Diss. ETH No. 22213

Broadcast Amplification

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by

Pavel Raykov
Master of Science in Informatics
Università della Svizzera italiana

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. Stefan Dziembowski, co-examiner
Dr. Martin Hirt, co-examiner

2014

Acknowledgments

I would like to thank my advisors Martin Hirt and Ueli Maurer for their wise guidance and extensive support during my PhD studies. I would also like to thank Stefan Dziembowski for agreeing to serve as a co-examiner.

My gratitude also goes to all the current and former members of the Information Security and Cryptography Research Group at ETH Zurich that have been my colleagues during the studies. In particular, I would like to thank Christian Badertscher, Sandro Coretti and Christian Matt for proofreading my thesis.

Abstract

Byzantine broadcast (or simply broadcast) is a communication primitive that allows a designated party, the sender, to consistently distribute a value he holds among n parties, some of which—the Byzantine parties—may exhibit arbitrary malicious behavior. Broadcast guarantees that (1) the correct parties obtain the same value and (2) if the sender is correct, then this value is the one sent by him.

Broadcast is one of the most fundamental distributed primitives. It is employed in protocols for a variety of tasks, such as voting, bidding, collective contract signing, or multi-party computation.

In this thesis we study broadcast amplification, i.e., how to transform weak notions of broadcast into stronger ones. We consider two types of broadcast amplification: domain amplification and availability amplification.

In the domain-amplification setting, we study whether the ability to broadcast a small number of bits ℓ' can be amplified to broadcast of a larger number of bits $\ell > \ell'$. We give both feasibility and impossibility results, showing for which n, ℓ, ℓ' such amplification is possible. For example, we show that among $n = 3$ parties, the ability to broadcast $\ell' = 2$ bits can be amplified to broadcast of arbitrarily long messages (e.g., $\ell = 1$ gigabyte).

In the availability-amplification setting, we study whether the ability to broadcast can be prolonged in time, i.e., we study whether from temporarily available broadcast channels, one can create a setup that can later be used to implement broadcast. We show that the ability to broadcast once is already enough for a setup allowing to broadcast many times at a later point in time.

Zusammenfassung

Byzantinischer Broadcast (oder Broadcast) ist eine Kommunikationsprimitive, die es einer bestimmten Partei, dem Sender, erlaubt, einen Wert in seinem Besitz konsistent unter n Parteien zu verteilen. Einige der Parteien, die sog. byzantinischen Parteien, können dabei beliebiges Fehlverhalten aufweisen. Broadcast garantiert erstens, dass die korrekten Parteien stets den selben Wert erhalten, und zweitens, dass dieser Wert derjenige des Senders ist, wenn immer sich dieser korrekt verhält.

Broadcast ist eine der fundamentalsten verteilten Primitiven und findet in unzähligen Bereichen, wie beispielsweise Voting, Bidding, kollektiver Vertragsunterzeichung oder verteilter Berechnung (sog. Multi-Party Computation), Anwendung.

In dieser Dissertation analysieren wir Broadcastverstärkung, d.h., wir untersuchen, wie schwache Varianten von Broadcast in stärkere transformiert werden können. Wir betrachten zwei Arten von Broadcastverstärkung: Inputbereichverstärkung und Verfügbarkeitsverstärkung.

Bezüglich Inputbereichverstärkung analysieren wir, ob, ℓ'-bit Broadcast zu ℓ-bit Broadcast für $\ell > \ell'$ verstärkt werden kann. Wir liefern sowohl Machbarkeits- als auch Unmöglichkeitsresultate in Abhängigkeit von n, ℓ, ℓ'. So zeigen wir unter anderem, dass in einem Setting mit $n = 3$ Parteien 2-bit Broadcast zu Broadcast für beliebig lange Nachrichten verstärkt werden kann (z.B. $\ell = 1$ Gigabyte).

Im Bereich der Verfügbarkeitsverstärkung untersuchen wir, ob Broadcastfähigkeit verlängert werden kann, d.h., ob temporär verfügbare Broadcastkanäle zum Erstellen eines Setups, von dem aus später Broadcast implementiert werden kann, ausreichen. Wir zeigen, dass bereits mit einmaligem Broadcast ein Setup erreicht werden kann, das für beliebig viele spätere Broadcasts genügt.

