all other parties. Thus, all honest parties will receive at least $n/2$ correct signatures on M in round 4, and every honest party P_i will output $m_i = M, g_i = 2$ in round 4.

Before proving the second required property, we first show that no two honest parties P_i, P_j send messages (M_i, σ'_i) and (M_j, σ'_j) in round 3 with $M_i \neq M_j$. To see this, note that in round 3, the message M_i (resp., M_j) is either equal to $\perp$ or to the message sent by the dealer to P_i (resp., P_j) in the first round. So if the dealer sent a valid signature on the same message to parties P_i, P_j in the first round, the claim is obviously true. On the other hand, in any other case at least one of P_i, P_j will not send any message at all in round 3 (as at least one of $m_i = \perp$ or $m_j = \perp$ will then hold).

Say a value M^* is *certified* if, in round 4, an honest player holds (M^*, j_1, σ'_1, …, j_ℓ, σ'_ℓ) with $\ell \geq n/2$, distinct $\{j_k\}_{k=1}^{\ell}$, and σ'_k a valid signature of M^* with respect to the public key of party P_{j_k} for $1 \leq k \leq \ell$; in this case, we say the parties $\{P_{j_k}\}_{k=1}^{\ell}$ *certify* M^*. Note that any certified value is certified by at least one honest party. Since any honest parties who sign a message in round 3 sign the *same* message, as argued in the previous paragraph, it follows that at most one value is certified.

Now, say there is an honest party P_i who outputs some message m_i and $g_i = 2$. It follows easily that any honest party P_j who did not output $g_j = 2$ immediately in round 4 will output $g_j = 1$ (and hence we have $g_j \geq 1$). Since, as we have just argued, at most one value can be certified, it follows that all honest parties output m_i.

■

We remark that it is possible to construct authenticated gradecast protocol for $t \geq n/2$ [GKKO]. However, we do not need this result here. On the other hand, it is impossible to achieve gradecast for $t \geq n/3$ without any setup assumption. This follows directly from the impossibility proof for broadcast when $t \geq n/3$ as given in [PSL80, LSP82].

2.2.2 Verifiable Secret Sharing (VSS)

Verifiable Secret Sharing (VSS)[CGMA85] extends the concept of secret sharing [Bla79, Sha79] in the sense that it considers the presence of malicious parties, rather than just honest-but-curious parties.

Definition 3 (Verifiable secret sharing): A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $P^* \in \mathcal{P}$ holds initial input s, is a *VSS protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Validity Each honest party P_i outputs a value s_i at the end of the second phase (the *reconstruction phase*). Furthermore, if the dealer is honest then $s_i = s$.

Secrecy If the dealer is honest at the end of the first phase (the *sharing phase*), then at the end of this phase the joint view of the malicious parties is independent of the dealer's input s.

Reconstruction At the end of the sharing phase the joint view of the honest parties defines a value s' (which can be computed in polynomial time from this view) such that all honest parties will output s' at the end of the reconstruction phase. $\diamondsuit$

2.2.2.1 The Case of $t < n/3$

While VSS protocols tolerating $t < n/3$ malicious parties are known in the literature (cf. [GIKR01, FGG$^+$06]), we present a new VSS protocol below that can be used to optimize the round complexity of protocols constructed in Chapter 3 and Chapter 4.

Lemma 2.2.3 *There exists a VSS protocol tolerating* $t < n/3$ *malicious parties such that the round complexity of its sharing phase is* $(7, 1)$ *and the round complexity of its reconstruction phase is* $(1, 0)$.

Proof The VSS protocol due to Gennaro, et al. [GIKR01] uses 3 rounds of broadcast in the sharing phase. Below, we give a VSS protocol that uses only 1 round of broadcast in the sharing phase. On a high level, our protocol is obtained by applying two modifications to the protocol in [GIKR01]:

- Instead of using a 'random pad' technique to detect inconsistent shares — which invokes one round of broadcast — we use a different method that does not require broadcast at all.

- After the above step, two rounds of broadcast still remain. 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.

Let $\mathbb{F}$ be a finite field with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n]$ can be mapped injectively to $\mathbb{F}$. When the we say the dealer is *disqualified* this means that execution of the protocol halts, and all honest parties output some default value (0, say) in the reconstruction phase.

Sharing Phase

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

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 P^* do the following:

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

Round 6 A party forwards all the messages it 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[2] $X \neq Y$, 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 (in round 7):

- For every message m P_k received in round 5, P_k broadcasts m.
- For every ordered pair (i, j), if in round 6 P_k received messages of the form "$(P_i, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, from $t+1$ different parties, then P_k broadcasts $b'_{k,i} \overset{\text{def}}{=} h_k(i)$ and $c'_{k,i} \overset{\text{def}}{=} g_k(i)$.
- Similarly, for every ordered pair (i, j), if in round 6 P_k received messages of the form "$(P_j, i, j) : X$" and "$(P^*, i, j) : Y$," with $X \neq Y$, from $t + 1$ different parties, then P_k broadcasts $b'_{k,j} \overset{\text{def}}{=} h_k(j)$ and $c'_{k,j} \overset{\text{def}}{=} g_k(j)$.

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

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 *sad* 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 since broadcast is invoked in round 7, all parties agree on whether a party is unhappy or sad.

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 sad parties exceeds t.

   Note that all parties agree whether a dealer is disqualified.

5. A party that is neither unhappy nor sad is said to be *happy.* If the dealer has not been disqualified, then a happy party P_i keeps the polynomials $g_i(x)$ and $h_i(y)$ that it received from the dealer in the first round. An unhappy party P_i takes the polynomials broadcasted by the dealer in

round 7 as $g_i(x)$ and $h_i(y)$. (We do not define what sad parties do, since it is not hard to see that if the dealer is not disqualified then all sad parties are malicious.)

Reconstruction Phase

Round 1 If P_i was happy by the end of the sharing phase, then P_i sends $s_i := g_i(0)$ to all parties; otherwise, P_i sends nothing.

Output determination Party P_i proceeds as follows: if P_j was happy by the end of the sharing phase, let s_j be the value P_j sent to P_i in the previous round; otherwise, set $s_j := g_j(0)$ (where $g_j(x)$ is the polynomial broadcast by the dealer in round 7 of the sharing phase). Let $g(y)$ be the degree-t polynomial resulting from applying Reed-Solomon error-correction [RS60] to $(s_1, s_2, \dots, s_n)$. Output $g(0)$.

We begin our analysis of the protocol with two observations:

(Ob. 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.

(Ob. 2) 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, (Ob. 1) holds. If a party P_i announces a message m, 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 $\geq t+1$ parties forward m to all parties in round 6. Hence (Ob. 2) holds.

We now prove secrecy. For the rest of this paragraph, assume the dealer is honest. We claim that the information the malicious parties have about the dealer's secret s by the end of the sharing phase consists entirely of the polynomials sent to the malicious parties by the dealer in round 1; secrecy then follows since F is a degree-t bivariate polynomial and there are at most t malicious parties. To prove the claim, we first show that no additional information is leaked in rounds 2 through 6. Let P_i and P_j be two honest parties. P_i will not send "$\mathsf{complain}(i,j)$" to the dealer in round 3. Hence, regarding the pair (i,j), parties P_i, P_j and the dealer send "no complaint" to other parties in round 5. Therefore no information about $F(j,i)$ is revealed in rounds 2 through 6. Next, we show that round 7 does not leak any additional information. Suppose P_i is honest, and consider an arbitrary P_k. If P_i sends "$(P_i,i,k) : X$" (resp., "$(P_i,k,i) : X$") to all parties in round 5 for some string X, then the dealer sends "$(P^*,i,k) : X$" (resp., "$(P^*,k,i) : X$") to all parties in round 5 (for the same X). In round 6, for any $Y \neq X$, an honest party receives at most t copies of "$(P_i,i,k) : Y$" or "$(P^*,i,k) : Y$" (resp., "$(P_i,k,i) : Y$" or "$(P^*,k,i) : Y$") from the malicious parties. Hence no honest P_j broadcasts $h_j(i)$ or $g_j(i)$ in round 7. Similarly, the dealer does not broadcast $g_i(x)$ or $h_i(y)$ in round 7.

Next we prove the validity and reconstruction properties. In fact, we will prove something stronger: we show that by the end of the sharing phase, if the dealer has not been disqualified (notice that an honest dealer will never be disqualified), then there exists a degree-t bivariate polynomial $F'(x,y)$ such that P_i holds the polynomials $g_i(x) \overset{\text{def}}{=} F'(x,i)$ and $h_i(y) \overset{\text{def}}{=} F'(i,y)$, and, if the dealer is honest, then $F'(0,0) = s$. Note that if the above holds, every honest party sends $s_i = g_i(0) = F'(0,i)$ to all other parties in the reconstruction phase. Since $n > 3t$ and there are at most t "bad" shares in $\{s_1, s_2, \ldots, s_n\}$, Reed-Solomon error-correction recovers the polynomial $g(y) = F'(0,y)$ and hence all honest parties output $g(0) = F'(0,0)$. Recall that if the dealer is disqualified, then a default value is shared.

In the case of an honest dealer, the dealer will never be disqualified and an honest party will never be unhappy or sad. It follows readily that each P_i holds the polynomials $g_i(x) \overset{\text{def}}{=} F(x,i)$ and $h_i(y) \overset{\text{def}}{=} F(i,y)$. Hence the validity property holds for the case of an honest dealer.

Next we consider the case of a malicious dealer who is not disqualified.

We first show that for any honest P_i, P_j that are not unhappy, $h_j(i) = g_i(j)$. Assume the contrary. Then P_i sends "$\mathsf{complain}(i,j)$" to the dealer in round 3 and announces "$(P_i, i, j) : g_i(j)$" in round 5 (following (Ob. 1)). Depending on the actions of the dealer, P_j announces "$(P_j, i, j) : h_j(i)$" or "$(P_j, i, j) :$ no complaint". No matter what the dealer announces regarding the pair (i,j), at least one of P_i, P_j becomes unhappy. (Note that a non-disqualified dealer has to announce something regarding (i,j) due to (DQ. 1).) This contradicts the assumption that both P_i and P_j are not unhappy.

We now use the following claim [FM97, Lemma 2]:

Claim 2.2.4 *Let $x_1, x_2, \ldots, x_{t+1}$ be distinct elements in $\mathbb{F}$, and $Q_1(y), \ldots, Q_{t+1}(y)$ be polynomials of degree at most t. Then there exists an unique bivariate polynomial*

$F(x, y)$ of degree at most t in both variables such that $F(x_i, y) = Q_i(y)$ for $i = 1, \ldots, t+1$.

Let $\mathcal{H}$ be the set of honest players that are happy. If the dealer is not disqualified, there are at least $2t+1$ happy parties at the end of the sharing phase (due to (DQ. 3)) and at least $t+1$ of them are honest. Hence $|\mathcal{H}| \geq t+1$.

Without loss of generality, assume $P_1, \ldots, P_{t+1} \in \mathcal{H}$. By Claim 2.2.4, there exists a unique bivariate polynomial $F'(x, y)$ of degree at most t in both variables such that $F'(i, y) = h_i(y)$ for $i \in [t+1]$. Before continuing, note that (using Claim 2.2.4 again) there exists an $F''(x, y)$ such that $F''(x, i) = g_i(x)$ for $i \in [t+1]$. But then for any $i, j \in [t+1]$, we have $F'(i, j) = h_i(j) = g_j(i) = F''(i, j)$ and so in fact $F' = F''$. Now consider a happy honest party P_i such that $i \notin [t+1]$. Since $h_i(j) = g_j(i)$ and $g_i(j) = h_j(i)$ for any $j \in [t+1]$, it follows that $F'(x, i) = g_i(x)$ and $F'(i, y) = h_i(y)$.

Let us now prove that $F'(x, i) = g_i(x)$ and $F'(i, y) = h_i(y)$ for all honest parties P_i. There are three cases to consider:

1. If $P_i \in \mathcal{H}$ then it is shown as above.

2. If P_i is unhappy, then due to (DQ. 2) the dealer must have broadcasted $g_i(x)$ and $h_i(y)$. For any $P_j \in \mathcal{H}$, following (Ob. 2) and the definition of unhappy, P_j broadcasts $b'_{j,i} = h_j(i) = F'(j, i)$ and $c'_{j,i} = g_j(i) = F'(i, j)$. Since P_j does not become sad, $g_i(j) = b'_{j,i} = F'(j, i)$ and $h_i(j) = c'_{j,i} = F'(i, j)$. Since $|\mathcal{H}| \geq t+1$ and $g_i(x), h_i(y)$ are polynomials of degree t, it follows that $g_i(x) = F'(x, i)$ and $h_i(y) = F'(i, y)$.

3. The final case is that P_i is sad. We show that no such (honest) party exists. To see this, consider a party $P_j \in \mathcal{H}$. Because P_i and P_j are both not unhappy, we have $h_i(j) = g_j(i) = F'(i, j)$. Since $|\mathcal{H}| \geq t+1$ and $h_i(y)$ is a polynomial of degree at most t, we conclude that $h_i(y) = F'(i, y)$. Similarly, $g_i(x) = F'(x, i)$. We have already shown (in the second case, above) that for all polynomials $g_k(x)$ and $h_k(y)$ broadcast by the dealer in round 7 where P_k is unhappy, we have $g_k(x) = F'(x, k)$ and $h_k(y) = F'(k, y)$. Thus, for all such polynomials g_k and h_k we have $h_i(k) = F'(i, k) = g_k(i)$, $g_i(k) = F'(k, i) = h_k(i)$ and so P_i should not be sad.

The reconstruction property follows.

■

2.2.2.2 The Authenticated Case ($t < n/2$)

We prove an analogue of the previous section for the case of *authenticated* VSS. We remark that VSS protocols tolerating $t < n/2$ malicious parties without assuming a PKI are known in the literature (cf. [Rab94, CDD$^+$99]). However the protocol we give below can be used to optimize the round complexity of protocols constructed in Chapter 3 and Chapter 4. We also remark that it follows from the definition VSS is impossible for $t \geq n/2$ (even for the authenticated case).

Lemma 2.2.5 *There exists an authenticated VSS protocol tolerating $t < n/2$ malicious parties such that the round complexity of its sharing phase is* $(5, 1)$ *(i.e., the sharing phase requires five rounds of interaction and one of the five rounds uses* *broadcast) and the round complexity of its reconstruction phase is* $(1, 0)$.

Proof We assume a finite field $\mathbb{F}$ with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n]$ can be mapped injectively to $\mathbb{F}$. If the dealer is *disqualified* then execution of the protocol halts, and all parties output some default value in the reconstruction phase. Finally, we say an ordered sequence of values $(v_1, \ldots, v_n) \in \mathbb{F}^n$ is *t-consistent* if there exists a polynomial f of degree at most t such that $f(i) = v_i$ for $1 \le i \le n$.

The following protocol is adapted from [CDD+99]. The round complexity of the sharing phase is $(4, 2)$ and the round complexity of the reconstruction phase is $(1, 0)$. Later, we will show how to modify the protocol so that the round complexity of the sharing phase becomes $(5, 1)$.

Sharing Phase

Round 1 The dealer chooses a random bivariate polynomial $F \in \mathbb{F}[x, y]$ of degree at most t in each variable with $F(0, 0) = s$. Let $a_{i,j} = b_{i,j} \overset{\text{def}}{=} F(i, j)$. The dealer sends to party P_i the values $a_{1,i}, \ldots, a_{n,i}$ and $b_{i,1}, \ldots, b_{i,n}$, along with a digital signature on each such value.

Round 2 If P_i receives all values (with valid signatures) from the dealer as specified in round 1, and $(a_{1,i}, a_{2,i}, \ldots, a_{n,i})$ and $(b_{i,1}, b_{i,2}, \ldots, b_{i,n})$ are both t-consistent, then P_i computes signature $\sigma_{j,i}$ on $(j, i, a_{j,i})$, and sends $(a_{j,i}, \sigma_{j,i})$ to party P_j for all $1 \le j \le n$, else P_i sends "Complaint: dealer" to all other parties.

Round 3 If P_i sent "Complaint: dealer" to all other parties in round 2, then P_i broadcasts "Complaint: dealer"; else if P_i received "Complaint: dealer" from

P_j, or P_i did not receive a valid signature $\sigma_{i,j}$ (with respect to the public key of P_j) on $(i, j, a_{i,j})$ from P_j, then P_i broadcasts $b_{i,j}$, as well as the signature of the dealer on $b_{i,j}$ (in this case, we say the value $b_{i,j}$ has been *made public*).

Round 4 If a party P_j broadcast a complaint in round 3, then the dealer broadcasts $\vec{a}_j = (a_{1,j}, \ldots, a_{n,j})$, $\vec{b}_j = (b_{j,1}, \ldots, b_{j,n})$.

For any party P_i that did not broadcast a complaint in round 3,

1. If a party P_j broadcast a complaint in round 3, then then P_i broadcasts $a_{j,i}$ and $b_{i,j}$ with the dealer's signature on these two values.
2. If a party P_j broadcasts a value $b_{j,i}$ (with a valid signature from the dealer on $b_{j,i}$) in round 3, then P_i broadcasts $a_{j,i}$ and the signature of the dealer on $a_{j,i}$.

The dealer is disqualified if any one of the following conditions hold:

1. If there exists a party P_j that broadcast a complaint in round 3, but the dealer did not respond to it in round 4; or the dealer did respond but either $\vec{a}_j$ or $\vec{b}_j$ broadcasted by the dealer is not t-consistent, or $a_{j,j} \neq b_{j,j}$.
2. There exists a pair (i, j) such that each of $a_{i,j}$ and $b_{i,j}$ has been broadcasted by the dealer in round 4 or has been broadcasted by a party along with the dealer's signature on the value, and $a_{i,j} \neq b_{i,j}$.

Note that all parties agree on whether a dealer is disqualified.

Reconstruction Phase

Round 1 For every j such that P_i has a valid signature $\sigma_{i,j}$ (with respect to the public key of P_j) on $(i, j, b_{i,j})$, party P_i sends $(b_{i,j}, \sigma_{i,j})$ to all other parties. Note that for all other j, party P_i has already broadcast $b_{i,j}$ (with the dealer's signature) in round 3 of the sharing phase.

Output determination For each $1 \leq j \leq n$, party P_i verifies the signatures on the values received from P_j in the previous round, and disqualifies P_j if any of the signatures are invalid.

Now, for each P_j which is not yet disqualified, party P_i has values $\vec{b}_i^j \overset{\text{def}}{=} (b_{j,1}, \ldots, b_{j,n})$ (each of these values was either received from P_j in the previous round or was broadcast by P_j in round 3 of the sharing phase). If $\vec{b}_i^j$ is not t-consistent, P_i disqualifies P_j.

Let $\mathcal{H}_i$ be the set of non-disqualified parties, from the perspective of P_i. For each $j \in \mathcal{H}_i$, party P_i interpolates $\vec{b}_i^j$ to obtain a polynomial $f'_j(y)$ of degree at most t (recall that $\vec{b}_i^j$ is t-consistent). Next, P_i interpolates the $\{f'_j(y)\}_{j \in \mathcal{H}_i}$ to obtain bivariate polynomial $F'(x, y)$ of degree at most t in both variables (the proof below will show that this is possible). Output $F'(0, 0)$.

We first prove secrecy of the above protocol. If the dealer is honest, no honest party will send "Complaint: dealer" in step 2. Furthermore, if both P_i, P_j are honest, then P_j will receive a valid signature $\sigma_{j,i}$ (with respect to the public key of P_i) on $(j, i, a_{j,i})$ from P_i. Hence the value of $a_{j,i}$ will not be broadcast in step 3 nor step 4. It follows that the information the adversary has about s by the end of the sharing phase consists entirely of the values sent to the corrupted parties by the dealer in round 1. Secrecy follows since F is a bivariate polynomial of degree at most t in both variables.

We next prove validity. It is easy to see that an honest dealer is never disqualified. Let P_i, P_j be two parties that remain honest throughout the entire execution. The vector $\vec{b}_i^j$ (in the reconstruction phase) matches the values sent by the dealer in round 1 and furthermore $P_j \in \mathcal{H}_i$; thus, P_i recovers $f'_j(y) = F(j, y)$ for every honest P_j. For any malicious $P_k \in \mathcal{H}_i$, the value $b_{k,j}$ that P_i holds was either signed by P_j (in round 2), or broadcast by P_j (in round 4), and so $b_{k,j} = F(k, j)$. Since this holds for at least $t + 1$ honest parties P_j and $\vec{b}_i^k$ is t-consistent (else $k \notin \mathcal{H}_i$), we conclude that P_i recovers $f'_k(y) = F(k, y)$ in this case as well. So interpolating the $\{f'_j(y)\}_{j\in\mathcal{H}_i}$ yields $F(x, y)$ (interpolation can be done since $|\mathcal{H}_i| \geq t + 1$), and the output of P_i is the dealer's secret $F(0, 0)$.

Finally, we prove reconstruction. The case of an honest dealer has been proven above. The case when the dealer is disqualified is obvious, so consider a (corrupted) dealer who is not disqualified.

Let $\mathcal{U}$ be the indices of a set of $t + 1$ parties who are honest at the end of the sharing phase. For an honest party P_i, let $\vec{b^i} = (b_{i,1}, \ldots, b_{i,n})$ denote the values that P_i will "effectively" send to other parties in the reconstruction phase (note that some of these values may, in fact, already have been broadcast). Let $f'_i(y)$ be the result of interpolating $\vec{b^i}$ (this is well-defined since $\vec{b^i}$ is t-consistent for honest P_i), and let $F'(x, y)$ be the result of interpolating $\{f'_i(y)\}_{i\in\mathcal{U}}$. We will show that regardless of the actions of the adversary in the reconstruction phase, each honest party outputs $F'(0, 0)$.

By construction of F', we have $b_{i,k} = F'(i,k)$ for $i \in \mathcal{U}$. We claim that $a_{k,i} = F'(k,i)$ for $i \in \mathcal{U}$. Let $g_i'(x)$ be the result of interpolating $\vec{a}^i = (a_{1,i}, \ldots, a_{n,i})$ (again, this is well-defined since $\vec{a}^i$ is t-consistent for P_i honest). Note that for $j \in \mathcal{U}$ we have $g_i'(j) \stackrel{\text{def}}{=} a_{j,i} = b_{j,i}$ or else the dealer would have been disqualified. So $g_i'(x)$ agrees with $F'(x,i)$ on $t+1$ points and hence these polynomials must be identical, proving the claim.

Applying a similar argument (using the fact that, for P_i honest and $j \in \mathcal{U}$, we have $b_{i,j} = a_{i,j} = F'(i,j)$ or else the dealer is disqualified), we see that for any honest P_i the vector $\vec{b}^i$ interpolates to $f_i'(y) = F'(i,y)$. Furthermore, it is easy to see that if P_i, P_j remain honest then $P_i \in \mathcal{H}_j$. For any corrupted $P_k \in \mathcal{H}_j$ and honest P_i, the value $b_{k,i}$ that P_k sends to P_j in the reconstruction phase was either signed by P_i (in round 2) or broadcast by P_i (in round 4), and so $b_{k,i} = F'(k,i)$. Since this holds for at least $t+1$ honest parties P_i and $\vec{b}_j^k$ is t-consistent (else $k \notin \mathcal{H}_j$), we conclude that P_j recovers $f_k'(y) = F'(k,y)$ in this case as well. So interpolating the $\{f_i'(y)\}_{i \in \mathcal{H}_j}$ yields $F'(x,y)$ (interpolation can be done since $|\mathcal{H}_j| \geq t+1$), and the output of P_j is $F'(0,0)$.

Reducing the number of broadcasts: The sharing phase of the above protocol uses two rounds of broadcast. Basically, they are used in the following manner:

1. In round 3, if certain conditions hold, then P_i broadcasts some message x.
2. In round 4, if P_i broadcasts some particular message y in round 3, then P_j broadcasts some message z.

Call the above protocol Π. We construct an authenticated VSS protocol Π' such that the round complexity of its sharing phase is $(5, 1)$. The first two rounds (of the sharing phase) of Π' are the same as that of Π. The third round of the sharing phase of Π is replaced by two rounds in Π':

Round 3 If P_i is supposed to broadcast x in round 3 of the sharing phase of Π, then P_i sends x to all parties, along with a signature on x.

Round 4 If P_k receives x with a valid signature from P_i, then P_k forwards x (with the signature of P_i on x) to all parties.

The fifth (last) round of Π' proceeds as follow:

Round 5 If P_j receives at least one copy of y (with a valid signature of P_i on y) in round 4 (of Π'), then P_j broadcasts (P_i, y, z). If P_k receives x from P_i in round 3 (with a valid signature of P_i on x), then P_k broadcasts (P_i, x).

To determine if the dealer is disqualified by the end of the sharing phase of Π', we use the same conditions as that of Π but we consider the set of broadcast values as follows:

- If at least $t+1$ parties broadcast (P_i, x) in round 5 of Π', then P_i is considered to have broadcast x in round 3 of Π.

- If, in round 5 of Π', at least $t + 1$ parties broadcast (P_i, x) and P_j broadcasts (P_i, x, z), then P_j is considered to broadcast z in round 4 of Π.

- If, in round 5 of Π', at least $t + 1$ parties broadcast (P_i, y) but party P_j does not broadcast (P_i, y, z) (even if it is supposed to), then P_j is considered to broadcast a default value z in round 4 of Π. (It follows from our discussion below that P_j must be corrupted.)

The reconstruction phase of Π' is the same as that of Π. To claim that Π' is a VSS protocol, we first note that

- In Π, if an honest party P_i is supposed to broadcast x in round 3, then in Π', P_i sends x to all parties (with a signature on x) in round 3, and in round 5 at least $t + 1$ honest parties broadcast (P_i, x).

- In Π, if an honest party P_j is supposed to broadcast z in round 4 after P_i broadcast y in round 3, then in Π', if $t+1$ parties broadcast (P_i, y) in round 5, since at least one of these $t+1$ parties is honest, P_j will receive at least 1 copy of y (with a valid signature of P_i on y) in round 4 and thus it will broadcast (P_i, y, z) in round 5.

Also, note that in round 5 of the sharing phase of Π', some party P_j may broadcast (P_i, y, z) but less than $t+1$ parties broadcast (P_i, y). However, the message (P_i, y, z) does not have any effect on the output determination as the honest parties ignore it. Notice that this can happen only when P_i is corrupted (P_j may or may not be honest). Validity and reconstruction then follow from the proof of Π. To claim secrecy, we simply observe that if the dealer is honest, then no honest party is going to send "Complaint: dealer" in round 2. Thus, no additional information is revealed to the adversary during round 3 to round 5 of the sharing phase of Π'.

We now get an authenticated VSS protocol such that the round complexity of its sharing phase is $(5, 1)$. ■

Chapter 3

Expected Constant-Round Broadcast Protocols

In this chapter, we construct expected constant-round broadcast protocols for both the case $t < n/3$ (where we do not make any setup assumption) and the case $t < n/2$ (the authenticated setting, where a PKI is available). We develop both protocols in parallel so as to highlight the high-level similarities in each. In Section 3.1, we introduce a variant of VSS (see Def. 3) called *moderated VSS*, and we show how to construct protocols for moderated VSS without invoking broadcast, by using VSS and gradecast (see Def. 2) as primitives. In Section 3.1.1, we define *oblivious leader election* and construct the corresponding protocols from moderated VSS. In Section 3.2, we construct broadcast protocols from oblivious leader election.

As mentioned in the Introduction, if a *probabilistic* broadcast protocol is used to emulate multiple invocations of a broadcast channel, then subtle issues related to the parallel and sequential *composition* of the various broadcast sub-protocols arise. We discuss these issues and show how to handle them in Section 3.3.

3.1 Moderated VSS

We introduce a variant of VSS called *moderated VSS*, in which there is a distinguished party (who may be identical to the dealer) called the *moderator*. Roughly speaking, the moderator "simulates" a broadcast channel for the other parties *dur-*

ing the sharing phase. At the end of the sharing phase, parties output a boolean flag indicating whether or not they trust the moderator. If the moderator is honest, all honest parties set this flag to 1. Furthermore, if any honest party sets this flag to 1 then the protocol achieves all the properties of VSS. A formal definition follows.

Definition 4 (Moderated VSS): A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where there is a distinguished dealer $P^* \in \mathcal{P}$ who holds an initial input s and a moderator $P^{**} \in \mathcal{P}$ (who may possibly be the dealer), is a *moderated VSS protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Output Requirement Each honest party P_i outputs a bit f_i at the end of the first phase (called the *sharing phase*), and a value s_i at the end of the second phase (called the *reconstruction phase*).

Completeness If the moderator is honest during the sharing phase, then each honest party P_i outputs $f_i = 1$ at the end of this phase.

Soundness If there exists an honest party P_i who outputs $f_i = 1$ at the end of the sharing phase, then the protocol achieves VSS; specifically: (1) if the dealer is honest then all honest parties output s at the end of the reconstruction phase, and the joint view of all the malicious parties at the end of the sharing phase is independent of s, and (2) the joint view of the honest parties at the end of the sharing phase defines an efficiently-computable value s' such that all honest parties output s' at the end of the reconstruction phase. $\diamondsuit$

We stress that if all honest parties P_i output $f_i = 0$ at the end of the sharing phase, then no guarantees are provided; e.g., honest parties may output different values at the end of the reconstruction phase, or the malicious parties may learn the dealer's secret in the sharing phase.

The main result of this section is the following, which holds for any $t < n/2$:

Theorem 3.1.1 *Assume there exists a VSS protocol* Π *tolerating* t *malicious parties, such that the round complexities of its sharing phase and its reconstruction phase are* (s, s_b) *and* $(r, 0)$*, respectively. Furthermore, assume there exists a gradecast protocol tolerating* t *malicious parties with round complexity* $(g, 0)$*. Then, there exists a moderated VSS protocol* Π' *tolerating* t *malicious parties with the round complexities of the sharing phase and reconstruction phase being* $(s + (2g-1)s_b, 0)$ *and* $(r, 0)$*, respectively.*

Proof We show how to "compile" Π so as to obtain the desired Π'. Essentially, Π' is constructed by replacing each broadcast in Π with two invocations of gradecast: one by the party who is supposed to broadcast the message, and one by the moderator P^{**}. In more detail, Π' is defined as follows: At the beginning of the protocol, all parties set their flag f to 1. The parties then run an execution of Π. When a party P is directed by Π to send message m to P', it simply sends this message. When a party P is directed by Π to broadcast a message m, the parties run the following "broadcast emulation" subroutine:

1. P gradecasts the message m.

2. The moderator P^{**} gradecasts the message it output in the previous step.

3. Let (m_i, g_i) and (m'_i, g'_i) be the outputs of party P_i in steps 1 and 2, respectively. Within the underlying execution of Π, party P_i will use m'_i as the message "broadcast" by P.

4. Furthermore, P_i sets $f_i := 0$ if either (or both) of the following conditions hold: (1) $g'_i \neq 2$, or (2) $m'_i \neq m_i$ and $g_i = 2$.

Party P_i outputs f_i at the end of the sharing phase, and outputs whatever it is directed to output by Π at the end of the reconstruction phase.

It is easy to see that the round complexities of the sharing phase and reconstruction phase of Π' are $(s + (2g-1)s_b, 0)$ and $(r, 0)$, respectively. We now prove that Π' is a moderated VSS protocol tolerating t malicious parties. We first show the completeness property, that is, if the moderator is honest during the sharing phase then no party P_i ever sets $f_i := 0$ when it is honest. To see this, note that if P^{**} is honest then $g'_i = 2$ each time the broadcast emulation subroutine is executed. Furthermore, if P_i outputs some m_i and $g_i = 2$ in step 1 of that subroutine then, by definition of gradecast, P^{**} also outputs m_i in step 1. Hence $m'_i = m_i$ and f_i remains 1.

To show the soundness property of moderated VSS, we first note that the reconstruction phase of Π does not use broadcast. Thus the reconstruction phase of Π' is the same as that of Π. Now consider any execution of the broadcast emulation subroutine during the sharing phase of Π'. We show that if there exists an honest party P_i who holds $f_i = 1$ upon completion of that subroutine, then the functionality of broadcast was achieved (in that execution of the subroutine). It follows that if P_i

holds $f_i = 1$ at the end of the sharing phase, then Π' provided a faithful execution of all broadcasts during the sharing phase of Π and so the functionality of VSS is achieved.

If P_i holds $f_i = 1$, then $g'_i = 2$. (For the remainder of this paragraph, all variables are local to a particular execution of the broadcast emulation subroutine.) Since $g'_i = 2$, the properties of gradecast imply that any honest party P_j holds $m'_j = m'_i$ and so all honest parties agree on the message that was "broadcast." Furthermore, if the "dealer" P (in the broadcast emulation subroutine) is honest then $g_i = 2$ and $m_i = m$. So the fact that $f_i = 1$ means that $m'_i = m_i = m$, and so all honest parties use the message m "broadcast" by P in their underlying execution of Π. ■

By applying the above theorem to the VSS protocol of Lemma 2.2.3 (resp., the authenticated VSS protocol of Lemma 2.2.5) and the gradecast protocol of Lemma 2.2.1 (resp., the authenticated gradecast protocol of Lemma 2.2.2), we obtain:

Corollary 3.1.2 *There exists a moderated VSS protocol tolerating $t < n/3$ malicious parties with round complexity of the sharing phase and reconstruction phase being $(12, 0)$ and $(1, 0)$ respectively.*

Corollary 3.1.3 *There exists an authenticated moderated VSS protocol tolerating $t < n/2$ malicious parties with round complexity of the sharing phase and reconstruction phase being $(12, 0)$ and $(1, 0)$ respectively.*

3.1.1 From Moderated VSS to Oblivious Leader Election

In this section, we construct an oblivious leader election (OLE) protocol based on any moderated VSS protocol. The following definition of oblivious leader election is adapted from [FG03]:

Definition 5 (Oblivious leader election): A two-phase protocol for parties $P_1, \ldots, P_{n-1}, P_n$ is an *oblivious leader election protocol with fairness δ tolerating t malicious parties* if each honest party P_i outputs a value $v_i \in [n]$, and the following condition holds with probability at least δ (over random coins of the honest parties) for any adversary controlling at most t parties:

> There exists a $j \in [n]$ such that (1) each honest party P_i outputs $v_i = j$, and (2) P_j was honest at the end of the first phase.

If the above event happens, then we say *an honest leader was elected.* $\diamondsuit$

If the adversary is static, then we can define an oblivious leader election protocol as a single-phase protocol: with constant probability, all honest parties output a common value j such that P_j is honest. However, if the adversary is adaptive, then the adversary can corrupt P_j as soon as the value of j is known. Thus, in our definition, we only require P_j to be honest up to a certain point (i.e., the end of the first phase).

Intuitively, in our construction of OLE, a random coin $c_i \in [n^4]$ is generated for each party P_i. This is done by having each party P_j select a random value $c_{j,i} \in [n^4]$ and then share this value using moderated VSS with P_i acting as moderator. The $c_{j,i}$ are then reconstructed and c_i is computed as $c_i = \sum_j c_{j,i} \bmod n^4$. An honest party then outputs i minimizing c_i. Since moderated VSS (instead of VSS) is used, each party P_k may have a different view regarding the value of the $\{c_i\}$. However:

- If P_i is honest then (by the properties of moderated VSS) all honest parties reconstruct the same values $c_{j,i}$ (for any j) and hence compute an identical value for c_i.

- If P_i is dishonest but there exists an honest party P_j such that P_j outputs $f_j = 1$ in all invocations of moderated VSS where P_i acts as the moderator, then (by the properties of moderated VSS) all honest parties compute an identical value for c_i.

Relying on the above observations, we devise a way such that all honest parties output the same i (such that P_i is furthermore honest) with constant probability.

Theorem 3.1.4 *Assume there exists a moderated VSS protocol tolerating t malicious parties. Then there exists a OLE protocol with fairness $\delta = \frac{n-t}{n} - \frac{1}{n^2}$ tolerating t malicious parties. Specifically, if $n \geq 3$ and $t < n/2$ then $\delta \geq 1/2$. In addition, the round complexity of phase 1 (resp., phase 2) of the OLE protocol is equal to the round complexity of the sharing phase (resp., the reconstruction phase) of the moderated VSS protocol.*

Proof Each party P_i begins with $\mathsf{trust}_{i,j} = 1$ for $j \in \{1, \ldots, n\}$.

Phase 1 Each party P_i chooses random $c_{i,j} \in [n^4]$ for $1 \leq j \leq n$. The following is executed n^2 times in parallel for each ordered pair (i, j):

All parties execute the sharing phase of a moderated VSS protocol in which P_i acts as the dealer with initial input $c_{i,j}$, and P_j acts as the moderator. If a party P_k outputs $f_k = 0$ in this execution, then P_k sets $\mathsf{trust}_{k,j} := 0$.

Upon completion of the above, let $\mathsf{trust}_k \overset{\text{def}}{=} \{j : \mathsf{trust}_{k,j} = 1\}$.

Phase 2 The reconstruction phase of the moderated VSS protocol is run n^2 times in parallel to reconstruct the secrets previously shared. Let $c^k_{i,j}$ denote P_k's view of the value of $c_{i,j}$. (If a reconstructed value lies outside $[n^4]$, then $c^k_{i,j}$ is assigned some default value in the correct range.) Each party P_k sets $c^k_j := \sum_{i=1}^{n} c^k_{i,j} \bmod n^4$, and outputs $j \in \mathsf{trust}_k$ that minimizes c^k_j.

We prove that the protocol satisfies Definition 5. Following execution of the above, define:

$$\mathsf{trusted} = \left\{ k : \begin{array}{c} \text{there exists a } P_i \text{ that was honest at the end of phase 1} \\ \text{for which } k \in \mathsf{trust}_i \end{array} \right\}.$$

If P_i was honest in phase 1, then $i \in \mathsf{trusted}$. Furthermore, by the properties of moderated VSS, if $k \in \mathsf{trusted}$ then for any honest P_i, P_j and any $1 \leq \ell \leq n$, we have $c^i_{\ell,k} = c^j_{\ell,k}$ and hence $c^i_k = c^j_k$; thus, we may freely omit the superscript in this case. We claim that for $k \in \mathsf{trusted}$, the coin c_k is uniformly distributed in $[n^4]$. Let $c'_k = \sum_{\ell : P_\ell \text{ malicious in phase 1}} c_{\ell,k} \bmod n^4$ (this is the contribution to c_k of the parties that are malicious in phase 1), and let P_i be a party that was honest in phase 1. Since $k \in \mathsf{trusted}$, the properties of VSS hold for all secrets $\{c_{\ell,k}\}_{\ell=1}^{n}$ and thus c'_k is independent of $c_{i,k}$. (If we view moderated VSS as being provided unconditionally, independence holds trivially. When this is instantiated with a protocol for moderated VSS, independence follows from the information-theoretic security of moderated VSS [KLR06].) It follows that c_k is uniformly distributed in $[n^4]$.

By union bound, with probability at least $1 - \frac{1}{n^2}$, all coins $\{c_k : k \in \mathsf{trusted}\}$ are distinct. Conditioned on this event, with probability at least $\frac{n-t}{n}$ the party with the minimum c_j among the set $\mathsf{trusted}$ corresponds to a party which was honest in phase 1. This concludes the proof.

■

Combining Theorem 3.1.4 with Corollaries 3.1.2 and 3.1.3, we obtain:

Corollary 3.1.5 *There exists a OLE protocol with fairness* $2/3$ *tolerating* $t < n/3$ *malicious parties, with the round complexities of phase 1 and phase 2 being* $(12, 0)$ *and* $(1, 0)$ *respectively. (Note that when* $n < 4$ *the result is trivially true.)*

Corollary 3.1.6 *There exists an authenticated OLE protocol with fairness* $1/2$ *tolerating* $t < n/2$ *malicious parties, with the round complexities of phase 1 and phase 2 being* $(12, 0)$ *and* $(1, 0)$ *respectively. (Note that when* $n < 3$ *the result is trivially true.)*

3.2 From Oblivious Leader Election to Broadcast

In this section, we construct broadcast protocols from oblivious leader election. In section 3.2.1, we give the construction for the case $t < n/3$. In section 3.2.2, we give the construction for the authenticated setting ($t < n/2$). We note that our protocols do not provide simultaneous termination (i.e., parties may terminate the protocol at different rounds). The round complexity of a protocol is defined to be the round in which the *last* honest party terminates.

3.2.1 The Case of $t < n/3$

We consider the unauthenticated case (i.e., $t < n/3$) in this section. We construct a broadcast protocol for binary values based on oblivious leader election. This also serves as a warmup for the authenticated case.

Theorem 3.2.1 *If there exists a OLE protocol with fairness δ tolerating $t < n/3$ malicious parties and the round complexities of phase 1 and phase 2 are $(r_1, 0)$ and $(1, 0)$ respectively, then there exists a broadcast protocol tolerating t malicious parties that terminates in (expected) $\max\{6, r_1\} + 1 + 6/\delta$ rounds.*

Proof We first describe a binary broadcast protocol that terminates in (expected) $1+(1+1/\delta)(6+r_1)$ rounds. Later, we will show how to improve its round complexity to (expected) $\max\{6, r_1\} + 1 + 6/\delta$ rounds.

Each party P_i uses two local binary variables: $\texttt{exitBA}_i$ and $\texttt{use\_leader}_i$. Both variables are initially set to `false`.

Step 1 The dealer sends its input bit b to all parties. Let b_i be the bit P_i receives from the dealer.

Step 2 Each P_i sends b_i to all parties. Let $b_{j,i}$ be the bit P_i receives from P_j. (When this step is run at the outset of the protocol, a default value is used if P_i does not receive anything from P_j. In subsequent iterations, if P_i does not receive anything from P_j then P_i leaves $b_{j,i}$ unchanged.)

Step 3 Each party P_i sets $\mathcal{S}_i^b := \{j : b_{j,i} = b\}$ for $b \in \{0,1\}$. If $|\mathcal{S}_i^0| \geq t+1$, then P_i sets $b_i := 0$. If $|\mathcal{S}_i^0| \geq n-t$, then P_i sets $\texttt{exitBA}_i := \texttt{true}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 4 Each party P_i defines $\mathcal{S}_i^b$ as in step 3. If $|\mathcal{S}_i^1| \geq t+1$, then P_i sets $b_i := 1$. If $|\mathcal{S}_i^1| \geq n-t$, then P_i sets $\texttt{exitBA}_i := \texttt{true}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

If $\texttt{exitBA}_i = \texttt{false}$, then P_i sets $\texttt{use\_leader}_i := \texttt{true}$.

Step 5 Each party P_i defines $\mathcal{S}_i^b$ as in step 3. If $|\mathcal{S}_i^0| \geq t+1$, then P_i sets $b_i := 0$. If $|\mathcal{S}_i^0| \geq n-t$, then P_i sets $\texttt{use\_leader}_i := \texttt{false}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 6 Each party P_i defines $\mathcal{S}_i^b$ as in step 3. If $|\mathcal{S}_i^1| \geq t+1$, then P_i sets $b_i := 1$. If $|\mathcal{S}_i^1| \geq n-t$, then P_i sets $\texttt{use\_leader}_i := \texttt{false}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 7 All parties execute the OLE protocol; let ℓ_i be the output of P_i. Each P_i does the following: if $\texttt{use\_leader}_i = \texttt{true}$, then P_i sets $b_i := b_{\ell_i,i}$. If $\texttt{exitBA}_i = \texttt{true}$, then P_i outputs b_i and terminates; otherwise, P_i goes to step 2.

We refer to an execution of step 2 through 7 as an *iteration*. First we claim that if an honest P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 3 or 4 of some iteration, then all honest parties P_j hold $b_j = b_i$ by the end of step 4 of that same iteration. Consider the case when P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 3. (The case when P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 4 is exactly analogous.) This implies that $|S_i^0| \geq n - t$ and hence $|S_j^0| \geq n - 2t \geq t + 1$ and $b_j = 0$. Since this holds for all honest players P_j, it follows that in step 4 we have $|S_j^1| \leq t$ and so b_j remains 1.

Next, we show that if — immediately prior to any given iteration — no honest parties have terminated and there exists a bit b such that $b_i = b$ for all honest P_i, then by the end of step 4 of that iteration all honest parties P_i hold $b_i = b$ and $\texttt{exitBA}_i = \texttt{true}$. This follows easily (by what we have argued in the previous paragraph) once we show that there exists an honest party who sets $\texttt{exitBA}_i := \texttt{true}$ while holding $b_i = b$. Consider the case $b = 0$ (the case $b = 1$ is exactly analogous). In this case $|\mathcal{S}_i^0| \geq n - t$ in step 3 for any honest P_i. Thus, any honest P_i sets $\texttt{exitBA}_i := \texttt{true}$ and holds $b_i = 0$ by the end of this step.

Arguing exactly as in the previous two paragraphs, one can similarly show: (i) If — immediately prior to any given iteration — there exists a bit b such that $b_i = b$ for all honest P_i, then by the end of step 5 of that iteration all honest parties P_i hold $b_i = b$, $\texttt{exitBA}_i = \texttt{true}$, and $\texttt{use\_leader}_i = \texttt{false}$ (and hence all honest parties output b and terminate the protocol in that iteration). (ii) If an honest party P_i sets $\texttt{exitBA}_i := \texttt{true}$ in some iteration, then all honest parties P_j hold $b_j = b_i$ and $\texttt{use\_leader}_j = \texttt{false}$ by the end of step 6 of that iteration. (iii) If an honest party P_i sets $\texttt{use\_leader}_i := \texttt{false}$ in some iteration, then all honest parties P_j

hold $b_j = b_i$ by the end of step 6 of that same iteration.

Next, we show that if an honest party P_i outputs $b_i = b$ (and terminates) in some iteration, then all honest parties output b and terminate by the end of the next iteration. Note that if P_j fails to receive b_i from P_i, then $P_{i,j}$ is unchanged; thus, if P_i terminates with output $b_i = b$, it can be viewed as if P_i keeps on sending $b_i = b$ in the next iteration. (In particular, note that P_i must have sent $b_i = b$ in step 6.) Hence it suffices to show that by the end of the (current) iteration, $b_j = b$ for all honest parties P_j. But this is implied by (ii), above.

Finally, we show that if an honest leader[1] P_ℓ is elected in step 7 of some iteration, then all honest parties P_i terminate by the end of the next iteration. By (i), it is sufficient to show that $b_i = b_{\ell,i} = b_\ell$ at the end of step 7 of the current iteration. Consider two sub-cases: if all honest P_j hold $\texttt{use\_leader}_j = \texttt{true}$ then this is immediate. Otherwise, say honest P_i holds $\texttt{use\_leader}_i = \texttt{false}$. By (iii), $b_\ell = b_i$ at the end of step 6, and hence all honest parties P_j have $b_j = b_i$ by the end of step 7.

If the OLE protocol elects an honest leader with probability δ, then the expected number of iterations until a leader is elected is therefore at most $1/\delta$, and the expected number of iterations is at most $1 + 1/\delta$. Steps 2–6 of each iteration require only one round each, while in step 7 of an iteration the two phases of an OLE protocol are run. Thus the above protocol terminates in (expected) $1 + (1 + 1/\delta)(6 + r_1)$ rounds.

[1]This implies that P_ℓ was uncorrupted in step 6 of the iteration in question.