Contents

1 Introduction **13**
1.1 Distributed Computation 13
1.1.1 Modeling Distributed Computation 14
1.2 Broadcast 17
1.2.1 Implementing Broadcast 17
1.2.2 Broadcast Amplification 20
1.3 Contributions 22
1.4 Outline 23

2 Tools **25**
2.1 General Tools 25
2.1.1 Collision-Resistant Hash Functions 25
2.1.2 Universal Hash Functions [CW79] 25
2.1.3 Identifying Predicates 26
2.1.4 Resolution Functions 27
2.2 Additional Distributed Primitives 28
2.2.1 Weak Broadcast 28
2.2.2 Graded Broadcast 29
2.2.3 Verifiable Secret Sharing 30

3 Domain Amplification **33**
3.1 Contributions 33
3.2 Domain-Amplification Setting 34
3.3 Impossibility Proofs of Domain Amplification 35
3.4 Existing Domain-Amplification Protocols 36
3.5 Single-Sender Model 37
3.5.1 Domain Amplification for **3** Parties 38
3.5.2 Lower Bounds in the Single-Sender Model 40
3.5.3 Summary 43

3.6 Multi-Sender Model 43
3.6.1 Domain Amplification for $\mathbf{n}$ Parties 43
3.6.2 A Protocol with Polynomial Communication 48
3.6.3 Lower Bounds in the Multi-Sender Model 50
3.6.4 Summary 51
3.7 On The Round Complexity of Domain-Amplification Protocols 52
3.8 Impossibility Proofs for Dynamic Usage Pattern 54
3.8.1 Single-Sender Model 54
3.8.2 Multi-Sender Model 54
3.8.3 Round Complexity Theorem 59

4 Communication Optimal Multi-Valued Broadcast for $\mathbf{t} < \mathbf{n}$ 61
4.1 Protocols Overview 62
4.2 Cryptographically Secure Construction 62
4.2.1 Block Broadcast Protocol CryptoBlockBC 63
4.2.2 Constructing Broadcast for Long Messages 65
4.3 Information-Theoretically Secure Construction 66
4.3.1 Block Broadcast Protocol ITBlockBC 66
4.3.2 Constructing Broadcast for Long Messages 69
4.4 Instantiating $\mathfrak{BD}$ in Domain-Amplification Protocols with Concrete Bit Broadcast Protocols 70

5 Availability Amplification 73
5.1 Availability-Amplification Setting 73
5.2 Contributions 74
5.3 The Broadcast Scheme for $n = 3$ and $t \leq 3$ 75
5.3.1 Detectable Information Checking 75
5.3.2 Detectable Verifiable Secret Sharing 78
5.3.3 Broadcast Scheme 81
5.3.4 Efficient Parallel Setup 83
5.4 Broadcast Scheme for any n and $t < n/2$ 84
5.4.1 Refreshing the Setup 85
5.4.2 Application to MPC 85
5.5 A More Efficient Scheme for any n ant $t < n/2$ 86
5.5.1 Additional Preliminaries 86
5.5.2 A Sketch of the Protocol $\mathsf{BCfromSetup}_n^+$ 88

Appendices 91

A On the Constructions of Liang and Vaidya [LV11a, LV10a, LV10b] 93

Bibliography 95

Chapter 1

Introduction

1.1 Distributed Computation

As an example, consider a setting with n parties who can communicate by exchanging messages and whose goal is to execute a voting procedure. As an input each of the parties holds the vote he wants to cast. Let the voting output be the winning candidate (or the set of candidates with the highest number of votes in case there are several of them with the same number of votes). Additionally, we require that the votes cast by the parties remain private. The voting procedure is implemented via a protocol instructing each of the parties what messages to send to which parties in which periods of time and how to obtain the resulting output. The goal of such a protocol is to achieve voting security guarantees while tolerating a potential misbehaviour of parties. Such a misbehaviour is modelled with a centralized adversary who can observe the parties internal states or even take full control of them. In particular, a protocol for the voting procedure must tolerate *any* adversarial strategy that cannot be known a priori.

The voting procedure shown above is an example of a distributed computation task. In general, such a task can be arbitrary ranging from parties' clocks synchronization to electing a leader among them. Investigating which distributed computation tasks can be realized with which security guarantees is an important area in computer science.

1.1.1 Modeling Distributed Computation

In this section we model distributed computation. In particular, we show which different models exist with which degrees of freedom.

Parties

We denote n parties by $P_1, P_2, \ldots, P_n$. Let $\mathcal{P}$ denote the set of parties, i.e., $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$. The parties are also called players. In the distributed systems community they are also referred to as "processes". Throughout the thesis, we will assume that the parties are masculine, i.e., we will refer to an individual party using the pronoun **"he"**.

For a set of parties $S \subseteq \mathcal{P}$ let $\overline{S}$ denote $\mathcal{P} \setminus A$.