We can, however, do better. The key observation is that the first phase of the OLE protocol can be carried out in advance of step 7, and in particular can be carried out in parallel with steps 1–6. (A similar observation was made in [Fel88].) Even more, we can run multiple invocations of the first phase of the OLE protocol and "save them" until needed. Applying these ideas, we obtain the following broadcast protocol:

1. Run $\ell \overset{\text{def}}{=} \lceil r_1/6 \rceil$ executions of the first phase of the OLE protocol. These are scheduled so that the final 6 rounds coincide with steps 1–6 of the first iteration.

2. For the remainder of the protocol, continually run ℓ parallel executions of the first phase of the OLE protocol in parallel with the "main" protocol. These parallel executions will terminate every r_1 rounds, just as the ℓ previous executions get "used up."

We now have a broadcast protocol that terminates in (expected) $\max\{6, r_1\}$ $+1 + 6/\delta$ rounds. ■

Ben-Or and El-Yaniv [BE03] showed how to transform a binary broadcast protocol into a (multi-valued) broadcast protocol. While their solution is simple, it incurs overhead on the round complexity. On the other head, parallel composition (see Section 3.3) can be used to transform a binary broadcast protocol into a (multi-valued) broadcast protocol without using any additional rounds (though the solution is more complicated). In combination with the above theorem and Corollary 3.1.5, we now have:

Corollary 3.2.2 *There exists a broadcast protocol tolerating $t < n/3$ adaptive corruptions that terminates in (expected) 22 rounds.*

3.2.2 The Authenticated Case ($t < n/2$)

We consider the authenticated case ($t < n/2$) in this section.

Theorem 3.2.3 *If there exists an authenticated OLE protocol with fairness δ tolerating $t < n/2$ malicious parties and the round complexities of phase 1 and phase 2 are $(r_1, 0)$ and $(1, 0)$ respectively, then there exists a broadcast protocol tolerating t malicious parties that terminates in (expected) $\max\{7, r_1\} + 8 + 7/\delta$ rounds.*

Proof We start by constructing a broadcast protocol that terminates in (expected) $1 + (2 + 1/\delta)(7 + r_1)$ rounds. Later, we show how to improve its round complexity to (expected) $\max\{7, r_1\} + 8 + 7/\delta$ rounds.

Let V be the domain of possible input values, let $\phi \in V$ be some default value and let $\perp$ be some special value that is *not* in V. Each P_i begins with an internal variable $\mathsf{Iteration\_left}_i$ set to ∞. To avoid having to say this every time, we make the implicit requirement that if $\mathsf{Iteration\_left}_i \neq \infty$ then the value of v_i is "locked" and remains unchanged (i.e., even if the protocol description below says to change it).

We say that P_i has a *(valid) certificate for* v if $v \in V$ and there exist $k > n/2$ distinct indices $j_1, \ldots, j_k$ such that P_i holds $\sigma_{j_1,i}, \ldots, \sigma_{j_k,i}$ which are valid signatures on v with respect to the public keys of $P_{j_1}, \ldots, P_{j_k}$. In this case, we will also call $(v, j_1, \ldots, j_k, \sigma_{j_1,i}, \ldots, \sigma_{j_k,i})$ a *certificate for* v.

Step 1 The dealer sends its input value to all parties. Let v_i be the value P_i receives from the dealer.

Step 2 Party P_i computes a signature σ_i of v_i and sends (v_i, σ_i) to all parties.

Step 3 Let $(v_{j,i}, \sigma_{j,i})$ be the message received by party P_i from P_j. If these messages yield a certificate for v_i, then P_i sends a certificate for v_i to all parties. Otherwise P_i sends nothing and sets $v_i := \perp$.

Step 4 If in the previous round P_i received a valid certificate for some $v^* \neq v_i$, then P_i sets $v_i := \perp$.

If $v_i \neq \perp$, then P_i computes a signature[2] σ'_i of v_i and sends (v_i, σ'_i) to all parties.

Step 5 Let $(v_{j,i}, \sigma'_{j,i})$ be the message received by party P_i from P_j (if any) in the previous round. If these messages yield a certificate for v_i, then P_i sends a certificate for v_i to all parties and sets $\mathsf{Iteration\_left}_i := 1$; otherwise P_i sends nothing and sets $v_i := \perp$.

Step 6 If in the previous round P_i received a valid certificate on some value v^*, then P_i sends a certificate on v^* to all parties and sets $v_i := v^*$. Otherwise, P_i sends nothing and sets $v_i := \perp$.

Step 7 If in the previous round P_i received a valid certificate on some value v^*, then P_i sends v^* to all parties; otherwise, P_i sends $\perp$ to all parties. Let $v^*_{j,i}$ be the value P_i received from P_j in this round.

[2]The current round number is also signed to distinguish this signature from others.

Step 8 All parties execute the OLE protocol; let ℓ_i be the output of P_i. If $v_i = \perp$ and $v^*_{\ell_i,i} \neq \perp$, then P_i sets $v_i := v^*_{\ell_i,i}$. If $v_i = v^*_{\ell_i,i} = \perp$, then P_i sets $v_i := \phi$. If $\mathsf{Iteration\_left}_i = 0$, then P_i outputs v_i and terminates the protocol. If $\mathsf{Iteration\_left}_i = 1$, then P_i sets $\mathsf{Iteration\_left}_i := 0$ and goes to step 2. If $\mathsf{Iteration\_left}_i = \infty$, then P_i goes to step 2.

We refer to an execution of steps 2 through 8 as an *iteration*. We first claim that if — immediately prior to any given iteration — there exists a value v such that $v_i = v$ for all honest P_i and no honest parties have yet terminated, then all honest parties will terminate and output v by the end of the following iteration. (This in particular proves the correctness of the protocol for the case of an honest dealer.) To see this, note that in this case all honest parties P_i compute a signature σ_i of v and send (v, σ_i) to all parties in step 2. In step 3, the messages received by an honest party yield a certificate for v, thus all honest parties send a certificate for v to all parties. In step 4, an honest P_i receives a valid certificate for v, it computes a signature σ'_i of v and sends (v, σ_i) to all parties. In step 5, the messages received by an honest party yield a certificate for v, and an honest P_i sets $\mathsf{Iteration\_left}_i := 1$. Thus all honest parties will terminate and output v by the end of the next iteration.

Now consider the first iteration in which an honest party P_i sets $\mathsf{Iteration\_left}_i$ $:= 1$ (in step 5). We claim that, by the end of that iteration, $v_j = v_i$ for all honest P_j and no honest parties will have yet terminated. The claim regarding termination is immediate since honest parties do not terminate until the iteration following the one in which they set $\mathsf{Iteration\_left} := 1$ (and we are considering the first such iteration). As for the first property, we make an observation that at any given step there is at most one value $v \in V$ for which some honest party has a certificate on v (otherwise, some honest party must have signed two different values; but this cannot occur). It then follows that any honest party P_j setting $\mathsf{Iteration\_left}_j := 1$ in this iteration must also hold $v_j = v_i$. For an honest party P_j holding $\mathsf{Iteration\_left}_j = \infty$ after step 5, since P_i sends a certificate for v_i to all parties (in step 5) the earlier observation again implies that P_j sets $v_j := v_i$ in step 6. Since $v_j = v_i \neq \perp$ (else P_i would not have set $\mathsf{Iteration\_left}_i := 1$), P_j will not change the value of v_j in step 8. This establishes the claim, and implies that if any honest party terminates then all honest parties terminate with the same output.

To complete the proof that the above protocol is a broadcast protocol, we show that if an honest leader P_ℓ is elected in some iteration then all honest parties will hold the same value v by the end of that iteration. By what we have argued in the previous paragraph, we only need to consider the case where $\mathsf{Iteration\_left}_i = \infty$ for all honest P_i in step 8. If $v_i = \perp$ for all honest P_i by the end of step 6, the claim is immediate since every honest P_i will change their value of v_i to the honest leader's value (or ϕ, as appropriate) in step 8. Otherwise, $v_i \neq \perp$ by the end of step 6 for some honest party P_i. By the observation mentioned earlier, every honest party P_j holds $v_j \in \{v_i, \perp\}$ by the end of step 6. Furthermore, P_ℓ receives a valid certificate on v_i from P_i in step 6 and so (again using the earlier observation) sends $v_\ell^* = v_i$ to all parties in step 7. Hence every honest party P_j holds $v_j = v_i$ by the end of that iteration.

If the OLE protocol has fairness δ, then the expected number of iterations until an honest leader is elected is therefore at most $1/\delta$, and the expected number of iterations is $2 + 1/\delta$. Steps 2-7 of each iteration require one round each, while in step 8 of an iteration the two phases of an OLE protocol are run. Thus the above protocol terminates in (expected) $1 + (2 + 1/\delta)(7 + r_1)$ rounds.

As in the proof of Theorem 3.2.1, we can do better by executing the first phase of the OLE protocol in advance of step 8, and then running multiple invocations of the first phase of the OLE protocol and "saving them" until needed. We then obtain the following broadcast protocol:

1. Run $\ell \overset{\text{def}}{=} \lceil r_1/7 \rceil$ executions of the first phase of the OLE protocol. These are scheduled so that the final 7 rounds coincide with steps 1–7 of the first iteration.

2. For the remainder of the protocol, continually run ℓ parallel executions of the first phase of the OLE protocol in parallel with the "main" protocol. These parallel executions will terminate every r_1 rounds, just as the ℓ previous executions get "used up."

Thus, we now have a broadcast protocol that can terminate in (expected) $\max\{7, r_1\} + 8 + 7/\delta$ rounds. ■

In combination with the above Theorem and Corollary 3.1.6, we now have:

Corollary 3.2.4 *There exists an authenticated broadcast protocol tolerating $t < n/2$ adaptive corruptions that terminates in (expected) 34 rounds.*

3.3 Parallel and Sequential Composition

Suppose we have a protocol Π that is designed under the assumption that a broadcast channel is available; when run in a point-to-point network, one approach is to replace each invocation of the broadcast channel in Π with an invocation of an expected constant-round (authenticated) broadcast protocol bc; i.e., to set $\Pi' = \Pi^{\mathsf{bc}}$. However, there are two subtle problems with this approach that must be dealt with:

Parallel composition. In protocol Π, all n parties may access the broadcast channel in the same round; this results in n parallel executions of bc in protocol Π^{bc}. Although the expected round complexity of *each* execution of bc is constant, the expected number of rounds for *all* n executions of bc to terminate may no longer be constant.

A general technique for handling this issue is proposed by [BE03]; their solution is somewhat complicated. In our case, however, we may rely on an idea of Fitzi and Garay [FG03] that applies to OLE-based protocols such as ours. The main idea is that when multiple broadcast sub-routines are run in parallel, only a *single* leader election (per iteration) is required for all these sub-routines. Using this approach, the expected round complexity for n parallel executions will be identical to the expected round complexity of a single execution.

Sequential composition. A second issue is that protocol bc does not provide simultaneous termination. (As noted in [LLR02], this is inherent for any expected constant-round broadcast protocol.) This may cause problems for sequential executions of bc within Π^{bc}, since subsequent executions of bc may not have all honest parties starting at the same round. As an example of what can go wrong, assume some protocol Π (that relies on a broadcast channel) requires some party P_i to broadcast values in rounds 1 and 2. Let $\mathsf{bc}_1, \mathsf{bc}_2$ denote the corresponding invocations of broadcast within the composed protocol Π^{bc} (which runs in a point-to-point network). Then, because honest parties in bc_1 do not terminate in the same round, honest parties may begin execution of bc_2 in different rounds. But security of bc_2 is no longer guaranteed in this case!

At a high level, we can fix this by making sure that bc_2 remains secure as long as all honest parties begin execution of bc_2 within a certain number of rounds. Specifically, if honest parties are guaranteed to terminate bc_1 within g rounds of each other, then bc_2 should remain secure as long as all honest parties start within g rounds. We now show how to achieve this for an arbitrary number of sequential executions of bc, without blowing up the round complexity too much.

Let $\mathsf{rc}(\Pi)$ denote the (expected) round complexity of a protocol Π.[3] The *staggering gap* of Π is defined as follows:

Definition 6 A protocol Π has *staggering gap* $g = \mathsf{gap}(\Pi)$ if any honest parties P_i, P_j are guaranteed to terminate Π within g rounds of each other.

We begin with the following result by Lindell, et al. [LLR02, Lemma 3.1]:

Lemma 3.3.1 *Let* bc *be a protocol for (authenticated) broadcast with staggering gap* g *and expected round complexity* r*. Then for any constant* $c \geq 0$ *there exists a*

[3]Recall that the round complexity of a run of a protocol having a non-zero staggering gap is the round in which the *last* honest party terminates.

protocol $\mathsf{Expand}'_c(\mathsf{bc})$ *which achieves (authenticated) broadcast as long as all honest parties begin execution of* $\mathsf{Expand}'_c(\mathsf{bc})$ *within* c *rounds of each other.* $\mathsf{Expand}'_c(\mathsf{bc})$ *has expected round complexity* $(2c+1)\cdot r$ *and staggering gap* $c+g\cdot(2c+1)$*.*

Next, we prove the following lemma:

Lemma 3.3.2 *Let* bc *be a protocol for (authenticated) broadcast. Then for any constant* $c \geq 0$ *there exists a protocol* $\mathsf{Expand}_c(\mathsf{bc})$ *which achieves the same security as* bc *as long as all honest parties begin execution of* $\mathsf{Expand}_c(\mathsf{bc})$ *within* c *rounds of each other. Furthermore,*

$$\mathsf{rc}\left(\mathsf{Expand}_c(\mathsf{bc})\right) = (2c+1)\cdot \mathsf{rc}(\mathsf{bc}) + 1,$$

and the staggering gap of $\mathsf{Expand}_c(\mathsf{bc})$ *is 1 (as long as all honest parties begin execution within* c *rounds of each other).*

This result holds unconditionally for the case of $t < n/3$ *malicious parties, and under the assumption of a PKI and secure digital signatures for* $t < n/2$*.* [4]

Proof We describe protocol $\mathsf{Expand}_c(\mathsf{bc})$. Each party P_i executes $\mathsf{Expand}'_c(\mathsf{bc})$. When P_i terminates execution of $\mathsf{Expand}'_c(\mathsf{bc})$ with output v_i, it sends "exit, v_i" to all parties (along with a signature of this message in the authenticated case). Furthermore, at every round (including during execution of $\mathsf{Expand}'_c(\mathsf{bc})$), P_i does the following:

Unconditional case: If there exists a value v such that P_i has received "exit, v" from $t+1$ distinct parties, then P_i sends "exit, v" to all parties.

[4]We note that our solution for the unconditional case is very similar to the solution by Ben-Or and El-Yaniv [BE03], we include it for the sake of completeness.

If P_i has received "exit, v" from $2t + 1$ distinct parties (possibly including itself), then it terminates $\mathsf{Expand}_c(\mathsf{bc})$ with output v (even if its execution of $\mathsf{Expand}'_c(\mathsf{bc})$ has not yet completed).

Authenticated case: If there exists a value v such that P_i has received valid signatures from $t+1$ distinct parties (possibly including itself) on "exit, v," then P_i forwards "exit, v" along with these $t + 1$ signatures to all parties and terminates $\mathsf{Expand}_c(\mathsf{bc})$ with output v (even if its execution of $\mathsf{Expand}'_c(\mathsf{bc})$ has not yet completed).

We first show that $\mathsf{Expand}_c(\mathsf{bc})$ has staggering gap 1. Let round k be the first round in which some honest party P_i terminates $\mathsf{Expand}_c(\mathsf{bc})$ with output v. Then:

Unconditional case: When P_i terminates $\mathsf{Expand}_c(\mathsf{bc})$ in round k, it has received $2t+1$ copies of "exit, v," at least $t+1$ of which are from honest parties. Hence all honest parties have received at least $t + 1$ copies of "exit, v" by round k and have sent "exit, v" by round $k + 1$. Since there are at least $2t + 1$ honest parties, it follows that all honest parties receive $2t + 1$ copies of "exit, v" (and hence terminate $\mathsf{Expand}_c(\mathsf{bc})$) by round $k + 1$.

Authenticated case: When P_i terminates $\mathsf{Expand}_c(\mathsf{bc})$ in round k, it has received $t + 1$ valid signatures on "exit, v" which it forwards to all parties. Hence all honest parties receive the forwarded signatures in round $k + 1$ and terminate by that round.

To conclude the proof, we show that $\mathsf{Expand}_c(\mathsf{bc})$ achieves (authenticated) broadcast. It is easy to see that all honest parties output the same value. Furthermore, note that no honest party terminates $\mathsf{Expand}_c(\Pi)$ until *some* honest party has terminated $\mathsf{Expand}'_c(\Pi)$. Since $\mathsf{Expand}'_c(\mathsf{bc})$ achieves (authenticated) broadcast, when the dealer is honest any honest players who run $\mathsf{Expand}'_c(\mathsf{bc})$ to completion will output the dealer's message (in $\mathsf{Expand}'_c(\mathsf{bc})$). It follows that when the dealer is honest all honest players will output the dealer's message (in $\mathsf{Expand}_c(\mathsf{bc})$).

We remark that this proof does not apply to *arbitrary* protocols (rather, we have explicitly stated the lemma only for protocols achieving broadcast) since we use the fact that all honest parties should terminate with *identical* outputs. ■

We claim that combining the above two lemmas give a solution. To see this, suppose we want to sequentially compose protocols $\mathsf{bc}_1, \ldots, \mathsf{bc}_\ell$, each having staggering gap g. We simply run ℓ sequential executions of $\mathsf{bc}'_i = \mathsf{Expand}_1(\mathsf{bc}_i)$ instead. Each bc'_i has staggering gap 1, meaning that honest parties terminate within 1 round of each other. From the above lemma, each bc'_{i+1} is secure as long as honest parties begin execution within 1 round of each other, so things are ok.

Applying this technique to compile a protocol Π (which uses a broadcast channel in each of its $\mathsf{rc}(\Pi)$ rounds) to a protocol Π' (running in a point-to-point network), we obtain

$$\mathsf{rc}(\Pi') = \mathsf{rc}(\Pi) \cdot (3 \cdot \mathsf{rc}(\mathsf{bc}) + 1)\,.$$

In particular, if Π is a constant-round protocol and bc is an expected constant-round broadcast protocol, then Π' runs in an expected constant number of rounds.

Chapter 4

Round-Efficient Secure Multiparty Computation

In this chapter, we construct constant-round secure multiparty computation protocols that use only one round of broadcast. In Section 4.2, we consider the case $t < n/3$ under the plain model (i.e., no setup assumption). In Section 4.3, we consider the authenticated case $t < n/2$. Throughout the chapter, we assume the existence of one-way functions.

4.1 Techniques and Overview

We give a high-level overview of the main techniques we use. 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 parties,[1] and $c = ab$. Following the results in [BMR90, Bea91a, DI05], assuming the existence of one-way functions, if in a "setup phase," the parties share their inputs along with sufficiently-many multiplication triples, then the parties can carry out secure multiparty computation in a *constant* number of rounds *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

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

Section 3.1. Recall that in such protocols, there is a distinguished party P_m known as the *moderator*. Given a protocol Π, designed under the assumption of a broadcast channel, the moderated version of Π is a protocol Π' that does not require any invocation of broadcast and has the following properties (roughly speaking):

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

In Section 3.1, we have shown how to compile a VSS protocol into its moderated version, while increasing the overall round complexity by at most a constant multiplicative factor. (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.) It is not hard to verify that the proof extends for more general classes of functionalities.

Let Π_i denote some constant-round 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., [BGW88, Rab94, Bea89, GRR98, CDD$^+$99, DI05]. We compile Π_i into a moderated protocol Π'_i where P_i 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. Recall that P_i is the moderator in Π'_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 Π'_{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 Π'_{i^*} achieves the functionality of Π_{i^*}. Since Π_{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 Π_i to Π'_i (following Section 3.1), each round of broadcast in Π_i is replaced by six rounds in Π'_i for the case $t < n/3$, and eight rounds for the authenticated case $t < n/2$. Proceeding directly thus yields secure MPC protocols with relatively high round complexity: after all, existing constructions of protocols Π_i achieving the needed functionality do not attempt to minimize the number of rounds of broadcast. In the subsequent sections, 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.

4.2 The Case of $t < n/3$

Roadmap Using the VSS protocol given in Section 2.2.2.1, we can generate two random values a and b: (i) each party P_i picks two random values a_i and b_i, shares it using the sharing phase of the VSS protocol (ii) the parties reconstruct the values a_1, $\ldots$, a_n, b_1, $\ldots$, b_n and compute $a \overset{\text{def}}{=} \sum_i a_i$ and $b \overset{\text{def}}{=} \sum_i b_i$. However, the properties guaranteed by a VSS protocol are not strong enough to share random multiplication triples. In Section 4.2.1, we extend the definitions of VSS to what we call VSS with 2(3)-level sharing and construct protocols that satisfy the extended definitions. In Section 4.2.2, using VSS with 2(3)-level sharing, we construct a protocol that shares random multiplication triples. Finally, in Section 4.2.3, we implement the setup phase using one round of broadcast and then we show a constant-round secure multiparty computation protocol without additional invocation of broadcast.

4.2.1 Generalized Secret Sharing and VSS

Throughout, we assume a finite field $\mathbb{F}$ of characteristic 2 which contains all values s we are interested in, $[n]$ (interpreted appropriately) as a subset. We start by defining different levels of secret sharing.

Definition 7 (1-level sharing): We say a value s has been *1-level shared* if there exists a polynomial $F_s(x)$ with degree at most t such that (1) $F_s(0) = s$ and (2) party P_i holds the share $s_i \overset{\text{def}}{=} F_s(i)$. In this case, we say that $F_s(x)$ *1-level shares* s, or alternatively, s is *1-level shared* by $F_s(x)$.

Note that if s_a, s_b are 1-level shared by the polynomials $F_{s_a}(x)$ and $F_{s_b}(x)$ respectively, then for any publicly-known $\alpha, \beta \in \mathbb{F}$ the value $\alpha s + \beta s'$ is 1-level shared as well. To see this, observe that $\alpha F_{s_a}(x) + \beta F_{s_b}(x)$ is a polynomial with degree at most t, $\alpha F_{s_a}(0) + \beta F_{s_b}(0) = \alpha s_a + \beta s_b$ and each P_i can compute the value $\alpha F_{s_a}(i) + \beta F_{s_b}(i)$.

Definition 8 (2-level sharing): We say a value s has been *2-level shared* if

(1) There exists a polynomial $F_s(x)$ of degree at most t that 1-level shares s.

(2) For $i \in [n]$, the value $s_i \overset{\text{def}}{=} F_s(i)$ has been 1-level shared by the polynomial $F_{s_i}(x)$.

(3) Each honest party P_i knows the polynomial $F_{s_i}(x)$.

$\diamondsuit$

Note that if s has been 2-level shared, then s has been 1-level shared as well.

Definition 9 (3-level sharing): We say a value s has been *3-level shared* if

(1) There exists a polynomial $F_s(x)$ of degree at most t that 1-level shares s.

(2) For $i \in [n]$, the value $s_i \overset{\text{def}}{=} F_s(i)$ has been 2-level shared; in particular, this implies s_i is 1-level shared by a polynomial $F_{s_i}(x)$.

(3) Each party P_i knows the polynomial $F_{s_i}(x)$.

$\diamondsuit$

Note that if s has been 3-level shared, then s has been 2-level shared as well. Also note that if both s_a, s_b have been 3-level shared, then for any publicly-known $\alpha, \beta \in \mathbb{F}$ the value $\alpha s + \beta s'$ is 3-level shared as well. We now generalize the definition of VSS (cf. Definition 3) as follows:

Definition 10 (Generalized verifiable secret sharing): A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $P^* \in \mathcal{P}$ holds 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:

Validity By the end of the first phase, some value s' is 2-level (resp., 3-level) shared. Moreover, if the dealer is honest then $s' = s$.

Secrecy If the dealer is honest during the first phase (the *sharing phase*), then at the end of this phase the joint view of the malicious parties is independent of the dealer's input s.

Reconstruction All honest parties will output s' at the end of the reconstruction phase. $\diamondsuit$

Next, we construct VSS protocols with 2(3)-level sharing. If s is 1-level shared, and $t < n/3$, then the parties can reconstruct s in one round: each party P_i sends $F_s(i)$ to all other parties, and then each party obtains the polynomial $F_s(x)$ (and hence $s \overset{\text{def}}{=} F_s(0)$) by applying Reed-Solomon error-correction [RS60] to the received values. Therefore, we focus on constructing the sharing phase of the VSS protocols. We first construct a VSS protocol with 2-level sharing; then, based on this protocol, we construct a VSS protocol with 3-level sharing.

Lemma 4.2.1 *There exists a VSS protocol with 2-level sharing tolerating* $t < n/3$ *malicious parties such that the round complexity of its sharing phase is* $(7, 1)$.

Proof We observe that the VSS protocol given in Lemma 2.2.3 is in fact a VSS protocol with 2-level sharing, as long as a default value s' is 2-level shared when the (corrupted) dealer is disqualified. Secrecy follows directly from the definition of VSS. We now argue validity. Following the proof in Lemma 2.2.3, by the end of the sharing phase, as long as the dealer has not been disqualified, there exists a bivariate degree-t polynomial $F'(x, y)$, such that each P_i holds the polynomials $g_i(x) \overset{\text{def}}{=} F'(x, i)$, $h_i(y) \overset{\text{def}}{=} F'(i, y)$. Hence the value $F'(0, 0)$ is 2-level shared since:

- Each P_i knows $F'(0, i) = g_i(0)$, so $F'(0, 0)$ is 1-level shared by the polynomial $F'(0, y)$
- For $i \in [n]$, the value $F'(0, i)$ has been 1-level shared by the polynomial $g_i(x) = F'(x, i)$, as each P_j knows the value $g_i(j) = F'(j, i) = h_j(i)$
- P_i knows the polynomial $g_i(x)$

Moreover, if the dealer is honest, then $F'(0, 0) = s$. Thus validity holds. ■

We now move on to construct a VSS protocol with 3-level sharing:

Lemma 4.2.2 *There exists a VSS protocol with 3-level sharing tolerating* $t < n/3$ *malicious parties such that the round complexity of its sharing phase is* $(8, 1)$.

Proof The idea of our construction is as follows: using the sharing phase of a VSS protocol with 2-level sharing, the dealer shares the secret s, as well as n values g_1, $\ldots$, g_n such that $g_i = f_s(i)$ for all $i \in [n]$, where $f_s(x)$ is the polynomial that 1-level shares s. To enable P_i to reconstruct the polynomial $F_{g_i}(x)$ that 1-level shares g_i, each party P_j sends its share $F_{g_i}(j)$ to P_i afterwards, then P_i applies Reed-Solomon error correction to reconstruct the polynomial. If the dealer is honest, then s would have been 3-level shared. However, a corrupted dealer can cheat by sharing a value $g_i \neq f_s(i)$. To prevent this from happening, the parties reconstruct the values $\{g_i - f_s(i)\}$ and check if all of them are equal to 0. Note that this is possible since both g_i and $f_s(i)$ have been 1-level shared.

We give the protocol specification below. When we say the dealer is *disqualified* we mean that execution of the protocol halts, and a default value s' is 3-level shared (via some default polynomials). We use the VSS protocol Π given in Lemma 4.2.1 as a building block for the following reason: the dealer gets to choose the polynomial that shares the secret prior to any message exchange. This enables us to run the $n+1$ invocations of the sharing phase in parallel.

Step 1 The dealer shares s using the sharing phase of the VSS protocol Π. Let $f_s(y)$ be the polynomial that 1-level shares s. The dealer shares $g_1 \overset{\text{def}}{=} f_s(1), \ldots, g_n \overset{\text{def}}{=} f_s(n)$ using n additional invocations of the sharing phase of Π.

Step 2 If the dealer is disqualified in any invocation of the sharing phase in the previous step, then it is disqualified. Otherwise, s, g_1, $\ldots$, g_n have been 2-level shared. Let $f_{s_i}(x)$ be the polynomial that 1-level shares $f_s(i)$, and let

$g_i(x)$ be the polynomial that 1-level shares g_i. For $1 \leq j \leq n$, each P_i sends $d_{i,j} \overset{\text{def}}{=} f_{s_j}(i) - g_j(i)$ to all parties; P_i also sends $g_j(i)$ to P_j.

Output determination Let $d_1(x)$, …, $d_n(x)$ be the degree-t polynomials resulting from applying Reed-Solomon error-correction to $\{d_{1,1}, \ldots, d_{n,1}\}$, …, $\{d_{1,n}$, …, $d_{n,n}\}$. If there exists a $j \in [n]$ such that $d_j(0) \neq 0$, then the dealer is disqualified. We note that all parties have the same view on $\{d_j(0)\}$ since both $f_{s_i}(0)$ and $g_i(0)$ have been 1-level shared. Otherwise, P_i computes the polynomial $g_i(x)$ by applying Reed-Solomon error-correction to the shares it received.

We first observe that the round complexity of the sharing phase of the above protocol is $(8, 1)$: The round complexity of step 1 is $(7, 1)$ and step 2 takes one round.

We first prove secrecy. For the rest of this paragraph, assume the dealer is honest. Following the secrecy property of the underlying VSS protocol with 2-level sharing, if P_i is honest, then the adversary does not learn any new information about $f_s(i)$ when the dealer shares $g_i \overset{\text{def}}{=} f_s(i)$ using VSS. On the other hand, if P_i is corrupted, then the adversary already knows the value of $f_s(i)$ and thus it does not learn any additional information when the dealer shares g_i. Hence the view of the adversary remains independent of s after step 1. If the dealer is honest, then $d_i(0) = 0$. Learning the polynomial $d_i(x)$ in step 2 does not give the adversary any additional information about g_i.

Next, we prove validity. It is easy to see that an honest dealer will not be disqualified. Thus the value s is 3-level shared. Now consider a dealer who is not disqualified (the case of a disqualified malicious dealer is trivial). In this case, $d_1(0) = \ldots = d_n(0) = 0$. Hence $f_{s_i}(0) = g_i$ for all i. Thus $f_s(0)$ has been 3-level shared since:

- $f_s(0)$ is 1-level shared by the polynomial $f_s(x)$.
- For $i \in [n]$, $g_i = f_s(i)$ has been 2-level shared.
- Let $g_i(x)$ be the polynomial 1-level sharing g_i. P_i knows the polynomial $g_i(x)$ as P_i reconstructs it in the output determination step.

■

4.2.2 Generating Random Multiplication Triples

In this section, we construct a $(17, 3)$-round protocol for generating random multiplication triples. (We describe the protocol for generating one such triple, but it can be parallelized to generate as many as needed.) Specifically, at the end of the protocol there will exist three values a, b, c such that:

1. a, b, and c are 1-level shared.
2. $c = ab$.
3. Given the view of the adversary, a and b are uniformly distributed in $\mathbb{F}$.

We then use a technique from [FGG$^+$06] to reduce the round complexity of the protocol to $(11, 3)$.

We start by recalling the following technical lemma concerning multiplication of shares [GRR98]:

Lemma 4.2.3 *Let $A(x), B(x)$ be two polynomials of degree at most t, 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 $\beta_1, \ldots, \beta_{2t+1} \in \mathbb{F}$.*

On a high level, our protocol proceeds as follows:

1. Two random values a and b are shared among the parties such that the view of the adversary is independent of a and b.

2. Let $A(x)$ and $B(x)$ be the polynomials that 1-level shares a and b respectively. P_i shares the product of $A(i)$ and $B(i)$ and proves that the sharing is done correctly. The method we use for proving is adapted from [BGW88]. The high level idea is as follows: $A(x)B(x)$ is a polynomial of degree $2t$. Suppose P_i shares a random polynomial $F_i^{(t)}(x)$ of degree t with the leading coefficient same as that of $A(x)B(x)$. Then $G_i^{(t)}(x) \overset{\text{def}}{=} A(x)B(x) - F_i^{(t)}(x)x^t$ is a polynomial of degree $2t-1$; moreover, $A(0)B(0) = G_i^{(t)}(0)$ and each honest party P_j knows the value of $G_i^{(t)}(j)$. Repeating the above process $t-1$ times, we will get a degree$-t$ polynomial $G_i^{(1)}(x)$ such that $G_i^{(1)}(0) = A(0)B(0)$ and each honest party P_j knows the value $G_i^{(1)}(j)$.

   However, if P_i is corrupted, then $G_i^{(1)}(x)$ may not be a degree$-t$ polynomial. For instance, the leading coefficient of $F_i^{(t)}(x)$ may not be equal to that of $A(x)B(x)$. To prevent the above from happening, P_i is required to share a degree$-t$ polynomial $G_i(x) = G_i^{(1)}(x)$. Again, if P_i is corrupted, these two polynomials may not be the same. But each honest party P_j can check if $G_i(j)$ is equal to $G_i^{(1)}(j)$. If $G_i(j) = G_i^{(1)}(j)$ for all honest P_j, and since the degree of $G_i^{(1)}(x)$ is at most $2t$, it then follows that $G_i(x) = G_i^{(1)}(x)$.

3. Following Lemma 4.2.3, c has been shared as well.

We now give the details:

Step 1 Each party P_i shares two random values $a^{(i)}$ and $b^{(i)}$ using the sharing phase of a VSS protocol with 3-level sharing.

Step 2 Let $a = \sum_i a^{(i)}$ and $b = \sum_i b^{(i)}$. Note that both a and b have been 3-level shared. Let $A(x)$ and $B(x)$ be the polynomials that 1-level share a and b respectively. Notice that $a_i \overset{\text{def}}{=} A(i)$ and $b_i \overset{\text{def}}{=} B(i)$ have been 2-level shared, and P_i knows the polynomial $A_i(x)$ (respectively $B_i(x)$) that 1-level shares a_i (respectively b_i). Let $D_i(x) \overset{\text{def}}{=} A_i(x)B_i(x) = a_ib_i + c_1x + \cdots + c_{2t}x^{2t}$. Each P_i acts as the dealer in $t+1$ (parallel) invocations of the sharing phase of the VSS protocol in Lemma 4.2.1, but with the following modifications on how P_i picks the random bivariate polynomials in the first round:

- In the 1st invocation, P_i picks the random bivariate polynomial such that the coefficient of y^t is equal to c_{2t}. Let $F_i^{(t)}(x,y)$ be the bivariate polynomial chosen by P_i, and let $D_i^{(t)}(y) \overset{\text{def}}{=} F_i^{(t)}(0,y) \overset{\text{def}}{=} r_{t,0} + r_{t,1}y + \cdots + r_{t,t-1}y^{t-1} + c_{2t}y^t$.

- In the 2nd invocation, P_i picks the random bivariate polynomial such that the coefficient of y^t is equal to $(c_{2t-1} - r_{t,t-1})$. Let $F_i^{(t-1)}(x,y)$ be the bivariate polynomial chosen by P_i, and let $D_i^{(t-1)}(y) \overset{\text{def}}{=} F_i^{(t-1)}(0,y) \overset{\text{def}}{=} r_{t-1,0} + r_{t-1,1}y + \cdots + r_{t-1,t-1}y^{t-1} + (c_{2t-1} - r_{t,t-1})y^t$.

- In the 3rd invocation, P_i picks the random bivariate polynomial such that the coefficient of y^t is equal to $(c_{2t-2}-r_{t,t-2}-r_{t-1,t-1})$. Let $F_i^{(t-2)}(x,y)$ be the bivariate polynomial chosen by P_i, and let $D_i^{(t-2)}(y) \stackrel{\text{def}}{=} F_i^{(t-2)}(0,y) \stackrel{\text{def}}{=} r_{t-2,0}+r_{t-2,1}y+\cdots+r_{t-2,t-1}y^{t-1}+(c_{2t-2}-r_{t,t-2}-r_{t-1,t-1})y^t$.

$$\cdots$$

- In the t^{th} invocation, P_i picks the random bivariate polynomial such that the coefficient of y^t is equal to $c_{t+1}-r_{t,1}-r_{t-1,2}-\cdots-r_{2,t-1}$. Let $F_i^{(1)}(x,y)$ be the bivariate polynomial chosen by P_i, and let $D_i^{(1)}(y) \stackrel{\text{def}}{=} F_i^{(1)}(0,y) \stackrel{\text{def}}{=} r_{1,0}+r_{1,1}y+\cdots+r_{1,t-1}y^{t-1}+(c_{t+1}-r_{t,1}-r_{t-1,2}-\cdots-r_{2,t-1})y^t$.

- In the $(t+1)^{\text{th}}$ invocation, P_i picks the random bivariate polynomial $F_i^{(0)}(x,y)$ with the restriction that $D_i^{(0)}(y) \stackrel{\text{def}}{=} F_i^{(0)}(0,y) = A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$ ($A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$ is a polynomial of degree at most t, see Claim 4.2.4).

Step 3 For $0 \le \ell \le t$, let $d_{j,i}^{(\ell)} \stackrel{\text{def}}{=} D_i^{(\ell)}(j)$. Following the protocol specification in Lemma 4.2.1, P_j knows these values; moreover, these values have been 1-level shared. Let $a_{j,i}$ and $b_{j,i}$ be the 1-level shares held by P_j with respect to the value a_i and b_i. Both $a_{j,i}$ and $b_{j,i}$ have been 1-level shared since a_i and b_i have been 2-level shared. If the following equality does not hold:

$$d_{j,i}^{(0)} = a_{j,i}b_{j,i} - \sum_{\ell=1}^{t} j^\ell d_{j,i}^{(\ell)},$$

then P_j broadcasts "complaint P_i".

Step 4 For each P_j that broadcasts "complaint P_i", the parties reconstruct the following values:

$$d_{j,i}^{(0)}, a_{j,i}, b_{j,i}, d_{j,i}^{(1)}, \ldots, d_{j,i}^{(\ell)}$$

The reconstruction is possible since all the above values have been 1-level shared. The parties verify if the complaint by P_j is valid. If the complaint is valid, then P_i is disqualified.

Output Determination Let T be the set of non-disqualified parties with the $2t+1$ smallest identifier. It follows from Claim 4.2.5 that such T exists. It follows from Claim 4.2.6 that for all $P_i \in T$, $a_i b_i$ has been 2-level shared. Following Lemma 4.2.3, $c = ab$ has been 2-level shared.

We first compute the round complexity of the above protocol. Step 1 invokes the sharing phase of a VSS protocol with 3-level sharing. From Lemma 4.2.2, the round complexity of this step is $(8, 1)$. The round complexity of step 2 is $(7, 1)$. Step 3 requires one round of broadcast and step 4 requires one round of interaction. Thus the round complexity of the above protocol is $(17, 3)$.

We now proceed to analyze the above protocol.

Claim 4.2.4 *$A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$ is a polynomial of degree at most t.*

Proof We have:

$$
\begin{aligned}
& A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y) \\
= \quad & a_ib_i + c_1y + \ldots + c_{2t-2}y^{2t-2} + c_{2t-1}y^{2t-1} + c_{2t}y^{2t} \\
- \quad & r_{t,0}y^t + r_{t,1}y^{t+1} + \ldots + r_{t,t-2}y^{2t-2} + r_{t,t-1}y^{2t-1} + c_{2t}y^{2t} \\
- \quad & r_{t-1,0}y^{t-1} + r_{t-1,1}y^t + \ldots + r_{t-1,t-1}y^{2t-2} + (c_{2t-1} - r_{t,t-1})y^{2t-1} \\
- \quad & r_{t-2,0}y^{t-2} + r_{t-2,1}y^{t-1} + \ldots + r_{t-2,t-1}y^{2t-3} + (c_{2t-2} - r_{t,t-2} - r_{t-1,t-1})y^{2t-2} \\
- \quad & \ldots \\
- \quad & r_{1,0}y + r_{1,1}y^2 + \ldots + r_{1,t-1}y^t + (c_{t+1} - r_{t,1} - r_{t-1,2} - \ldots - r_{2,t-1})y^{t+1} \\
= \quad & (c_t - r_{t,0} - r_{t-1,1} - \ldots - r_{1,t-1})y^t \\
+ \quad & (c_{t-1} - r_{t-1,0} - r_{t-2,1} - \ldots - r_{1,t-2})y^{t-1} \\
+ \quad & \ldots + (c_2 - r_{2,0} - r_{1,1})y^2 + (c_1 - r_{1,0})y + a_ib_i
\end{aligned}
$$

■

Claim 4.2.5 *If P_i remains honest by the end of the protocol, then P_i will not be disqualified.*

Proof It follows directly from the protocol specification. ■

Claim 4.2.6 *If P_i is not disqualified, then $D_i^{(0)}(0) = a_ib_i$ and thus a_ib_i has been 2-level shared.*

Proof We will show that $D_i^{(0)}(y) = A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$. It then follows that $D_i^{(0)}(0) = A_i(0)B_i(0) = a_ib_i$. Assume $D_i^{(0)}(y) \neq A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$.

Since $A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$ is a polynomial of degree at most $2t$ and $D_i^{(0)}(y)$ is a polynomial of degree at most t, and there are at least $2t+1$ honest parties, there must exist an honest party P_j such that $D_i^{(0)}(j) \neq A_i(j)B_i(j) - \sum_{\ell=1}^{t} j^\ell D_i^{(\ell)}(j)$. P_j will broadcast "complaint P_i" in step 3. All parties will reconstruct $D_i^{(0)}(j)$, $D_i^{(1)}(j)$, $\ldots$, $D_i^{(t)}(j)$, $A_i(j)$, $B_i(j)$ in step 4 and find that the complaint by P_j is valid. P_i will be disqualified. ■

Following the above two claims, by the end of the protocol, there exist three values a, b, and c such that $c = ab$ and all three values have been 2-level shared. What is left is to show that a, b are both randomly distributed from the view of the adversary.

Claim 4.2.7 *a and b are uniformly distributed given the view of the adversary.*

Proof Note that if P_i is honest, then by the secrecy of VSS, the view of the adversary is independent of $a^{(i)}$ and $b^{(i)}$ in step 1. Hence, $a^{(j)}$ and $b^{(j)}$ shared by a malicious party P_j are independent of $a^{(i)}$ and $b^{(i)}$. (This is so as $a^{(j)}$ and $b^{(j)}$ can be reconstructed from the view of the adversary.) It follows that a and b are randomly distributed after step 1.

We now show that the view of the adversary remains independent of a and b throughout steps 2 – 4. The proof in Lemma 4.2.1 can be directly adapted to show that, in step 2, in the first t invocations of the sharing phase of VSS in which an honest P_i acts as the dealer, the view of the adversary is independent of the coefficient of y^t of the corresponding bivariate polynomials. Thus the adversary does not gain any information in the first t invocations. Now consider the last

($(t+1)^{\text{th}}$) invocation. We claim that the view of the adversary remains independent of $F_i^{(0)}(0,0) = A_i(0)B_i(0)$ after this invocation. If P_i is honest, then $F_i^{(0)}(0,y) = A_i(y)B_i(y) - \sum_{\ell=1}^{t} y^\ell D_i^{(\ell)}(y)$. After the $(t+1)^{\text{th}}$ invocation, the only information the adversary learns about $F_i^{(0)}(0,y)$ are the shares held by corrupted parties P_j (i.e., $F_i^{(0)}(0,j)$). However, the adversary can compute the value $F_i^{(0)}(0,j) = A_i(j)B_i(j) - \sum_{\ell=1}^{t} j^\ell D_i^{(\ell)}(j)$ solely based on its view in the first t invocations. Thus the claim holds.

In step three, an honest party will never complain against another honest party. Hence the adversary does not learn any additional information in step four.

■

The round complexity of the above protocol is $(17,3)$. We can apply a technique from [FGG+06] to modify the protocol such that the round complexity is reduced to $(11,3)$:

- In step 2 of the above protocol, P_i acts as the dealer in $t+1$ (parallel) invocations of the sharing phase of the VSS protocol in Lemma 4.2.1 except that P_i picks the $t+1$ random bivariate polynomials $\{F_i^{(k)}(x,y)\}$ according to some specified constraints.

- In the modified protocol, in step 1, P_i acts as the dealer in $t+1$ (parallel) invocations of the sharing phase of the VSS protocol in Lemma 4.2.1 with the modification that P_i picks the $t+1$ random bivariate polynomials $\{R_i^{(k)}(x,y)\}$ completely at random; in step 2, P_i picks the $t+1$ random bivariate polynomials $\{F_i^{(k)}(x,y)\}$ according to the specified constraints of the above protocol and broadcasts the bivariate polynomials $\{F_i^{(k)}(x,y) - R_i^{(k)}(x,y)\}$. Note that a party can compute its shares of $\{F_i^{(k)}(x,y)\}$ solely based on its shares of $\{R_i^{(k)}(x,y)\}$ and the bivariate polynomials $\{F_i^{(k)}(x,y) - R_i^{(k)}(x,y)\}$.

In the modified protocol, the adversary does not gain any additional information. On the other hand, the round complexity of step 2 is now $(1,1)$ (instead of $(7,1)$). Thus, we have the following lemma:

Lemma 4.2.8 *There exists a protocol with round complexity* $(11,3)$ *tolerating* $t < n/3$ *malicious parties that generates random multiplication triples.*

4.2.3 Constant-Round MPC using One Round of Broadcast

In Section 4.2.3.1, we show how to implement the setup phase using one round of broadcast. Based on this result, in Section 4.2.3.2, we show how to construct a constant-round MPC protocol with round complexity $(O(1),1)$.

4.2.3.1 Implementing the Setup Phase with One Round of Broadcast

By running in parallel the VSS protocol in Lemma 4.2.1 and the protocol for generating random multiplication triples given in Section 4.2.2, we obtain a $(11,3)$-round protocol Π_i that simultaneously allows a party P_i to share its input and generate sufficiently-many random multiplication triples. We remark that, in the resulting protocol, broadcast is invoked in the 7th, 9th and 10th rounds.

We now show how to transform Π_i into a $(21, 1)$-round protocol Π'_i with the following properties:

- By the end of the protocol, all honest parties output a *common* bit $\mathsf{trust}(i)$;
- If P_i is honest, then $\mathsf{trust}(i) = 1$. Moreover, the view of the adversary remains independent of P_i's input.
- 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 $\mathbb{F}$.

Π'_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 Π_i. When a party P is directed by Π_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^{th} or 9^{th} round of Π_i, all parties run the following "simulated broadcast" sub-routine:

- P gradecasts the message m.
- 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 step 1 and step 2, respectively. Within the underlying execution of Π_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 10th round of Π_i, when a party P_j is directed to broadcast a message m, it simply broadcasts this message. In addition, P_j broadcasts 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 compilation from Π_i to Π'_i is essentially the same as the compilation of VSS to *moderated* VSS stated in Section 3.1, except that we retain the last invocation of broadcast in Π_i. The proof of correctness is also similar. We include the proof below for the sake of completeness.

We now prove that Π'_i achieves the claimed properties. Consider the case that P_i is honest. No honest party P_j sets $f_j := 0$. To see this, note that if P_i is honest then $g'_j = 2$ each time the simulated broadcast sub-routine is executed. Furthermore, if P_j outputs some m_j and $g_j = 2$ in step 1 of that sub-routine then, by definition of gradecast (see Definition 2), P_i also outputs m_j in step 1. Hence $m'_j = m_j$ and f_j remains 1. Therefore, at least $2t+1$ parties will broadcast $f = 1$, and so $D_i = 1$. The secrecy of the random multiplication triples following readily from the security of Π_i.