Communication Model

One distinguishes *synchronous* and *asynchronous* models. In the synchronous model a protocol execution proceeds in synchronous rounds as follows:

1. Every party receives all messages sent to him in the previous round;
2. every party performs local computation and sends the messages as instructed by the protocol;
3. once all the messages are sent, the parties proceed to the next round.

In contrast, in the asynchronous case no guarantee is given that a party receives all the messages sent to him since they can be delayed arbitrarily. Inherently, building protocols for the asynchronous model is more challenging than for the synchronous model.

In this thesis we consider the **synchronous** model only.

Available Communication Means

As stated in the voting example, the parties can communicate by exchanging messages. That is, it is assumed that they have a fully connected point-to-point network available (we call it bilateral channels). Such bilateral channels can be *insecure*, *authentic*, *confidential* (also called *private*) or *secure*. In the insecure case the adversary can see the messages sent over the channel, substitute them and inject new messages. In the authentic case the adversary can see the messages sent over the channel but cannot inject new messages or substitute the message being sent. In the confidential case the adversary cannot see the messages sent over the channel but can substitute them. In the secure case the channel combines properties of authentic and confidential channel.

In this thesis we consider only **authentic** and **secure** channels. We will omit specifying the channel type in the text if the statement holds for both authentic and secure channels.

Adversarial Power

We consider two types of adversaries: *computationally bounded* and *unbounded*. In the computationally bounded case we assume that the adversary can perform only limited computations (e.g., computable in polynomial time), while in the unbounded case the adversarial computation power is unlimited.

We distinguish two types of adversarial corruption: *passive* and *active*. In the passive case the adversary by corrupting a party can only see the party's state but cannot actively influence its behaviour. In the active case the adversary takes full control over a party once he is corrupted, e.g., the adversary can instruct the party to completely stop sending any messages or even send maliciously created ones. As a generalization of the pure passive or active corruption one might consider a *mixed* adversary [FHM98, HLM13] that can corrupt each party either passively or actively. One might also consider a *covert* adversary [AL07] that can act as strong as an active adversary but avoids to be caught on cheating.

We say that a party that is not corrupted is *correct* (or alternatively called *honest*).

There are two types of adversary depending on which sets of parties the adversary can corrupt: *threshold* and *general*. A threshold adversary can corrupt at most t parties, where t is a commonly known threshold. General adversaries [HM97] are a generalization of the threshold ones whose corruption power is characterized by the sets of the parties they can corrupt.

Depending on how the adversary corrupts parties, one distinguishes between *static* and *adaptive* adversaries. A static adversary must specify the corrupted parties before the protocol starts its execution. An adaptive adversary decides which parties to corrupt during the protocol execution depending on the information the adversary sees.

We also note that the adversary can influence message delays in bilateral channels. That is, in the synchronous case the messages sent by the parties controlled by the adversary can depend on the messages sent by correct parties to corrupted parties in the same round. In the asynchronous case the adversary can delay messages arbitrarily. Such adversarial power is identified by referring to the adversary as *rushing*.

In this thesis we consider adversaries that are **static, active, threshold** and **rushing**. We will not specify explicitly the computational power of the adversary tolerated, because it will be implied by the security notion that the protocol achieves (see below).

Traditionally it is assumed that the adversary is female (even sometimes the explicit name "Eve" is used), so we will use the pronoun **"she"** when referring to the adversary.

Protocol Security

We consider the following notions of protocol security: *cryptographic, information-theoretic* and *perfect*. Cryptographic security specifies that the protocol achieves its security goals as long as a computationally bounded adversary cannot break some cryptographic assumption, e.g., such an assumption might state that factoring is hard. Information-theoretic security states that even an unbounded adversary can make the protocol fail only with a negligible probability, while perfect security guarantees that the failure probability is exactly 0.

Information-theoretic (and perfect) security notions are also referred as *unconditional* security because they do not assume any cryptographic assumptions to hold.

In this thesis we consider all three notions of protocol security: **cryptographic, information-theoretic** and **perfect**.

Protocol Efficiency

Along with feasibility of the protocols we consider their efficiency. We consider two measures of the protocol complexity: *round* and *communication*. By round complexity we understand the number of rounds that the protocol takes (it is directly applicable to the synchronous computation only, however, one might consider analogous notions in the asynchronous case). The communication complexity of a protocol is defined by Yao [Yao79] to be the number of bits sent/received by correct parties during the protocol run.[1]

[1]When counting the number of bits received by correct parties, we take into account only messages which were *actively* received by them, i.e., messages which should be received according to the protocol specification.