To show the third property, assume $D_i = 1$. It implies that there exists at least one honest party P_j who holds $f_j = 1$ by the end of the protocol. Consider any execution of the simulated broadcast subroutine. We show that the functionality of broadcast was achieved in that execution. Since $f_j = 1$, then $g'_j = 2$. Since $g'_j = 2$, the properties of gradecast imply that any honest party P_k holds $m'_k = m'_j$ and so all honest parties agree on the message that was "broadcast". Hence Π'_i achieves the functionality of Π.

Note that although a malicious P_i can influence the value of $\mathsf{trust}(i)$ output by the parties, the third property of Π_i' implies that the influence is independent of the values of the multiplication triples being shared.

Using the gradecast protocol in Lemma 2.2.1, two rounds of broadcast in Π_i are replaced by 12 rounds of interactions in Π_i'. Thus the round complexity of Π_i' is $(21, 1)$.

The following implementation of the setup phase requires $(21, 1)$-rounds:

1. Run protocols $\{\Pi_i'\}_{i=1}^n$ in parallel.
2. A party P_i is disqualified if $\mathsf{trust}_j(i) \neq 1$. If P_i is disqualified, then a default value is used as the input for P_i.
3. 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 Π_{i^*}'.

4.2.3.2 The MPC Protocol

We start by reviewing the results from [Bea91a, DI05]. Then we show how to obtain a constant-round MPC protocol using one round of broadcast by combining their results and the result from the last section.

The following observation is due to Beaver [Bea91a]:

Lemma 4.2.9 *Suppose a random multiplication triple (a, b, c) and two values x and y have been 1-level shared. Then the value $x \cdot y$ can be 1-level shared after one round of secret reconstruction.*

Proof We include the proof for the sake of completeness. The parties reconstruct the values $d_x = x - a$ and $d_y = y - b$. Then, in a non-interactive manner, each party can compute its share of $d_x d_y + d_x b + d_y a + c$ (using its shares of a, b, c). Note that since $c = ab$, we have $d_x d_y + d_x b + d_y a + c = xy$. (Recall that the characteristic of the field is 2.) ■

Assuming the existence of one-way functions, Damgård and Ishai [DI05] give a multiparty computation protocol with round complexity $(O(1), O(1))$. We review their protocol in the Appendix A.1, but on a high level, their protocol Π_{DI} can be summarized as follows:

Step 1 (In parallel) Each party shares some values in $GF[2]$ using the sharing phase of a VSS protocol such that these values will be 1-level shared.

Step 2 Parties compute shares of degree-3 polynomials in the already shared values (over $GF[2]$).

Step 3 Based on the shares it obtained in the previous step, a party (locally) computes some values and send these to other parties.

Step 4 A party constructs its output based on the values it received in the previous step.

The protocol requires the values shared in step 1 to be in $GF[2]$. while the VSS protocols that we have presented assume the values are from $GF[2^k]$ where $2^k \geq n$. Our VSS protocols can still be used, but with an additional step: to ensure that each shared value x is equal to 0 or 1, parties reconstruct $x^2 - x$ and verify that it is equal to 0 before proceeding to step 2. If the outcome is not zero, then the corresponding party is disqualified and a default value for x is used.

Let K_1 be the total number of bits shared by all parties in step 1 and K_2 be total number of multiplications needed to evaluate the polynomials in step 2. We now present our MPC protocol:

Step 1 Using the protocol for implementing the setup phase as described in Section 4.2.3.1, $K_1 + K_2$ random multiplication triples, as well as the required values as stated in step 1 of Π_{DI} are 1-level shared.

Step 2 For every value x shared by a P_i, the parties reconstruct x^2-x. If $x^2-x \neq 0$, then P_i is disqualified. If P_i is disqualified, then a default value is used as the value of x. Following Lemma 4.2.9, this can be done using two rounds of secret reconstruction and consumes a random multiplication triple.

Step 3 Parties compute shares of degree-3 polynomials in the already shared values over $GF[2^k]$.

Step 4 The parties continue the protocol as specified in step 3 and step 4 of Π_{DI}.

In the above protocol, step 1 requires 21 rounds and 1 round of broadcast. Both step 2 and step 3 require 2 rounds. Step 4 requires 1 round. Thus, we have the following theorem:

Theorem 4.2.10 *Assuming the existence of one-way functions, there exists a secure multiparty computation protocol tolerating* $t < n/3$ *malicious parties with round complexity* $(26, 1)$.

Following Corollary 3.2.2, this immediately gives a MPC protocol with (expected) round complexity $(47, 0)$. However, we can do better. The protocol in Theorem 4.2.10 does not use broadcast until the 21th round. In Section 3.2.1, we have made the observation that some components of our broadcast protocol (in particular, the first phase of an OLE protocol) can be carried out in advance even before the values to be broadcast are known. This observation allows us to save 6 rounds of interaction. Thus, we have the following corollary:

Corollary 4.2.11 *Assuming the existence of one-way functions, there exists a secure multiparty computation protocol tolerating $t < n/3$ malicious parties with (expected) round complexity* 41*.*

4.3 The Authenticated Case ($t < n/2$)

In this section, we assume a PKI and a secure digital signature scheme (which can be constructed assuming the existence of one-way functions [Rom90]). 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 error correction to reconstruct shared values. Thus, we no longer have the linearity of VSS for free. To deal with this issue, we use the information checking tool from [CDD+99]. Roughly speaking, the information checking tool enables us to construct VSS protocols with linearity; moreover, the shares will be "authenticated" (i.e., a corrupted party will not be able to forge an invalid share),

so we no longer need error correction for secret reconstruction. Unfortunately, the protocols as described in [CDD$^+$99] require invocations of broadcast. We show how to eliminate the 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 4.2.2), in the protocol for sharing a random multiplication triple (a, b, c), the parties first share two random numbers 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 applying Lemma 4.2.3), 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, if $t < n/2$, then in the worst case, only $t+1$ (honest) parties will correctly share the product of their shares. Hence if a party does not share the product of its shares correctly, then an additional step is need to make the corresponding shares public.

Roadmap In Section 4.3.1, we define and construct an information checking tool that does not require broadcasting. In Section 4.3.2, we introduce the notion of IC-signature (which can be readily constructed from information checking tool).

Roughly speaking, IC-signature is like standard signature except that it has the linearity property. Using such notion, in Section 4.3.3, we define and construct VSS protocols with 2(3)-level sharing and IC-authentication. Shares generated by these protocols have the linearity property and are "authenticated". In Section 4.3.4, using the above VSS protocols, we construct a protocol that generates random multiplication triples, and then an authenticated MPC protocol that uses one round of broadcast.

Throughout this section, we assume that each party P_i maintains a set of binary variables $\{\mathsf{trust}_{i,j}\}$ which are all initialized to 1. Our protocols are designed such that whenever an honest P_i sets $\mathsf{trust}_{i,j} = 0$, with all but negligible probability, P_j is malicious. For each triple (i, j, k), we assume that P_i and P_k share a common secret $\alpha_{i,j,k} \neq 0 \neq 1$ such that the view of the adversary is independent of it if both P_i and P_k are honest. This can be achieved by having P_i picks and sends a random value $\alpha_{i,j,k} \neq 0 \neq 1$ to P_k at the beginning.

4.3.1 Information Checking Tool

We start by defining and constructing 2-cast protocol which then we will use to construct the information checking tool.

Definition 11 (2-cast): A protocol for three parties where there are two receivers P_i and P_j, and a distinguished sender holds initial input m, is a *2-cast protocol* if the following conditions hold for any adversary:

- All honest parties output the same message m'.

- If the sender is honest, then $m = m'$. $\diamondsuit$

Lemma 4.3.1 *There exists an authenticated* 2*-cast protocol with round complexity* $(2, 0)$.

Proof It follows from observation that the following protocol is a 2-cast protocol:

Round 1 The sender signs and sends m to the two receivers P_i and P_j, and then output m. Let (m_i, σ_i) and (m_j, σ_j) be the message and the signature received by P_i and P_j respectively.

Round 2 The two receivers exchange the signature and the message they receive in the first round.

- If $m_i = m_j$ and σ_i and σ_j are both valid signatures of m_i and m_j, then the receivers output m_i;
- else if σ_i is a valid signature on m_i but σ_j is not a valid signature on m_j, then the receivers output m_i;
- else if σ_j is a valid signature on m_j but σ_i is not a valid signature on m_i, then the receivers output m_j;
- else the receivers output a default message.

■

We now move on to present the definition of information checking tool. In an information checking tool, there are 3 parties, the dealer, the intermediary and the recipient. Roughly speaking, the tool is a weaken version of a standard signature scheme: after receiving a message from the dealer, the intermediary can forward the message to the recipient and convince the recipient that the message is originated from the dealer. However, the intermediary will not be able to convince an arbitrary party (other than the recipient) that it has received such a message from the dealer. This relaxed definition enables us to construct the tool with linearity property. The formal definition (adapted from [CDD$^+$99]) is given below.

Definition There are 3 parties, the dealer P_i, the intermediary P_j and the recipient P_k. An information checking (IC) tool consists of three phases:

1. ICsend: all three parties take part in this phase. P_i hands a secret s to P_j and some auxiliary data to both P_j and P_k. P_j accepts s or rejects s.

2. ICauth: all three parties take part in this phase. If P_j accepts s in ICsend, then P_j ensures that (an honest) P_k will accept s in $\mathsf{ICreveal}$.

3. $\mathsf{ICreveal}$: only P_j and P_k take part in this phase. In this phase P_k receives a value s' from P_j, along with some auxiliary data. P_k either accepts s' or sets $\mathsf{trust}_{k,j} := 0$.

The IC scheme has the following properties:

- Correctness:
  - If P_i, P_j and P_k are honest, and P_i has a secret s, then P_i will accept s in ICSend and P_j will accept s in $\mathsf{ICreveal}$.

– If P_j and P_k are honest, and P_j accepts s in ICSend, then P_k will accept s in $\mathsf{ICreveal}$ with all but negligible probability.

– If P_i and P_k are honest, then in $\mathsf{ICreveal}$, with all but negligible probability, P_k will not accept any value s' that is different from s.

– If an honest party P_k sets $\mathsf{trust}_{k,i} = 0$ (resp. $\mathsf{trust}_{k,j} = 0$) during the execution of the scheme, then with all but negligible probability, P_i (resp. P_j) is corrupted.

- Secrecy: If P_i and P_j are honest, then the view of P_k in the phases ICsend and ICauth is independent of s.

- Linearity: Suppose P_i, P_j, P_k have executed ICsend and ICauth for two different values s_1 and s_2. P_j can reveal $\beta s_1 + \gamma s_2$ to P_k for some known constants β and γ in $\mathsf{ICreveal}$. $\diamondsuit$

Protocol Construction The protocol below is based on the protocol in [CDD+99]. Here, by utilizing the PKI, we are able to avoid using broadcast and (slightly) simplify the original protocol.

We say a triple (a, b, c) is $\alpha_{i,j,k}$-consistent if there exists a degree-1 polynomial w such that $w(0) = a$, $w(1) = b$ and $w(\alpha_{i,j,k}) = c$. In the protocol description below, we write α for $\alpha_{i,j,k}$.

$\underline{\mathsf{ICsend}(P_i,\ P_j,\ P_k,\ s, \alpha)}$ (Round Complexity : $(1, 0)$)

1. P_i signs and sends s to P_j; P_i picks random y, s', y', computes z and z' such that (s, y, z) and (s', y', z') are α-consistent; P_i sends s', y, y' to P_j; P_i sends z

and z' to P_k. If P_j receives a valid signature of s, then P_j accepts s else P_j rejects s.

<u>$\mathsf{ICauth}(P_i, P_j, P_k, s, \alpha)$</u> (Round Complexity : $(5, 0)$)

1. If P_j accepts s in ICsend, then P_j picks a random d and 2-casts $(d, s'+ds, y'+dy)$ to P_i and P_k else P_j 2-casts (Not taking part) and exits this phase.

2. If P_i detects that P_j has cheated in step 1, then P_i 2-casts (s, y) to P_j and P_k; P_k adjusts the value of z such that (s, y, z) are α-consistent; else if $(s' + ds, y' + dy, z' + dz)$ is not α-consistent, then P_k sets $\mathsf{trust}_{k,i} := 0$.

3. If P_j detects that P_i has cheated in step 2, then P_j sends s (along with a signature of P_i on s) and y to P_k; if that happens, then P_k adjusts the value of z such that (s, y, z) is α-consistent.

<u>$\mathsf{ICreveal}(P_j, P_k, s, \alpha)$</u> (Round Complexity : $(1, 0)$)

1. P_j sends (s, y) to P_k; if $\mathsf{trust}_{k,j} = 0$, then P_k will always reject; else if $\mathsf{trust}_{k,i} = 0$, then P_k will always accept; else P_k accepts or sets $\mathsf{trust}_{k,j} := 0$ depending on whether (s, y, z) is α-consistent.

Proof of Correctness We now move on to prove that the above protocol is indeed an information checking tool.

- Correctness:

  - If P_i, P_j and P_k are all honest, then it is easy to see that P_j will accept s in ICsend and P_k will accept s in ICval.

– If P_j and P_k are honest, and P_j accepts s in ICsend, then there are four possible cases:

1. P_i does not 2-cast anything in step 2 of ICauth and $(s' + ds, y' + dy, z' + dz)$ is α-consistent.
2. P_i does not 2-cast anything in step 2 of ICauth and $(s' + ds, y' + dy, z' + dz)$ is not α-consistent.
3. P_i 2-cast $(\bar{s}, \bar{y})$ in step 2 of ICauth and $(\bar{s} \neq s$ or $\bar{y} \neq y)$.
4. P_i 2-casts (s, y) in step 2 of ICauth phase.

Following the descriptions of the protocol, an honest P_k will always accept what an honest P_j sent it in ICval for case (2) and case (4). For case (3), an honest P_j will send (s, y) with a valid signature from P_i on s to P_k in step 3. P_k will adjust the value of z accordingly. Hence P_k will accept s in ICval. For case (1), P_k will not accept s if and only if (s, y, z) is not α-consistent. We claim that if (s, y, z) is not α-consistent, then the probability that $(s' + ds, y' + dy, z' + dz)$ is α-consistent is negligible over random choice of d. Assume there exists two values a and b such that $a \neq b$, both $(s'+as, y'+ay, z'+az)$ and $(s'+bs, y'+by, z'+bz)$ are α-consistent. Then (s, y, z) is α-consistent too. Hence if (s, y, z) is not α-consistent, there exists at most one value of d such that $(s' + ds, y' + dy, z' + dz)$ is α-consistent. The claim then follows.

– If P_i and P_k are honest, then note that the view of P_j during ICsend and ICauth is independent of α. If P_k accepts a value $s' \neq s$, then it means that a malicious P_j has guessed the value of α correctly. However, even if multiple IC protocols can be executed between the triple (P_i, P_j, P_k) using the same value α, a malicious P_j can only have one chance of guessing the right value of α (P_k will set $\mathsf{trust}_{k,j} := 0$ if P_j makes a wrong guess). Hence P_k will only accept $s' \neq s$ with a negligible probability.

– It follows from the above proof that if P_k is honest, then for any honest party $P_a \in \{P_i, P_j\}$, then P_k will set $\mathsf{trust}_{k,a} := 0$ only with negligible probability.

- Secrecy: If P_i and P_j are honest, then in ICsend and $\mathsf{ICreveal}$, the adversary (if P_k is corrupted) learns $z, z', d, s' + ds, y' + dy$. However, since P_i and P_j are both honest, $(s' + ds, y' + dy, z' + dz)$ is always α-consistent. Given $s' + ds$, $z' + dz$ and α, the adversary can compute $y' + dy$ by itself. It is easy to see that the distribution of $z, z', d, s' + ds$ is independent of s. Hence secrecy holds.

- Linearity: Suppose P_i, P_j, P_k have executed ICsend and ICauth for two different values s_1 and s_2 using the same α. Now P_j wants to reveal $\beta s_1 + \gamma s_2$ to P_k for some known constants β and γ. In $\mathsf{ICreveal}$, P_j can send $(\beta s_1 + \gamma s_2, \beta y_1 + \gamma y_2)$ to P_k; P_k accepts or rejects depending on $\mathsf{trust}_{k,i}$, $\mathsf{trust}_{k,j}$ and whether $(\beta s_1 + \gamma s_2, \beta y_1 + \gamma y_2, \beta z_1 + \gamma z_2)$ is $\alpha_{i,j,k}$-consistent.

4.3.2 IC-Signature

It will be handy to have the notion of IC-signature (a similar definition appeared in [CDD+99, Section 4]) when we define and construct VSS protocols with

2(3)-level sharing. Roughly speaking, an IC-signature scheme is similar to a (standard) signature scheme but it can have the linearity property. There are two parties P_i and P_j, and P_i has a secret s. An IC-signature scheme has three phases:

- $\mathsf{ICssend}(P_i, P_j, s)$: Executes $\mathsf{ICsend}(P_i, P_j, P_k, s, \alpha_{i,j,k})$ for all parties P_k. If P_j receives the same s in all invocations, then P_j accepts s else P_j rejects.

- $\mathsf{ICsauth}(P_i, P_j, s)$: If P_j accepts s in $\mathsf{ICssend}$, then executes $\mathsf{ICsauth}(P_i, P_j, P_k, s, \alpha_{i,j,k})$ for all parties P_k.

- $\mathsf{ICsreveal}(P_j, s)$: If P_j accepts s in $\mathsf{ICssend}$, then executes $\mathsf{ICreveal}(P_j, P_k, s)$ for all parties P_k.

The round complexities of $\mathsf{ICssend}$, $\mathsf{ICsauth}$ and $\mathsf{ICsreveal}$ are $(1, 0)$, $(5, 0)$ and $(1, 0)$ respectively. If $\mathsf{trust}_{j,i} = 1$ after executions of $\mathsf{ICssend}(P_i, P_j, s)$ and $\mathsf{ICsauth}(P_i, P_j, s)$, then we say P_j *obtains a IC-signature of* s *from* P_i. If $\mathsf{trust}_{k,j} = 1$ after the execution of $\mathsf{ICsreval}(P_j, s)$, then we say P_k *accepts the IC-signature of* s from P_j.

Following the properties of the information checking scheme given in the previous section, it is easy to see that the following conditions hold with all but negligible probability:

- If P_j obtains a IC-signature of s from P_i, then an honest party P_k will accept s from P_j in $\mathsf{ICsreveal}$. Moreover, if P_i and P_j are both honest, then the view of the adversary is independent of s before $\mathsf{ICsreveal}$.

- If a (malicious) P_j does not obtain a IC-signature of s' from P_i, then no honest party P_k will accept s' in ICsreveal.

- If P_j obtains a IC-signature of s_1 and s_2 from P_i, then P_j can reveal the value of $\beta s_1 + \gamma s_2$ to all parties by executing $\mathsf{ICsreveal}(P_j, \beta s_1 + \gamma s_2)$.

4.3.3 Generalized Secret Sharing with IC-signatures

Next, we provide an analogue of Section 4.2.1 for secret sharing using IC-signatures. We start by extending the definitions of 2(3)-level sharing.

Definition 12 (2-level sharing with IC-authentication): We say that a value s has been 2-level shared with IC-authentication if the following two conditions hold:

- There exists $t+1$ polynomials $F_s(x), F_{s_1}(x), \ldots, F_{s_n}(x)$, each of degree at most t, such that $F_s(0) = s$, $F_{s_1}(0) = F_s(1)$, $\ldots$, $F_{s_n}(0) = F_s(n)$.
- Each honest P_i knows the polynomial $F_{s_i}(x)$. In addition, for all $j \in [n]$, either P_j has obtained a IC-signature on $F_{s_i}(j)$ from (potentially corrupted) P_i or the value of $F_{s_i}(j)$ has been made public.

$\diamondsuit$

We call $F_s(x)$ the polynomial that *1-level shares s*.

Definition 13 (3-level sharing with IC-authentication): We say that a value s has been 3-level shared with IC-authentication if the following conditions hold:

- s has been 2-level shared with IC-authentication.
- Let $F_s(x)$ be the polynomial that *1-level shares s*. For $1 \leq i \leq n$, the value

$F_s(i)$ has been 2-level shared; P_i knows the polynomial $F_{s_i}(x)$ that 1-level shares $F_s(i)$.

$\diamondsuit$

We now generalize the definition of VSS (cf. Definition 3) as follows:

Definition 14 (Generalized verifiable secret sharing): A two-phase 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 and IC-authentication tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Validity By the end of the first phase, some value s' is 2-level (resp., 3-level) shared with IC-authentication. Moreover, if the dealer is honest then $s' = s$.

Secrecy If the dealer is honest by the end of the first phase (the *sharing phase*), then at the end of the first phase the joint view of the malicious parties is independent of the dealer's input s.

Reconstruction All honest parties output s' at the end of the reconstruction phase.

$\diamondsuit$

Remark If both s and s' have been 2-level shared with IC-authentication, then following the linearity of IC-signature, $s + s'$ has been 2-level shared with IC-authentication as well.[2]

[2]It may be the case that for some pair (i, j), P_j has obtained a IC-signature on $F_{s_i}(j)$ from P_i while the value of $F_{s'_i}(j)$ has been made public. However, following the linearity of IC-signature, it means that P_j has obtained a IC-signature on $F_{s_i}(j) + F_{s'_i}(j)$ from P_i.

We now proceed to construct authenticated VSS protocol with 2(3)-level sharing and IC-authentication. First, we make the following observation related to secret reconstruction:

Lemma 4.3.2 *Suppose s has been 2(3)-level shared with IC-authentication. Assuming $t < n/2$, then the parties can reconstruct s using one round of interaction.*

Proof Note that by definition, if s has been 3-level shared, then s has been 2-level shared as well. If s has been 2-level shared, then s can be constructed by the following routine:

Message Exchange For each i, if the value of $F_{s_i}(k)$ has not been made public, then P_k executes $\mathsf{ICsreveal}(P_k, F_{s_i}(k))$.

Secret Reconstruction For each i, let

$$\mathcal{S}_i^k = \left\{ (j, F_{s_i}(j)) : \begin{array}{c} F_{s_i}(j) \text{ has been made public, or} \\ P_k \text{ accepts the IC-signature of } F_{s_i}(j) \text{ from } P_j \end{array} \right\}$$

and

$$\mathcal{S}^k = \left\{ (j, F'_{s_i}(0)) : \begin{array}{c} \text{A polynomial } F'_{s_i}(x) \text{ of degree at most } t \\ \text{can be obtained by interpolating the points in } \mathcal{S}_i^k \end{array} \right\}$$

Party P_k then outputs $s = F'_s(0)$ where $F'_s(x)$ is the polynomial obtained through interpolating the points in $\mathcal{S}^k$.

First, we show that if a polynomial $F'_{s_i}(x)$ of degree at most t can be interpolated from the set of points in $\mathcal{S}_i^k$, then $F'_{s_i}(x)$ is equal to $F_{s_i}(x)$. Consider an honest party P_j. Either $F_{s_i}(j)$ has been made public or P_k will (only) accept the

IC-signature of $F_{s_i}(j)$ from P_j (following the definition of IC-signature, see Section 4.3.2). Hence $F'_{s_i}(j) = F_{s_i}(j)$. Since there are at least $t+1$ honest parties, both $F'_{s_i}(x)$ and $F_{s_i}(x)$ are polynomials of degree at most t, it follows that $F'_{s_i}(x) = F_{s_i}(x)$.

Next, we show that if P_i is honest, then P_k can always interpolate the polynomial $F_{s_i}(x)$ from the set $\mathcal{S}_i^k$. If the value $F_{s_i}(j)$ has not been made public, then following the definition of IC-signature, P_k will only accept a value from P_j if that value is equal to $F_{s_i}(j)$. Moreover, if P_j is honest, then P_k will always accept the IC-signature of $F_{s_i}(j)$. Since there are at least $t+1$ honest parties, and $F_{s_i}(x)$ is of degree at most t, it follows that P_k can always interpolate the polynomial $F_{s_i}(x)$ from the set $\mathcal{S}_i^k$.

Since $F_s(x)$ is of degree at most t and there are at least $t+1$ honest parties, following from the previous paragraph, $F'_s(x) = F_s(x)$ can always be reconstructed from $\mathcal{S}^k$. Hence the party P_k can reconstruct $s = F_s(0)$. ■

Given the above lemma, we can now focus on constructing the sharing phase of VSS protocol with 2(3)-level sharing and IC-authentication.

4.3.3.1 VSS Protocol with 2-Level Sharing

The following protocol is based on the sharing phase of the authenticated VSS protocol for $t < n/2$ given in Lemma 2.2.5. We modify the protocol so that it is with 2-level sharing and IC-authentication.

We consider a finite field $\mathbb{F}$ with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n]$ can be injectively mapped to $\mathbb{F}$. If the dealer is *disqualified* then execution of the protocol halts, and a default value s' is 2-level shared. We say an ordered sequence of values $(v_1, \ldots, v_n) \in \mathbb{F}^n$ is *t-consistent* if there exists a polynomial $f(x)$ of degree at most t such that $f(i) = v_i$ for $1 \leq i \leq n$.

Step 1 The dealer chooses a random bivariate polynomial $F(x, y)$ of degree at most t in each variable with $F(0, 0) = s$. Let $a_{i,j} = b_{i,j} \stackrel{\text{def}}{=} F(i, j)$. The dealer sends to party P_i the values $a_{1,i}, \ldots, a_{n,i}$ and $b_{i,1}, \ldots, b_{i,n}$, along with a digital signature on each such value.

Step 2 If P_i receives all values (with valid signatures) from the dealer as specified in round 1, $\{a_{1,i}, a_{2,i}, \ldots, a_{n,i}\}$ and $\{b_{i,1}, b_{i,2}, \ldots, b_{i,n}\}$ are both t-consistent, then P_i executes $\mathsf{ICssend}(P_i, P_j, a_{j,i})$ for all j, else P_i sends "No IC-signature" to all P_j.

Step 3 The following two sub-routines are executed in parallel:

Sub-routine a If P_i does not send "No IC-signature" to P_j in step 2, then executes $\mathsf{ICsauth}(P_i, P_j, a_{j,i})$.

Sub-routine b If P_i sends "No IC-signature" to all P_j in step 2, then P_i broadcasts "Complaint: dealer"; else if P_i receives "No IC-signature" from a party P_j, or P_i does not accept $b_{i,j}$ in $\mathsf{ICssend}(P_j, P_i, a_{i,j})$, then P_i broadcasts $b_{i,j}$ and the signature of the dealer on $b_{i,j}$ (we say the value $b_{i,j}$ has been made public).

Step 4 For any party P_j that broadcasts a complaint in step 3,

- The dealer broadcasts $\vec{a}_j = (a_{1,j}, \ldots, a_{n,j})$, $\vec{b}_j = (b_{j,1}, \ldots, b_{j,n})$.

- If P_i does not broadcast a complaint in step 3, then P_i broadcast $a_{j,i}$ and $b_{i,j}$ with the dealer's signature on these two values.

For any party P_j that broadcasts a value $b_{j,i}$ (with a valid signature from the dealer on $b_{j,i}$) in step 3,

- If P_i does not broadcast a complaint in step 3, then P_i broadcasts $a_{j,i}$ and the signature of the dealer on $a_{j,i}$.

The dealer is disqualified if one of the following conditions hold:

- If a party P_j broadcasts a complaint in step 3, but the dealer does not respond to it in step 4, or either $\vec{a}_j$ or $\vec{b}_j$ broadcasted by the dealer is not t-consistent or $a_{j,j} \neq b_{j,j}$.
- There exists a pair (i, j) such that each of $a_{i,j}$ and $b_{i,j}$ has been broadcast by the dealer in step 4 or has been broadcast by a party along with the dealer's signature on the value, and $a_{i,j} \neq b_{i,j}$.

We now show that the above protocol implements the sharing phase of VSS with 2-level sharing and IC-authentication.

We first prove secrecy. If the dealer is honest, no honest party will send "No IC-signatures" to other parties in step 2. Furthermore, if P_i and P_j are both honest, then P_j will accept $b_{j,i} = a_{j,i}$ in $\mathsf{ICssend}(P_i, P_j, a_{j,i})$. Hence the value of $a_{j,i}$ will not be broadcast in step 3 nor step 4. It follows that the information the adversary has about s by the end of the sharing phase consists entirely of the values sent to the malicious parties by the dealer in round 1. Secrecy follows since $F(x, y)$ is a random bivariate polynomial of degree at most t.

Next we prove validity. The case for a disqualified malicious dealer is immediate. On the other hand, it is easy to see that an honest dealer will never be disqualified. To complete the proof, we are going to show that by the end of the protocol, if the dealer is not disqualified, then there exists a bivariate polynomial $F'(x, y)$ of degree at most t such that

- If the dealer is honest, then $F'(x, y) = F(x, y)$.
- Each P_i knows the polynomials $F'(x, i)$ and $F'(i, y)$. In addition, for each P_j, either P_i has obtained a IC-signature on $F'(i, j)$ from P_j (even if P_j is malicious) or the value of $F'(i, j)$ has been made public.

Takes $F'_s(x) = F'(0, x)$ and $F'_{s_i}(x) = F'(x, i)$, it follows that $F'(0, 0)$ has been 2-level shared with IC-authentication.

It follows readily from the protocol that the above holds for an honest dealer. We now consider a malicious dealer that is not disqualified. For each honest party P_i, let $\vec{a}_i = (a_{1,i}, \ldots, a_{n,i})$, $\vec{b}_i = (b_{i,1}, \ldots, b_{i,n})$ be the values held by P_i by the end of the protocol. Either P_i obtains $\vec{a}_i$ and $\vec{b}_i$ from the dealer in step 1 or the dealer broadcasts $\vec{a}_i$ and $\vec{b}_i$ in step 4. Since the dealer is not disqualified, $\vec{a}_i$ and $\vec{b}_i$ are both t-consistent.

If P_i and P_j are both honest, then $a_{i,j} = b_{i,j}$. Assume this is not true. There are four possible cases:

- Both P_i and P_j do not broadcast a complaint in step 3 (i.e., both parties do not send "No IC-signature" to other parties in step 2). Then it must be the case that P_i does not accept $b_{i,j}$ in $\mathsf{ICssend}(P_j, P_i, a_{i,j})$ in step 2. Hence P_i will make $b_{i,j}$ public in step 3. But then P_j will broadcast $a_{i,j}$ (with the signature of the dealer on it) in step 4. The dealer will be disqualified since $a_{i,j} \neq b_{i,j}$.

- P_i broadcasts a complaint in step 3 but P_j does not. Then the dealer broadcasts $b_{i,j}$ in step 4. P_j will broadcast $a_{i,j}$ in round 4 (with the signature of the dealer on it). Hence the dealer will be disqualified.

- P_j broadcasts a complaint in step 3 but P_i does not. This case is symmetric to the previous case.

- Both P_i and P_j broadcast a complaint in step 3: the dealer will be disqualified since the dealer broadcasts $a_{i,j}$ and $b_{i,j}$ in step 4 but the two values are not equal.

Since there are at least $t+1$ honest parties, and both $\vec{a}_i$ and $\vec{b}_i$ are t-consistent if P_i is honest, there exists a bivariate polynomial $F'(x, y)$ of degree at most t such that $F'(i, k) = b_{i,k}$ and $F'(k, i) = a_{k,i}$ for all honest parties P_i and all $1 \leq k \leq n$.

To complete the proof for the second item, suppose a party P_i does not obtain a IC-signature on $F'(i, j) = b_{i,j}$ from P_j in step 2. In that case, either P_i will make $F'(i, j)$ public in step 3 or P_i broadcasts a complaint in step 3 (and then the dealer will make the value public by broadcasting it in step 4).

The round complexity of the above protocol is $(8, 2)$: each step 1 and step 2 requires 1 round, sub-routine b of step 3 requires 1 round of broadcast, sub-routine a of step 3 requires 5 rounds, and step 4 requires 1 round of broadcast.

Reducing the number of rounds of broadcast: Using the method mentioned in the proof of Lemma 2.2.5, we can reduce the round complexity of the above protocol to $(8, 1)$. In more details, the above protocol invokes two rounds of broadcast in the following manner:

- In sub-routine b of step 3, if certain conditions hold, then P_i broadcasts a message x.
- In step 4, if P_i broadcasts a message y in sub-routine b of step 3, then P_j broadcasts a message z.

Call the above protocol Π. We construct a protocol Π' with round complexity $(8, 1)$. Π' is the same as Π except that:

1. In sub-routine b of step 3,
   (a) If P_i is supposed to broadcast x in Π, then P_i signs and sends x to all parties.
   (b) If P_k receives x with a valid signature from P_i, then P_k forwards x (with the signature of P_i on x) to all parties.
2. In step 4,
   - If P_j receives at least 1 copy of y (with a valid signature of P_i on y) from other parties in the previous step, then P_j broadcasts (P_i, y, z).
   - In parallel to the above, if P_k receives x with a valid signature from P_i in sub-routine b, then P_k broadcasts (P_i, x).

The party determines the output in Π' the same way as they determine the output in Π, except that they consider the set of broadcast values as follow:

- If at least $t+1$ parties broadcast (P_i, x) in step 4 of Π', then P_i is considered to broadcast x in step 3 of Π.
- If at least $t+1$ parties broadcast (P_i, x) and P_j broadcasts (P_i, x, z) in step 4 of Π', then P_j is considered to broadcast z in step 4 of Π.

In Π', sub-routine b of step 3 now requires 2 rounds, but the overall round complexity remains $(8, 1)$ (Notice that sub-routine a of step 3 requires 5 rounds).

Using the same argument as in Lemma 2.2.5, we can show that Π' implements the sharing phase of VSS with 2-level sharing and IC-authentication. Thus we have the following:

Lemma 4.3.3 *There exists an authenticated VSS protocol with 2-level sharing and IC-authentication tolerating $t < n/2$ malicious parties such that the round complexity of its sharing phase is $(8, 1)$ and the round complexity of its reconstruction phase is $(1, 0)$.*

4.3.3.2 VSS Protocol with 3-Level Sharing

In this section, we construct a protocol to implement the sharing phase of VSS with 3-level sharing and IC-authentication. Basically, the dealer will run $n+1$ invocations of the protocol from the previous section in parallel with the following modifications:

1. In the first step,

   - In the first invocation, the dealer picks a random bivariate polynomial $F_0(x, y)$ such that $F_0(0, 0) = s$.
   - In the k^{th} $(1 \leq k \leq n)$ invocation, the dealer picks random bivariate polynomials $F_k(x, y)$ with the restriction that $F_k(0, y) = F_0(k, y)$.

2. In the second step, to ensure that $F_k(0, y) = F_0(k, y)$ for all $1 \leq k \leq n$, each party P_i will do this additional checking:

   - Let $\{a^k_{1,i}, a^k_{2,i}, \dots, a^k_{n,i}\}$ be the shares P_i obtained in the k^{th} invocation. P_i interpolates the polynomial $F^i_k(x)$ from $\{a^k_{1,i}, a^k_{2,i}, \dots, a^k_{n,i}\}$ and check if $F^i_k(0) = a^0_{k,i}$.
   - If there exists a k such that $F^i_k(0) \neq a^0_{k,i}$, then P_i will always send "No IC-signature" to all parties in *all* $k + 1$ invocations (instead of executing $\mathsf{ICsauth}$).

3. In the forth step, if the dealer is required to broadcast the shares of a party P_i in a particular invocation, then the dealer broadcasts the shares of P_i in *all* invocations. At the end of the protocol, the dealer will be disqualified if there exists a k such that $F_k(0, i) \neq a^0_{k,i}$ (in addition to the existing conditions).

We now give the description of the protocol.

Step 1 The dealer chooses a random bivariate polynomial $F_0(x, y)$ of degree at most t in each variable with $F_0(0, 0) = s$. In addition, the dealer chooses n

random bivariate polynomials $F_1(x,y),\dots,F_n(x,y)$ of degree at most t in both variables with $F_k(0,y) = F_0(k,y)$ for $1 \le k \le n$. Let $a^k_{i,j} = b^k_{i,j} \overset{\text{def}}{=} F_k(i,j)$. The dealer sends to party P_i the values $a^k_{1,i},\dots,a^k_{n,i}$ and $b^k_{i,1},\dots,b^k_{i,n}$, along with a digital signature on each such value.

Step 2 If P_i receives all values (with valid signatures) from the dealer as specified in round 1, for all $0 \le k \le n$, $\{a^k_{1,i}, a^k_{2,i},\dots,a^k_{n,i}\}$, $\{b^k_{i,1}, b^k_{i,2},\dots,b^k_{i,n}\}$ are t-consistent, and for all $1 \le k \le n$, there exists a polynomial $F^i_k(x)$ of degree at most t such that $F^i_k(z) = a^k_{z,i}$ for all $1 \le z \le n$ and $F^i_k(0) = a^0_{k,i}$, then executes $\mathsf{ICssend}(P_i, P_j, a^k_{j,i})$ for all j and k, else P_i sends "No IC-signature" to all P_j.

Step 3 The following two sub-routines are executed in parallel:

Sub-routine a If P_i does not send "No IC-signature" to P_j in step 2, then executes $\mathsf{ICsauth}(P_i, P_j, a^k_{j,i})$ for all k.

Sub-routine b If P_i sends "No IC-signature" to all parties in step 2, then P_i broadcasts "Complaint: dealer"; else if P_i receives "No IC-signature" from P_j or P_i does not output (accept, $b^k_{i,j}$) in $\mathsf{ICssend}(P_j, P_i, a^k_{i,j})$ for any k, then P_i broadcasts $b^k_{i,j}$ and the signature of the dealer on $b^k_{i,j}$ for all k.

Step 4 For any party P_j that broadcasts a complaint in step 3, then for all $0 \le k \le n$,

- The dealer broadcasts $\vec{a}^k_j = (a^k_{1,j},\dots,a^k_{n,j})$, $\vec{b}^k_j = (b^k_{j,1},\dots,b^k_{j,n})$.
- If P_i does not broadcast a complaint in step 3, then P_i broadcast $a^k_{j,i}$ and $b^k_{i,j}$ with the dealer's signature on these two values.

For any party P_j that broadcasts a value $b^k_{j,i}$ (with a valid signature from the dealer on $b^k_{j,i}$) in step 3,

- If P_i does not broadcast a complaint in step 3, then P_i broadcasts $a^k_{j,i}$ and the signature of the dealer on $a^k_{j,i}$.

The dealer is disqualified and a default value is 3-level shared if one of the following conditions hold:

- If a party P_j broadcasts a complaint in step 3, but the dealer does not respond to it in step 4, or for some k either $\vec{a}^k_j$ or $\vec{b}^k_j$ broadcast by the dealer is not t-consistent, or $a^k_{j,j} \neq b^k_{j,j}$ or there does not exist a polynomial $F_k(x)$ of degree at most t such that $F_k(0) = a^0_{k,i}$ and $F_k(z) = a^k_{z,i}$ for all z.
- There exists a triple (i, j, k) such that each of $a^k_{i,j}$ and $b^k_{i,j}$ has been broadcasted by the dealer in step 4 or has been broadcasted by a party along with the dealer's signature on the value, and $a^k_{i,j} \neq b^k_{i,j}$.

We now argue that the above protocol implement the sharing phase of VSS with 3-levels sharing.

We first argue secrecy. It follows from the proof in the last section that if P_k is honest, then the adversary does not learn any additional information on $F_k(0,0) = F_0(0,k)$. Thus the view of the adversary remains independent of $s = F_0(0,0)$.

Next we argue validity. The case for an honest dealer or a disqualified malicious dealer is obvious. We consider a malicious dealer who is not disqualified. Following the proof in the last section, $F_k(0,0)$ has been 2-level shared for $0 \leq k \leq n$. To complete the proof, it suffices to argue that $F_k(0,y) = F_0(k,y)$ for $1 \leq k \leq n$. Since the dealer is not disqualified, then $F_k(0,i) = a^0_{k,i} = F_0(k,i)$ for all honest parties P_i. As $F_k(0,y)$ is a polynomial of degree at most t, $F_k(0,y) = F_0(k,y)$.

The above protocol has round complexity $(8,2)$. Using exactly the same technique as described in the last section, we can reduce the round complexity to $(8,1)$. Thus, we have the following lemma:

Lemma 4.3.4 *There exists an authenticated VSS protocol with 3-level sharing and IC-authentication tolerating $t < n/2$ malicious parties such that the round complexity of its sharing phase is $(8,1)$ and the round complexity of its reconstruction phase is $(1,0)$.*

4.3.4 Generating Random Multiplication Triples

We describe the protocol for generating one random multiplication triple, as in Section 4.2.2, the protocol can be parallelized to generate as many triples as needed.

The protocol given below is based on the multiplication protocol in [CDD$^+$99]. However, for the protocol given in [CDD$^+$99], each time when a (corrupted) party is disqualified in sharing product of its shares, the protocol is rewinded to reveal the shares of the disqualified party. As a result, the round complexity of the protocol may not be a constant since there can be a linear number of rewinding. We use VSS with 3-level sharing as a building block in our construction so that rewinding is not needed to reveal the shares of the disqualified parties (since these shares would have already been 2-level shared). Hence the round complexity of the overall protocol can remain to be a constant.[3]

Step 1 Each party P_i picks two random values a_i and b_i and shares them using the sharing phase of the VSS protocol with 3-level sharing and IC-authentication in section 4.3.3.2.

Let $A = \sum a_j$ and $B = \sum b_j$; let $A(x)$ and $B(x)$ be the polynomials that 1-level share A and B respectively; and let $A_i \overset{\text{def}}{=} A(i)$ and $B_i \overset{\text{def}}{=} B(i)$. For each $1 \leq i \leq n$, the following steps are carried out in parallel :

Step 2 The following three routines are carried out in parallel:

(a) Let $A_i(x)$ and $B_i(x)$ be the polynomials that 1-level share A_i and B_i respectively. P_i shares A_iB_i using the sharing phase of the VSS protocol with 2-level sharing and IC-authentication in Section 4.3.3.1. Let $C_i(x)$ be the polynomial that 1-level shares A_iB_i.

(b) P_i picks a random β_i. P_i shares β_i and β_iB_i using the sharing phase of the VSS protocol with 2-level sharing and IC-authentication in Section 4.3.3.1.. Let $f_{\beta_i}(x)$ and $f_{\beta_iB_i}(x)$ be the polynomials that 1-level share β_i and β_iB_i respectively.

(c) Each P_j picks a random r_i^j and shares it using the sharing phase of a VSS protocol.

[3]In [DI05], a primitive called EVSS is used to handle this problem. EVSS is similar to VSS with 3-level of sharing. However, we are able to construct more round-efficient protocol due to the usage of PKI.

Step 3 The parties reconstruct $\{r_i^j\}$ and computes $r_i = \sum r_i^j$.

Step 4 P_i broadcasts the following two polynomials:

- $f_i^1(x) = r_i A_i(x) + f_{\beta_i}(x)$
- $f_i^2(x) = f_i^1(0)B_i(x) - f_{\beta_i B_i}(x) - r_i C_i(x)$.

Step 5 If $f_i^1(k) \neq r_i A_i(k) + f_{\beta_i}(k)$ or $f_i^2(k) \neq f_i^1(0)B_i(k) - f_{\beta_i B_i}(k) - r_i C_i(k)$, then P_k broadcasts a complaint, as well as the signatures of P_i on $f_{\beta_i}(k)$, $f_{\beta_i B_i}(k)$ and $C_i(k)$ (if such values have not been made public)[4].

Step 6 Let $F_{a_j}(x)$ and $F_{b_j}(x)$ be the polynomials that 1-level share a_j and b_j respectively; let $F_{a_j^i}(x)$ and $F_{b_j^i}(x)$ be the polynomials that 1-level share $F_{a_j}(i)$ and $F_{b_j}(i)$ respectively (recall that a_j and b_j have been 3-level shared). If P_k broadcasts a complaint, then for all $1 \leq j \leq n$, P_i broadcasts $F_{a_j^i}(k)$, $F_{b_j^i}(k)$ and the signatures of P_k on each of the values (assuming the corresponding value has not been made public)[5].

Step 7 P_i is disqualified if *one* of the following conditions hold:

- P_k broadcasts a complaint in step 6 but P_i does not respond to it, or P_i fails to provide the signatures of P_k on the required values, or $f_i^1(k) \neq r_i \sum_j F_{a_j^i}(k) + f_{\beta_i}(k)$, or $f_i^2(k) \neq f_i^1(0) \sum_j F_{b_j^i}(k) - f_{\beta_i B_i}(k) - r_i C_i(k)$.
- $f_i^2(0) \neq 0$.

[4] P_k has these signatures if the VSS protocol given in Section 4.3.3.1 is used.

[5] P_k has these signatures since the VSS protocol given in Section 4.3.3.2 is used.

If P_i is disqualified, then the parties reconstruct the values of A_i and B_i (note that A_i and B_i have already been 2-level shared). Otherwise, it will follow from our proof that $C_i(0) = A_iB_i$ has been 2-level shared.

Output Determination Following Lemma 4.2.3, the parties can then compute the shares of C non-interactively.

We now prove the correctness of the protocol by showing the following three lemmas:

Lemma 4.3.5 *The values of $A = \sum a_i, B = \sum b_i$ are randomly distributed.*

Proof The above follows directly from the secrecy property of VSS. ■

Lemma 4.3.6 *If P_i is honest, then the view of the adversary remains independent of A_iB_i by the end of the protocol.*

Proof The polynomial $f_i^1(x)$ reveals no information on A_iB_i as the view of the adversary is independent of β_i. Furthermore, if P_i is honest, then an adversary corrupting t parties can reconstruct the polynomial $f_i^2(x)$ given its view in the first three steps: $f_i^2(x)$ is a polynomial of degree at most t, $f_i^2(0) = r_iA_iB_i + \beta_iB_i - \beta_iB_i - r_iC_i = 0$, and the adversary knows $f_i^2(k) = f_i^1(0)B_i(k) - f_{\beta_iB_i}(k) - r_iC_i(k)$ for every malicious party P_k. Hence broadcasting the polynomial $f_i^2(x)$ in step 4 does not give the adversary any new information. An honest party P_k will not broadcast a complaint against P_i in step 5. Thus, the adversary does not learn any additional information in steps 5 and 6. The lemma then follows as P_i will not be disqualified in step 7. ■

Following the above lemma, the view of the adversary remains independent of A and B after the protocol.

Lemma 4.3.7 *If P_i is not disqualified, $C_i(0) = A_i B_i$ with all but negligible probability.*

Proof If P_i is not disqualified, then $f_i^1(x) = r_i A_i(x) + f_{\beta_i}(x)$ and $f_i^2(x) = f_i^1(0)B_i(x) - f_{\beta_i B_i}(x) - r_i C_i(x)$ since $f_i^1(x), f_i^2(x)$ are polynomials of degree at most t and $f_i^1(k) = r_i A_i(k) + f_{\beta_i}(k), f_i^2(k) = f_i^1(0)B_i(k) - f_{\beta_i B_i}(k) - r_i C_i(k)$ for all honest parties P_k. Furthermore, $f_i^2(0) = 0$. Hence $(r_i A_i(0) + f_{\beta_i}(0))B_i(0) = f_{\beta_i B_i}(0) + r_i C_i(0)$. Therefore $r_i(A_i(0)B_i(0) - C_i(0)) = f_{\beta_i B_i}(0) - f_{\beta_i}(0)B_i(0)$. Note that r_i is revealed to the adversary only after the values of $f_{\beta_i B_i}(0)$ and $f_{\beta_i}(0)$ are fixed. Since r_i is randomly generated, the equality will hold only with negligible probability if $A_i(0)B_i(0) \neq C_i(0)$. ■

The round complexity of the above protocol is $(21, 5)$: the round complexities of both step 1 and step 2 are $(8, 1)$; step 3 requires 1 round of interaction; each of steps 4 to 6 require 1 round of broadcast; step 7 requires 1 round of interaction.

Reducing the round complexity: There are a number of modifications we can do to reduce the round complexity.

Modification 1. As in Section 4.2.2, we can apply the technique from [FGG$^+$06] to reduce the round complexities of step 1 and step 2 from $(16, 2)$ to $(9, 2)$: suppose P_i is required to share a value x in step 2 of the original protocol. Instead, P_i shares a random value r in step 1 and broadcasts $x - r$ in step 2.

Modification 2. We can save one round of broadcast by modifying steps 4 to 7 of the protocol as follows. The main idea is that we postpone the usage of broadcast in step 4 to step 6 by using a (regular) signature scheme.

Step 4 P_i sends $f_i^1(x) = r_iA_i(x) + f_{\beta_i}(x)$ and $f_i^2(x) = f_i^1(0)B_i(x) - f_{\beta_iB_i}(x) - r_iC_i(x)$ to all parties, along with a signature on each polynomial.

Step 5 P_k broadcasts a complaint, as well as the signatures of P_i on $f_{\beta_i}(k)$, $f_{\beta_iB_i}(k)$ and $C_i(k)$ (if these values have not been made public), if one of the following conditions hold:

- P_k does not receive the polynomials $f_i^1(x)$ or $f_i^2(x)$ (with valid signatures).
- $f_i^1(k) \neq r_iA_i(k) + f_{\beta_i}(k)$ or $f_i^2(k) \neq f_i^1(0)B_i(k) - f_{\beta_iB_i}(k) - r_iC_i(k)$.

Step 6

- P_k broadcasts the polynomials (along with signatures from P_i) it received in step 4.
- If P_k broadcasts a complaint, then for all $1 \leq j \leq n$, P_i broadcasts $F_{a_j^i}(k)$, $F_{b_j^i}(k)$ and the signatures of P_k on each of the values (assuming the corresponding value has not been made public).

Step 7 P_i is disqualified if one of the following conditions hold:

- Two different polynomials (with valid signatures from P_i) for $f_i^1(x)$ or $f_i^2(x)$ were broadcast in step 6.
- P_k broadcasts a complaint in step 6 but P_i does not respond to it or P_i fails to provide the signatures of P_k on the required values or $f_i^1(k) \neq r_i \sum_j F_{a_j^i}(k) + f_{\beta_i}(k)$ or $f_i^2(k) \neq f_i^1(0) \sum_j F_{b_j^i}(k) - f_{\beta_iB_i}(k) - r_iC_i(k)$ or $f_i^2(0) \neq 0$.

If P_i is disqualified, then the parties reconstruct the values of A_i and B_i (note that A_i and B_i have been 2-level shared).

Modification 3. After modification 2, two rounds of broadcasts are invoked in steps 5 and 6 in the following manner:

- In step 5, if certain conditions hold, then P_k broadcasts a message x.
- In step 6, if P_k broadcasts a message y in step 5, then P_i broadcasts a message z.

To reduce the number of rounds of broadcasts from two to one, we can apply the same technique that was used in Lemma 2.2.5 and Section 4.3.3.1, at the expense of an additional round of interaction. We modify steps 5–7 as follows:

Step 5

Sub-step a If one of the following conditions hold: (i) P_k does not receive the polynomials $f_i^1(x)$ or $f_i^2(x)$ (with valid signatures). (ii) $f_i^1(k) \neq r_iA_i(k) + f_{\beta_i}(k)$ or $f_i^2(k) \neq f_i^1(0)B_i(k) - f_{\beta_iB_i}(k) - r_iC_i(k)$, then P_k sends and signs the following to all parties: "P_k complains P_i", and the signatures of P_i on $f_{\beta_i}(k)$, $f_{\beta_iB_i}(k)$ and $C_i(k)$ (if the values have not been made public).

Sub-step b A party forwards all the messages it received in step a to P_i.

Step 6 • P_k broadcasts the polynomials (along with signatures from P_i) it received in step 4, as well as all the messages it received in step 5a.

- If P_i receives a message "P_k complains P_i" (with valid signature from P_k) in step 5b, then for all $1 \leq j \leq n$, P_i broadcasts $F_{a_j^i}(k)$, $F_{b_j^i}(k)$ and the signatures of P_k on each of the values (assuming the corresponding value has not been made public).

Step 7 P_i is disqualified if one of the following conditions hold:

- Two different polynomials (with valid signatures from P_i) for $f_i^1(x)$ or $f_i^2(x)$ were broadcast in step 6.
- In step 6, at least $t+1$ parties broadcast "P_k complains P_i", as well as the signatures of P_i on $f_{\beta_i}(k)$, $f_{\beta_i B_i}(k)$ and $C_i(k)$ (if the values have not been made public),

  and

  P_i does not respond to the complaint in step 6, or P_i fails to provide the signatures of P_k on the required values, or $f_i^1(k) \neq r_i \sum_j F_{a_j^i}(k) + f_{\beta_i}(k)$, or $f_i^2(k) \neq f_i^1(0) \sum_j F_{b_j^i}(k) - f_{\beta_i B_i}(k) - r_i C_i(k)$, or $f_i^2(0) \neq 0$.

If P_i is disqualified, then the parties reconstruct the values of A_i and B_i (note that A_i and B_i have been 2-level shared).

After the above three modifications, step 1 and step 2 now require 9 rounds (including 2 rounds of broadcast) in total; steps 3 and 4 require 1 round each, step 5 requires 2 rounds, step 6 requires 1 round of broadcast and step 7 requires 1 round of interaction. Thus, we have the following:

Lemma 4.3.8 *There exists an authenticated protocol with round complexity* $(15, 3)$ *tolerating* $t < n/2$ *malicious parties that generates random multiplication triples.*

4.3.5 Constant-Round MPC Protocols using One Round of Broadcast

By applying the same transformation as described in Section 4.2.3.1 to the random multiplication triples generation protocol, and using the authenticated gradecast protocol in Lemma 2.2.2 , we have a protocol implementing the setup phase with round complexity $(29, 1)$. Given this result, following the discussion in Section 4.2.3.2, we obtain the following theorem:

Theorem 4.3.9 *Assuming the existence of one-way functions and public-key infrastructure, there exists a secure multiparty computation protocol tolerating* $t < n/2$ *malicious parties with round complexity* $(34, 1)$*.*

Following Corollary 3.2.4, this immediately gives a MPC protocol with (expected) round complexity $(67, 0)$. However, we can do better. The protocol in Theorem 4.3.9 does not use broadcast until the 28th round. In Section 3.2.2, we have made the observation that some components of our broadcast protocol (in particular, the first phase of an OLE protocol) can be carried out in advance even before the broadcast values are known. This observation allows us to save 5 rounds of interaction. Thus, we have the following corollary:

Corollary 4.3.10 *Assuming the existence of one-way functions and public-key infrastructure, there exists a secure multiparty computation protocol tolerating* $t < n/2$

malicious parties with expected round complexity 62.

Part II

Feasibility of Broadcast in Radio Networks

Chapter 5

Feasibility of Broadcast in Radio Networks

In this chapter, we study the feasibility of broadcast in radio networks in the presence of malicious parties. We describe our network and adversarial model and give the necessary definitions in Section 5.1. We give a protocol in Section 5.2 that can achieve broadcast if the adversary cannot corrupt more than a certain number of parties in any neighborhood. We show the bound is tight in Section 5.3 in the sense that broadcast is impossible if the number of corrupted parties in some neighborhood is greater than that number.

5.1 Preliminaries

We recall the requirements of the broadcast problem. There is a distinguished party P_d known as the *dealer* that holds an initial message $\mathcal{M}$. A protocol is said to achieve *broadcast* if the following conditions hold:

1. All (honest) parties eventually output a common value v.
2. If the dealer is honest, then $v = \mathcal{M}$.

We consider the network model where parties are located on an infinite grid (each grid unit is a 1×1 square). A party can be uniquely identified by its grid location (x, y).

Assuming absence of *collisions*, if a party *multicasts* a message m, then all parties within distance r (in an appropriate metric) will receive the message. This distance r is known as the *transmission radius.* The set of parties within this radius is termed as the *neighborhood* of (x, y) and is denoted as $\mathsf{nbd}(x, y)$. Parties in $\mathsf{nbd}(x, y)$ are known as the *neighbors* of (x, y). (x, y) is considered to be in $\mathsf{nbd}(x, y)$. For convenience, we denote by $\mathsf{nbd}_2(x, y)$ the set of parties that are at most two hops away from (x, y), i.e.,

$$\mathsf{nbd}_2(x, y) \stackrel{\text{def}}{=} \{(x_2, y_2) : \exists (x_1, y_1)\ s.t.\ (x_2, y_2) \in \mathsf{nbd}(x_1, y_1) \wedge (x_1, y_1) \in \mathsf{nbd}(x, y)\}.$$

We let $\tilde{r} \stackrel{\text{def}}{=} \lfloor r \rfloor$, that is $\tilde{r}$ is the truncation of r in case r is not an integer.

Message Collision: When two parties P_i and P_j multicast at the same time, a message collision occurs at the parties in $\mathsf{nbd}(P_i) \cap \mathsf{nbd}(P_j)$. If parties are equipped with *collision detectors*, then parties in $\mathsf{nbd}(P_i) \cap \mathsf{nbd}(P_j)$ detect that a message collision has occurred and can substitute default messages instead. In the absence of a collision detector, there is no guarantee on what parties in $\mathsf{nbd}(P_i) \cap \mathsf{nbd}(P_j)$ receive.

We primarily present results in the L_∞ metric, where the distance between points (x_1, y_1) and (x_2, y_2) is given by $\max\{|x_1 - x_2|, |y_1 - y_2|\}$. Note that there is a total number of $(2\tilde{r} + 1)^2$ parties in a neighborhood. However, our protocols are also applicable in the L_2 (also known as the "Euclidean") metric, where the distance between points as before is given by $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. This issue will be briefly discussed in the corresponding section.

We consider a locally bounded adversarial model, i.e., the adversary is allowed to corrupt parties as long as no single neighborhood contains more than t corrupted parties. We assume there exists a pre-determined time division multiple access (TDMA) schedule such that if all parties follow the schedule, then no collision will occur. However, a corrupted party is allowed to deviate from the schedule and cause *message collision* and *spoof* the identity of another party for a bounded number of times (of course, we assume an honest party always follows the schedule). Let n_c and n_s be the corresponding bounds on the number of message collisions and address spoofing respectively that a corrupted party can perform. Both n_c and n_s are assumed to be known in advance by all parties.

Identity Spoofing: A corrupted party P_i is able to spoof an honest party P_j when it is the turn of P_j to multicast a message (according to the underlying TDMA schedule) but P_j has no message to send (according to the prescribed protocol). In this scenario, P_i can impersonate P_j by multicasting a message m with the sender identity falsely set to P_j. If this happens, then the parties in $\mathsf{nbd}(P_i) \cap \mathsf{nbd}(P_j)$ receive m and treat m as originating from P_j.

We remark that the problem of identity spoofing can be reduced to message collision by having P_j always multicasts something (e.g., a fixed dummy message) when it is its turn instead of remaining silent. However, this approach is communication inefficient, and our solution takes a different approach.

5.2 Feasibility Result

In this section, we present a protocol that can achieve broadcast for $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$ in the L_∞ metric, even if a corrupted party can cause n_c message collisions and n_s spoofed address. Our solutions use some known results in the literature; we we review them in Section 5.2.1. We describe our protocol in Section 5.2.2.

In the descriptions of protocols that follow, when we say a party multicasts a message m, we actually mean the party multicasts m in its next available turn (according to the TDMA schedule).

5.2.1 Tools

Our solutions use some known results in the literature, reviewed below.

Broadcast in Point-to-Point Networks We only review the essentials for our solution. Please refer to Section 1.1.1 for a more detailed review. Consider a fully connected point-to-point network where there is an authenticated channel connecting each pair of parties. An adversary cannot modify the messages sent between honest parties, but it can observe the messages. An execution of a *synchronous* protocol takes place in a sequence of rounds. In each round, parties send messages to each other depending on the messages they have received in the previous rounds. An adversary is said to be *rushing* if it can see the messages sent to corrupted parties in the current round before it decides the outgoing messages of faulty parties for that round. Let n be the total number of parties. The following result is well-known (see, e.g. [PSL80, GM98]):

Lemma 5.2.1 *There exists a synchronous protocol $\mathcal{B}_{p2p}$ that achieves broadcast in a fully connected point-to-point network in the presence of a rushing adversary corrupting $t < \frac{1}{3}n$ parties. $\mathcal{B}_{p2p}$ terminates in a fixed number of rounds and has the following property: within the same round, an honest party always sends the same message to all other parties.*

Broadcast in Radio Networks without Collisions or Identity Spoofing We review the broadcast algorithm $\mathcal{B}_{\textsf{no collision}}$ described in [BV05b] that achieves broadcast if $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$, assuming the L_∞ metric, no collisions, and no address spoofing.

1. **(Broadcast in $\textsf{nbd}(P_d)$):** The dealer P_d multicasts the message $\mathcal{M}$. Each neighbor P_i of P_d outputs the first value it receives from P_d and then multicasts a COMMITTED$(P_i, \mathcal{M})$ message.

2. **(Broadcast in the rest of the network):** Every party P_j (including the dealer and the neighbors of the dealer) follows the procedure below:

   - On receipt of a COMMITTED(P_i, v) message from neighbor P_i, record the message and multicast a HEARD(P_j, P_i, v) message.

   - On receipt of a HEARD(P_i, P_k, v) message from neighbor P_i, record the message (but do not propagate it further).

   - *(Output determination):* All P_j that are not neighbors of P_d continually check the following: if there exists a party P_q, a value v, and P_j has recorded $t+1$ messages $m_1, m_2, \ldots, m_{t+1}$ such that (i) for $1 \leq i \leq t+1$, message m_i is of the form COMMITTED(P_{a_i}, v) or HEARD$(P_{a_i}, P_{a'_i}, v)$; and (ii) $P_{a_1}, \ldots, P_{a'_1}, \ldots$ are all distinct neighbors of P_q, then P_j outputs the value v.

The following is implicit in the proof of [BV05b, Theorem 2]:

Claim 5.2.2 *Assuming the L_∞ metric, no collisions, no address spoofing, and $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$, $\mathcal{B}_{\textsf{no collision}}$ achieves broadcast. In addition, there exists a constant T (dependent on $t, \tilde{r}$) such that if the parties start executing $\mathcal{B}_{\textsf{no collision}}$ at time 0, all honest parties in $\textsf{nbd}_2(P_d)$ output $\mathcal{M}$ by time T.*

Moreover, the above holds even the adversary is given the following extra power: when a corrupted party $P_i \neq P_d$ performs a multicast, the neighbors of P_i can receive different *messages, subject to the choice of the adversary.*

5.2.2 Our Broadcast Protocol

Following Claim 5.2.2, broadcast can be achieved in the presence of an adversary that can cause message collisions and spoofed address if we obtain a protocol based on $\mathcal{B}_{\textsf{no collision}}$ such that:

1. **(Broadcast in $\textsf{nbd}(P_d)$):** In step 1, neighbors of the dealer agree on a common message (as the message originated from the dealer) before they propagate it to other parties in the network.

2. **(Broadcast in the rest of the network):** In step 2, whenever an honest party multicasts a message m, its neighbors are able to receive m correctly

despite the possible occurrence of collisions caused by corrupted parties in the vicinity. If the adversary can send spoofed address, then a party will accept a message m from a neighbor only if it is convinced that m originated from that neighbor.

Condition (1), above, is required to handle situations where a corrupted dealer can collude with other corrupted parties, and use collisions to send conflicting values to different neighbors. An example is as follows: a corrupted dealer multicasts two inconsistent messages; a corrupted party on its left causes a message collision the first time; a corrupted party on its right causes a message collision the second time. Now parties on the left will get a message different from those parties on the right. To this effect, we develop an agreement protocol among parties in $\mathsf{nbd}(P_d)$. We use a primitive called *weak broadcast* as a building block in this agreement protocol. Weak broadcast is defined as follows:

Definition 15 (Weak Broadcast) A party P_i performs a **weak broadcast** of a message m to a set of parties $\mathcal{S}$ within time T if the following conditions hold:

1. An honest party $P_j \in \mathcal{S}$ outputs a message m_j within time T.

2. If P_i is honest, then $m_j = m$ for all honest parties $P_j \in \mathcal{S}$.

Note that if P_i is corrupted, then two honest parties may output two different messages.

In Section 5.2.2.2, we show how to construct protocols for weak broadcast and, subsequently, broadcast. But first, in Section 5.2.2.1, we will show how to achieve agreement among the neighbors of the dealer, assuming each party in $\mathsf{nbd}(P_d)$ is capable of performing a weak broadcast to all other parties in $\mathsf{nbd}(P_d)$ within time T.

5.2.2.1 Agreement among Neighbors of the Dealer

We transform the broadcast protocol $\mathcal{B}_{p2p}$ (cf. Lemma 5.2.1) for $n = |\mathsf{nbd}(P_d)|$ parties and working in the point-to-point model to a broadcast protocol $\mathcal{B}_{\mathsf{nbd}(P_d)}$ for the set $\mathsf{nbd}(P_d)$ and working in the radio network model.

$\mathcal{B}_{\mathsf{nbd}(P_d)}$ simulates $\mathcal{B}_{p2p}$ round by round. Suppose in a given round of $\mathcal{B}_{p2p}$, party P_i is instructed to send the message m_i to other parties. To simulate one round of execution in $\mathcal{B}_{p2p}$, the parties run the following subroutine sequentially:

> for each party $P_i \in \mathsf{nbd}(P_d)$, party P_i does a weak broadcast of the message m_i to all parties in $\mathsf{nbd}(P_d)$.

Note that the weak broadcast may be viewed as establishing a *virtual* point-to-point link between pairs of parties in $\mathsf{nbd}(P_d)$. Thus, it is ensured that if P_i is honest, all other parties receive the same value from P_i. If P_i is corrupted, receipt of conflicting values is acceptable, as P_i is capable of sending different values to different parties in the point-to-point model (cf. Claim 5.2.2).

Finally, party P_i outputs whatever it is directed to output by $\mathcal{B}_{p2p}$. We note that if the round complexity of $\mathcal{B}_{p2p}$ is R, then $\mathcal{B}_{\mathsf{nbd}(P_d)}$ takes time $RT|\mathsf{nbd}(P_d)|$.

Lemma 5.2.3 *If $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$, then $\mathcal{B}_{\mathsf{nbd}(P_d)}$ ensures that all neighbors of the dealer output the same message m'. If the dealer is honest, then $m' = \mathcal{M}$.*

Proof $n = |\mathsf{nbd}(P_d)| = (2\tilde{r}+1)(2\tilde{r}+1)$. If $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$, then $t < \frac{1}{4}|\mathsf{nbd}(P_d)|$. The lemma then follows from the fact that $\mathcal{B}_{p2p}$ can tolerate a rushing adversary corrupting fewer than $\frac{1}{3}n$ parties. ■

5.2.2.2 Weak Broadcast and Broadcast

Depending on different assumptions (i.e., whether corrupted parties are allowed to spoof the identity of other parties, whether honest parties are equipped with collision detectors, etc.), we show how to obtain a weak broadcast protocol and then a protocol for broadcast in the entire network. The most general case is that corrupted parties are allowed to do address spoofing and parties are not equipped with collision detectors. However, we provide constructions for other cases to serve as a warmup.

(i) No Address Spoofing; Collision Detectors

Here we assume $n_s = 0$. An honest party may fail to receive a message from another honest party due to message collision; however, this can happen at most tn_c number of times. We observe that if an honest party P_i multicasts a message m for a total of $tn_c + 1$ times, then any neighbor of P_i will receive at least one copy of m successfully.

Based on the protocol $\mathcal{B}_{\textsf{no collision}}$, we construct a protocol $\mathcal{B}_{repeat}$ where a party P_i will execute the same instructions as in $\mathcal{B}_{\textsf{no collision}}$ except that:

- If P_i is instructed to multicast a message m in $\mathcal{B}_{\textsf{no collision}}$, then P_i multicasts the message m for a total of $tn_c + 1$ times.

- If P_i is instructed to carry out an action after receipt of a message m from P_j in $\mathcal{B}_{\textsf{no collision}}$, then P_i carries out the corresponding action only when it receives m from P_j for the first time.

$\mathcal{B}_{repeat}$ will be used as a building block for our weak broadcast protocol. For the sake of completeness, we include the protocol description for $\mathcal{B}_{repeat}$:

1. The dealer $P_{d'}$ multicasts the message $\mathcal{M}$ for a total of $tn_c + 1$ times. Each neighbor P_i of $P_{d'}$ outputs the first value v it heard from $P_{d'}$.

2. If P_i is a neighbor of $P_{d'}$ and it outputs a value v, then it multicasts the message COMMITTED(P_i, v) for a total of $tn_c + 1$ times.

3. Every party P_j (including the dealer and the neighbors of the dealer) follows the procedure below:

    - On receipt of a COMMITTED(P_i, v) message from a neighbor P_i for the first time, record the message and multicast HEARD(P_j, P_i, v) for a total of $tn_c + 1$ times.

    - On receipt of a HEARD(P_i, P_k, v) message from a neighbor P_i for the first time, record the message (but do not re-propagate).

    - Output the value v and multicast COMMITTED(P_j, v) $tn_c + 1$ times if: not already committed to a value, and there exists a party P_q and $t + 1$ recorded messages $m_1, m_2, \ldots, m_{t+1}$ such that (1) for all i, either $m_i =$ COMMITTED(P_{a_i}, v) or m_i=HEARD$(P_{a_i}, P_{a_{i'}}, v)$, and (2) $\{P_{a_i}, P_{a_{i'}}\}$ are all distinct neighbors of P_q.

The dealer $P_{d'}$ mentioned above is the dealer in the protocol $\mathcal{B}_{repeat}$ (used as a building block), and is not to be confused with the dealer of the overall broadcast. As can be seen, $\mathcal{B}_{repeat}$ primarily differs from $\mathcal{B}_{\mathsf{no\ collision}}$ in that messages are repeated sufficiently-many times so that they will eventually be received even if there are collisions. We have:

Lemma 5.2.4 *Assume the L_∞ metric, and $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$. Then there exists a constant T (depending on r) such that the following holds: If the dealer $P_{d'}$ in $\mathcal{B}_{repeat}$ is honest and all parties execute $\mathcal{B}_{repeat}$ for time T, then all honest parties in $\mathsf{nbd}_2(P_{d'})$ will output m.*

Proof This follows from Claim 5.2.2. ■

Achieving Weak Broadcast Note that in $\mathcal{B}_{repeat}$, if the party $P_{d'}$ is corrupted then an honest party may not output a value. However, it is easy to modify $\mathcal{B}_{repeat}$ to obtain a weak broadcast protocol.

Lemma 5.2.5 *Assume the L_∞ metric, and $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$. Then for any party $P_{d'}$ there exists a protocol $\mathcal{B}_{\mathsf{weak\ broadcast}}$ that allows a party $P_i \in \mathsf{nbd}(P_{d'})$ (which can be potentially corrupted) to perform a weak broadcast to $\mathsf{nbd}(P_{d'})$ within time T.*

Proof In $\mathcal{B}_{\mathsf{weak\ broadcast}}$ (with P_i as dealer of the weak broadcast), parties execute $\mathcal{B}_{repeat}$ for a period of time T. After time T, if a party has not yet been able to output a value, then a party outputs a default value. Note that $\mathsf{nbd}(P_d) \subseteq \bigcup_{i \in \mathsf{nbd}(P_d)} \mathsf{nbd}_2(i)$. The lemma then follows from Lemma 5.2.4. ■

Achieving Broadcast Following Lemma 5.2.5, every party in $\mathsf{nbd}(P_d)$ can perform a weak broadcast to $\mathsf{nbd}(P_d)$. Thus, we have the primitive required to run protocol

$\mathcal{B}_{\mathsf{nbd}(P_d)}$. We can now obtain a broadcast protocol resilient to a bounded number of collisions. This protocol $\mathcal{B}_{reliable\ broadcast}$ is a modified version of $\mathcal{B}_{repeat}$, where the first step of $\mathcal{B}_{repeat}$ is changed to the following:

> The parties execute the protocol $\mathcal{B}_{\mathsf{nbd}(P_d)}$ with the dealer P_d using the message $\mathcal{M}$ as the input. Let m_i be the output of party P_i in $\mathcal{B}_{\mathsf{nbd}(P_d)}$. Each neighbor P_i of P_d outputs m_i.

With this change, we obtain a protocol achieving broadcast.

The above protocol can also be used in the absence of a collision detector, and in the presence of address spoofing, after minor modifications. We discuss various scenarios below.

(ii) No Address Spoofing; No Collision Detectors

The construction is similar to case (i) except that in the transformation of $\mathcal{B}_{\mathsf{no\ collision}}$ into $\mathcal{B}_{repeat}$:

- If a party P_i is instructed to multicast a message m in $\mathcal{B}_{\mathsf{no\ collision}}$, then in $\mathcal{B}_{repeat}$ P_i multicasts the message m for a total of $2tn_c + 1$ times.

- If a party P_i is instructed to carry out an action after receipt of a message m from P_j in $\mathcal{B}_{\mathsf{no\ collision}}$, then in $\mathcal{B}_{repeat}$ party P_i carries out the corresponding action only when it receives $tn_c + 1$ copies of m from P_j.

Note that if an honest party P_i multicasts a message m for a total of $2tn_c + 1$ times, then a neighbor of P_i will receive at least $tn_c + 1$ legitimate copies of m. Now, if a party P_j receives $tn_c + 1$ copies of m from P_i, then P_j can conclude that m has not been corrupted due to message collisions.

(iii) Address Spoofing; Collision Detectors

In the transformation of $\mathcal{B}_{\textsf{no collision}}$ into $\mathcal{B}_{repeat}$, we now do the following:

- If a party P_i is instructed to multicast a message m in $\mathcal{B}_{\textsf{no collision}}$, then in $\mathcal{B}_{repeat}$ party P_i multicasts the message m for a total of $t(n_c + n_s) + 1$ times.

- If a party P_i is instructed to carry out an action after receipt of a message m from P_j in $\mathcal{B}_{\textsf{no collision}}$, then in $\mathcal{B}_{repeat}$ party P_i carries out the corresponding action only when it receives $tn_s + 1$ copies of m from P_j.

If an honest party P_i multicasts a message m for a total of $t(n_c+n_s)+1$ times, then a neighbor of P_i will receive at least $tn_s + 1$ legitimate copies of m. On the other hand, if a party P_j receives $tn_s + 1$ copies of m claimed to be originated from P_i, then P_j can conclude that m indeed originated from P_i.

(iv) Address Spoofing; No Collision Detectors

In the transformation of $\mathcal{B}_{\textsf{no collision}}$ into $\mathcal{B}_{repeat}$, we now do the following:

- If a party P_i is instructed to multicast a message m in $\mathcal{B}_{\textsf{no collision}}$, then in $\mathcal{B}_{repeat}$ party P_i multicasts the message m for a total of $t(2n_c + n_s) + 1$ times.

- If a party P_i is instructed to carry out an action after receipt of a message m from P_j in $\mathcal{B}_{\textsf{no collision}}$, then in $\mathcal{B}_{repeat}$ party P_i carries out the corresponding action only when it receives $t(n_c + n_s) + 1$ copies of m from P_j.

If an honest party P_i multicasts a message m for a total of $t(2n_c + n_s) + 1$ times, then a neighbor of P_i will receive at least $t(n_c + n_s) + 1$ legitimate copies of

m. On the other hand, if a party P_j receives $t(n_c + n_s) + 1$ copies of m claimed to be originated from P_i, then P_j can conclude that m indeed originated from P_i.

Thus, we obtain the following theorem:

Theorem 5.2.6 *In the L_∞ metric, if $t < \frac{1}{2}\tilde{r}(2\tilde{r}+1)$ then there exists a protocol that achieves broadcast as long as there is a bound on the number of collisions caused and spoofed messages sent by each corrupted party.*

In fact, an analogue of Claim 5.2.2 exists for the L_2 metric (due to [BV05b, Section VIII.]):

Claim 5.2.7 *Assuming the L_2 metric, no collisions, no address spoofing, and $t < 0.23\pi\tilde{r}^2$, $\mathcal{B}_{\textsf{no collision}}$ achieves broadcast. In addition, there exists a constant T (dependent on $t, \tilde{r}$) such that if the parties start executing $\mathcal{B}_{\textsf{no collision}}$ at time 0, all honest parties in $\mathsf{nbd}_2(P_d)$ output $\mathcal{M}$ by time T.*

Moreover, the above holds even the adversary is given the following extra power: when a faulty party $P_i \neq P_s$ performs a multicast, the neighbors of P_i can receive different *messages, subject to the choice of the adversary.*

Thus, applying the same transformation as outlined before, we obtain that:

Theorem 5.2.8 *In the L_2 metric, if $t < 0.23\pi\tilde{r}^2$ then there exists a protocol that achieves broadcast as long as there is a bound on the number of collisions caused and spoofed messages sent by each corrupted party.*

5.3 Impossibility Result

In this section, we present the lower bound result on the value of t when it is impossible to achieve broadcast. The idea is to have the adversary corrupt a set of parties that partitions the square grid into two halves, and act in a way such that an honest party in one half cannot learn the messages broadcasted by a dealer in the other half. Without loss of generality, we assume that the dealer is at $(0, 0)$. We remark that our impossibility result holds even if a corrupted party cannot cause message collisions nor carry out message spoofing.

Theorem 5.3.1 *If $t \geq \frac{1}{2}\tilde{r}(2\tilde{r}+1)$, it is impossible to achieve broadcast in L_∞ metric, even if a corrupted party cannot cause message collision nor carry out message spoofing.*

Proof We define two sets of parties $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows:

- If $\tilde{r}$ is even, then

$$\begin{aligned}\mathcal{S}_1 &\overset{\text{def}}{=} \{(x, y) : 1 \leq x \leq \tilde{r} \wedge x \text{ is odd}\}\\ \mathcal{S}_2 &\overset{\text{def}}{=} \{(x, y) : 1 \leq x \leq \tilde{r} \wedge x \text{ is even}\}\end{aligned}$$

- If $\tilde{r}$ is odd, then

$$\begin{aligned}\mathcal{S}_1 &\overset{\text{def}}{=} \{(x, y) : (1 \leq x < \tilde{r} \wedge x \text{ is odd})\\ &\qquad \vee(x = \tilde{r} \wedge y \text{ is odd})\}\\ \mathcal{S}_2 &\overset{\text{def}}{=} \{(x, y) : (1 \leq x < \tilde{r} \wedge x \text{ is even})\\ &\qquad \vee(x = \tilde{r} \wedge y \text{ is even})\}\end{aligned}$$

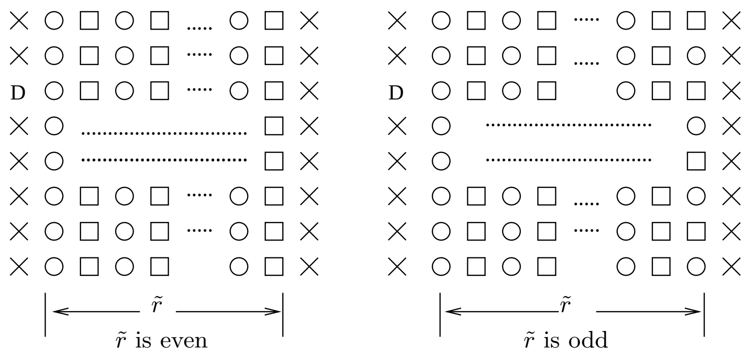

Figure 5.1: Lower Bound in L_∞ metric

Refer to Figure 5.1 for a pictorial representation of $\mathcal{S}_1$ and $\mathcal{S}_2$.

Note that no honest party $P(x, y)$ (for $x > r$) can distinguish the following two scenarios:

- Scenario 1:
  - The dealer broadcasts a message m.
  - The adversary corrupts all parties in $\mathcal{S}_1$ and all corrupted parties act as if the dealer broadcasted the message $\bar{m}$.
  - All parties in $\mathcal{S}_2$ are honest.
- Scenario 2:
  - The dealer broadcasts a message $\bar{m}$.
  - The adversary corrupts all parties in $\mathcal{S}_2$ and all corrupted parties act as if the dealer broadcasted the message m.
  - All parties in $\mathcal{S}_1$ are honest.

The only thing left is to argue the adversary has corrupted at most $\frac{1}{2}\tilde{r}(2\tilde{r}+1)$ neighbors of any honest party in both scenarios. But this is true since any party not in $\mathcal{S}_1(\mathcal{S}_2)$ has at most $\frac{1}{2}\tilde{r}(2\tilde{r}+1)$ neighbors in $\mathcal{S}_1(\mathcal{S}_2)$.

■

Chapter 6

Conclusions

In this dissertation, we have constructed round-efficient broadcast and secure multiparty computation protocols in the point-to-point network and studied the feasibility of broadcast in radio networks.

We have argued that if the ultimate goal is to optimize round complexity for point-to-point networks, 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. Towards this end, we have constructed constant-round secure multiparty computation protocols that use only a single round of broadcast. The key to the constructions is a new primitive that we introduce – *moderated* protocols. This new primitive also allows us to construct the first expected constant-round authenticated broadcast protocol for honest majority without any additional assumption.

We have initiated the study of broadcast in radio networks in the presence of adversarial faults. We have purposed an adversarial model to model the corruptions of parties in radio networks. Feasibility and impossibility results are present for the L_1-metric, and these results are tight under our adversarial model.

Some problems are left open by this dissertation, and we discuss these below:

Expected Constant-Round Broadcast Protocol for Dishonest Majority

We have shown the existence of an authenticated broadcast protocol for honest majority. However, it is of theoretical interest to determine if it is possible to achieve broadcast in an expected constant number of rounds in the presence of dishonest majority (i.e., more than half of the parties can be corrupted by the adversary). Recently, some progress has been made towards resolving this question [GKKO], but a complete answer is yet to be obtained.

Constant-Round Secure Multiparty Computation Protocol for Honest Majority Using One Round of Broadcast

In this dissertation, we have given a constant-round secure MPC protocol for honest majority using one round of broadcast assuming the existence of a public-key infrastructure (PKI). This immediately gives a constant-round secure MPC protocol using two rounds of broadcast *without* assuming a PKI (since a PKI can be setup using one round of broadcast). It will be interesting to see if the number of rounds of broadcast can be reduced from two to one.

Efficient Broadcast Protocol for Radio Networks

In our broadcast protocol, the communication overhead per each honest party grows as $\Omega(t(n_c + n_s))$ (recall that t is the maximum number of corrupted parties in a neighborhood, n_c and n_s being the maximum number of collisions and address spoofing caused by a corrupt party respectively). It is also to be noted that if the adversary performs the maximum number of disruptive actions permitted, the average cost of causing disruptions is $\Theta(n_c + n_s)$ per corrupted party. Thus, in our protocol, honest parties are required to send more messages than corrupted parties are assumed able to send!

One would desire a more communication-efficient broadcast protocol, or at least a protocol that requires honest parties to send out more messages only when the corrupted parties send out more messages. Our protocols do not achieve this since they are *proactive*, requiring parties to repeatedly send messages sufficiently-many times to overcome any collisions (or instances of address spoofing) that may occur. It would be of interest to determine whether a *reactive* protocol might perform better in the above regard.

Chapter A

Appendix

A.1 Multiparty computation protocol with round complexity $(O(1), O(1))$

In below, we sketch the protocol due to Damgård and Ishai[DI05] with emphasis on the part which requires interactions between parties. We refer the readers to [DI05] for a complete protocol description and explanation. We assume the function to be computed is described as a Boolean circuit. Let W be the total number of wires. And we number the wires from 0 to W-1.

Step 1 For each wire $w = 0, \ldots, W-1$,

- Party P_i shares a random bit λ_w^i by VSS. Let $\lambda_w = \sum \lambda_w^i \mod 2$. A party compute its share of λ_w.
- Party P_i shares 2 random keys s_{2w}^i, s_{2w+1}^i of length K by VSS (K being the security parameter). Again, the sharing is done in a bit by bit manner.

For each input bit b_w held by party P_j, P_j shares b_w by VSS.

Step 2 • For each input wire w and $i = 1, \ldots, n$, each server computes a share of the value $s_{2w+(b_w \oplus \lambda_w)}^i$. Note that the value can be written as the following polynomial:

$$((1 + b_w + \lambda_w) \mod 2)s^i_{2w} + ((b_w + \lambda_w) \mod 2)s^i_{2w+1}$$

If s^i_{2w} and s^i_{2w+1} are shared in a bit by bit manner, then computing the sharing of the value $s^i_{2w+(b_w \oplus \lambda_w)}$ can be reduced to computing the sharing of K degree-2 polynomials over $GF(2)$.

- For each wire w, the parties compute the shares of the value $b_w \oplus \lambda_w$.
- For each gate g in the circuit, suppose the two input wires are α and β, the output wire is γ and the corresponding operator is ?. For each $i = 1, \dots, n$, the parties compute shares of the values

$$a^{00,i}_g = s^i_{2\gamma+\delta^{00}_g} \quad ; \quad \delta^{00}_g = (\lambda_\alpha ? \lambda_\beta) \oplus \lambda_\gamma$$

$$a^{01,i}_g = s^i_{2\gamma+\delta^{01}_g} \quad ; \quad \delta^{01}_g = (\lambda_\alpha ? \bar{\lambda_\beta}) \oplus \lambda_\gamma$$

$$a^{10,i}_g = s^i_{2\gamma+\delta^{10}_g} \quad ; \quad \delta^{10}_g = (\bar{\lambda_\alpha} ? \lambda_\beta) \oplus \lambda_\gamma$$

$$a^{11,i}_g = s^i_{2\gamma+\delta^{11}_g} \quad ; \quad \delta^{11}_g = (\bar{\lambda_\alpha} ? \bar{\lambda_\beta}) \oplus \lambda_\gamma$$

Note that the values can be written as degree 3 polynomials in the already shared values.

Step 3 Let $a^{cd}_g = (a^{cd,1}_g, \dots, a^{cd,n}_g)$ for $c, d \in \{0, 1\}$. Define $A^{cd}_g = (a^{cd}_g, \delta^{cd}_g)$. Each party P_i computes a ciphertext (using the scheme described in [DI05]) based on (and only on) its share of A^{cd}_g and the keys $s^i_{2\alpha+c}$ and $s^i_{2\beta+d}$. Each party P_i sends the following items to the parties who are supposed to receive the outputs:

- The shares of the polynomials it obtained in step 2.
- The ciphertexts it computed in the current step.
- If P_j is entitled to receive the value of output wire w, then P_i sends its share of λ_w to P_j.

Step 4 P_i constructs its entitled output based on the information it received in the previous step.

Bibliography

[ASS$^+$03] S. Amitanand, I. Sanketh, K. Srinathant, V. Vinod, and C. P. Rangan. Distributed consensus in the presence of sectional faults. In *22nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 202–210, 2003.

[BCY89] G. Brassard, C. Crépeau, and M. Yung. Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds (extended abstract). In *Advances in Cryptology — Eurocrypt '89*, pages 192–195. Springer-Verlag, 1989.

[Bea89] D. Beaver. Multiparty protocols tolerating half faulty processors. In *Advances in Cryptology — Crypto '89*, pages 560–572. Springer-Verlag, 1989.

[Bea91a] D. Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology — Crypto '91*, pages 420–432. Springer-Verlag, 1991.

[Bea91b] D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[BG93] P. Berman and J. Garay. Cloture votes: $n/4$-resilient distributed consensus in $t + 1$ rounds. *Mathematical Systems Theory, special issue dedicated to fault-tolerant distributed algorithms*, 26(1):3–20, 1993.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology — Eurocrypt '92*, pages 307–323. Springer-Verlag, 1992.

[BB98] Z. Bar-Joseph and M. Ben-Or. A tight lower bound for randomized synchronous consensus. In *17th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 193–199, 1998.

[BJY97] M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology — Eurocrypt '97*, pages 280–305. Springer-Verlag, 1997.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *National Computer Conference*, volume 48, pages 313–317. AFIPS Press, 1979.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 503–513, 1990.

[B83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (Extended abstract). In *2nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 27–30, 1983.

[BE03] M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[Bra87] G. Bracha. An $O(\log n)$ expected rounds randomized Byzantine generals protocol. *Journal of ACM*, 34(4):910–920, 1987.

[BV05a] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In *24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 138–147, 2005.

[BV05b] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network: A simplified characterization. Technical Report, CSL, UIUC, May 2005. http://www.crhc.uiuc.edu/wireless/papers/bcast-addendum.pdf.

[BV07] V. Bhandari and N. H. Vaidya. Reliable broadcast in wireless networks with probabilistic failures. In *INFOCOM*, pages 715–723, 2007.

[CC85] B. Chor and B. Coan. A simple and efficient randomized Byzantine agreement algorithm. *IEEE Transaction on Software Engineering*, 11(6):531–539, 1985.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[CDD$^+$99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology — Eurocrypt '99*, pages 311–326. Springer-Verlag, 1999.

[CDG$^+$05] G. Chockler, M. Demirbas, S. Gilbert, C. Newport, and T. Nolte. Consensus and collision detectors in wireless Ad Hoc networks. In *24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 197–206, 2005.

[CFF$^+$05] J. Considine, M. Fitzi, M. Franklin, L. A. Levin, U. Maurer, and D. Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, July 2005.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648, 1996.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 383–395, 1985.

[CKPR01] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires $\Omega(\log n)$ rounds. In *33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 570–579, 2001.

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography (extended abstract). In *19th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 123–132, 2000.

[DI05] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology — Crypto '05*, pages 378–394. Springer-Verlag, 2005.

[DS83] D. Dolev and H.R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[DSS90] C. Dwork, D. Shmoys, and L. Stockmeyer. Flipping persuasively in constant time. *SIAM Journal on Computing*, 19(3):472–499, 1990.

[Fel88] P. Feldman. *Optimal Algorithms for Byzantine Agreement*. PhD thesis, Massachusetts Institute of Technology, 1988.

[FG03] 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 (PODC)*, pages 211–220, 2003.

[FGG$^+$06] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *Third Theory of Cryptography Conference (TCC)*, pages 329–342, 2006.

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

[FM85] P. Feldman and S. Micali. Byzantine agreement in constant expected time (and trusting no one). In *26th Annual Symposium on Foundations of Computer Science (FOCS)*, 1985.

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[FM00] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2000.

[FS89] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *Advances in Cryptology — Crypto '89*, pages 526–544. Springer-Verlag, 1989.

[GGN06] S. Gilbert, R. Guerraoui, and C. C. Newport. Of malicious motes and suspicious sensors: On the efficiency of malicious interference in wireless networks. In *10th International Conference on Principles of Distributed Systems (OPODIS)*, pages 215–229, 2006.

[GIKR01] 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 (STOC)*, pages 580–589, 2001.

[GIKR02] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology — Crypto '02*, pages 178–193. Springer-Verlag, 2002.

[GK96] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GKKO] J. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky. Round complexity of authenticated broadcast in the presence of a dishonest majority. Manuscript.

[GL05] S. Goldwasser and Y. Lindell. Secure computation without agreement. *Journal of Cryptology*, 18(3):247–287, 2005.

[GM98] 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.

[GRR98] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computation with applications to threshold cryptography. In *17th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 101–111, 1998.

[HT98] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *Advances in Cryptology — Crypto '98*, pages 408–423. Springer-Verlag, 1998.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, page 294, 2000.

[KBKV06] C.-Y. Koo, V. Bhandari, J. Katz, and N. H. Vaidya. Reliable broadcast in radio networks: the bounded collision case. In *25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 258–264, 2006.

[KK06] J. Katz and C.-Y. Koo. On expected constant-round protocols for Byzantine agreement. In *Advances in Cryptology — Crypto '06*, pages 445–462. Springer-Verlag, 2006.

[KK07] J. Katz and C.-Y. Koo. Round-efficient secure computation in point-to-point networks. In *Advances in Cryptology — Eurocrypt '07*, pages 311–328. Springer-Verlag, 2007.

[KKP01] E. Kranakis, D. Krizanc, and A. Pelc. Fault-tolerant broadcasting in radio networks. *Journal of Algorithms*, 39(1):47–67, 2001.

[KLR06] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118, 2006.

[KO04] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology — Crypto '04*, pages 335–354. Springer-Verlag, 2004.

[Koo04] C.-Y. Koo. Broadcast in radio networks tolerating Byzantine adversarial behavior. In *23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 275–282, 2004.

[KOS03] J. Katz, R. Ostrovsky, and A. Smith. Round efficiency of multi-party computation with a dishonest majority. In *Advances in Cryptology — Eurocrypt '03*, pages 578–595. Springer-Verlag, 2003.

[KY] A. Karlin and A. Yao. Probabilistic lower bounds for the Byzantine generals problem. Unpublished manuscript.

[Lin03] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.

[LLR02] Y. Lindell, A. Lysyanskaya, and T. Rabin. Sequential composition of protocols without simultaneous termination. In *21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 203–212, 2002.

[LLR06] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. *Journal of ACM*, 53(6):881–917, 2006.

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

[MR90] S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *Advances in Cryptology — Crypto '90*, pages 253–266. Springer-Verlag, 1990.

[MW94] Y. Moses and O. Waarts. Coordinated traversal: $(t+1)$-round Byzantine agreement in polynomial time. *Journal of Algorithms*, 17(1):110–156, 1994.

[Nie02] J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology — Crypto 2002*, pages 401–416. Springer-Verlag, 2002.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43, 1989.

[PP05a] A. Pelc and D. Peleg. Broadcasting with locally bounded byzantine faults. *Information Processing Letters*, 93(3):109–115, Feb 2005.

[PP05b] A. Pelc and D. Peleg. Feasibility and complexity of broadcasting with random transmission failures. In *24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 334–341, 2005.

[PRS02] M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 366–375, 2002.

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

[PW96] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.

[Rab83] M. O. Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–409, 1983.

[Rab94] T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *Journal of ACM*, 41(6):1089–1109, 1994.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–85, 1989.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[Ros00] A. Rosen. A note on the round-complexity of concurrent zero-knowledge. In *Advances in Cryptology — Crypto '00*, pages 451–468. Springer-Verlag, 2000.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite field. *SIAM Journal on Applied Mathematics*, 8(2):300–304, June 1960.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Tou84] S. Toueg. Randomized Byzantine agreements. In *3rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 163–178, 1984.

[Vai05] V. Vaikuntanathan. Brief announcement: broadcast in radio networks in the presence of byzantine adversaries. In *24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 167–167, 2005.

[Wai91] M. Waidner. *Byzantinische Verteilung ohne Kryptographische Annahmen trotz Beliebig Vieler Fehler (in German).* PhD thesis, University of Karlsruhe, 1991.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.