Setup

We distinguish two possible settings: *with trusted setup* and *without setup*. In the setting with trusted setup one assumes that a trusted party initially predistributes a correlated randomness which can later be used in the computation. The protocols that do not use setup are often said to realize the required functionality "from scratch".

An example of such a trusted setup can be public-key infrastructure (PKI) or pseudo-signatures setup [PW96].

In this thesis we consider protocols **with and without trusted setup**.

1.2 Broadcast

Broadcast is a distributed primitive that allows one specific party (called sender) to consistently distribute a value from a predefined domain $\mathcal{X}$ to the remaining parties. Without loss of generality, we assume that P_1 is the sender and $\mathcal{R} = \mathcal{P} \setminus \{P_1\}$ is the set of recipients. Let d-broadcast denote a broadcast primitive (also called broadcast channel) for message domain of size d.

Broadcast is one of the most fundamental primitives in distributed computing and is often used as a building block for constructing distributed computation protocols. In fact, it has even become standard to assume broadcast available directly as one of communication means given in the model.

Reducing the costs of the broadcast usage has become one of the goals when constructing efficient distributed computation protocols along with reducing their overall communication costs and round complexity [HMP00, KKK08, GGOR13].

1.2.1 Implementing Broadcast

In the model with active corruptions a protocol implementing broadcast is defined as follows:

Definition 1.1 *A protocol for the set of parties $\mathcal{P}$, where P_1 has an input value $v \in \mathcal{X}$ and each party in $\mathcal{R}$ outputs a value in $\mathcal{X}$, is called a* broadcast protocol for domain $\mathcal{X}$ *if the following conditions are satisfied:*[2]

CONSISTENCY: *All correct parties $P_i \in \mathcal{R}$ output the same value $v \in \mathcal{X}$.*

[2]The domain can without essential loss of generality be assumed to be $\mathcal{X} = [d]$, where here and below we define $[k] = \{1, \dots, k\}$.

VALIDITY: *If the sender P_1 is correct, then v is the input value of P_1.*
TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

The celebrated result of Pease, Shostak and Lamport states that broadcast can be implemented from scratch (i.e., from bilateral channels only) if and only if the number of active corruptions t is strictly smaller than $n/3$ [PSL80]. Communication-efficient implementations of broadcast from authenticated bilateral channels have been given in [BGP92, CW92, KS11]. An expected constant-round construction of broadcast from secure bilateral channels is given in [FM88]. A construction of broadcast from authenticated bilateral channels requiring $\mathcal{O}(\log n)$ rounds and tolerating $t < n/(4+\varepsilon)$ is given in [BOPV06].

If a trusted setup is available then broadcast can be implemented from bilateral channels for any $t < n$. Broadcast with cryptographic security can be realized if public-key infrastructure (PKI) is available for $t < n$ [DS83]. An expected constant round construction of broadcast from PKI and secure bilateral channels for $t < n/2$ is given in [KK06a, KK06b]. For the case of arbitrary $t < n$ it has been shown that no protocol can achieve broadcast in a constant number of rounds even if trusted setup is available [GKKO07]. Broadcast with information-theoretic security for $t < n$ can be realized if a pseudo-signature setup is available [PW96]. A more communication efficient information-theoretically secure construction based on pseudo-signatures for $t < n/2$ is given in [BHR07]. A study on which setups are sufficient for broadcast implementation is given in [FWW04].

Mutli-Valued Broadcast

Historically, the broadcast problem was introduced for binary values [PSL80]. However, in various applications *long* values are broadcast rather than single bits. Multi-valued broadcast protocols are designed to realize ℓ-bit broadcast while optimizing communication complexity of the protocol. Trivially, any protocol realizing multi-valued broadcast from bilateral channels (and a trusted setup) must communicate at least $\Omega(n\ell)$ bits. Protocols that achieve multi-valued broadcast with the optimal communication complexity from bilateral channels exist for $t < n/3$ [Pat11, LV11a] or $t < n/2$ [FH06] (assuming additional setup). Existing protocols achieving broadcast from bilateral channels and a trusted setup for $t < n$ [DS83, PW96] were initially designed to broadcast bits, but they can be easily adopted to broadcast long messages. A simple modification of the protocol by Dolev and Strong [DS83] is cryptographically secure and has communication complexity $\Omega(\ell n^2 + n^3\kappa)$. Analogously, the protocol by Pfitzmann and Waidner [PW96] is information-theoretically secure and has communication complexity $\Omega(\ell n^2 + n^6\kappa)$ [Fit03].

Extended Models

Apart from the trusted setup available one might also consider other additions to the model allowing to implement broadcast for $t \geq n/3$.

Fitzi et al. [FM00, CFF+05] showed that given a full network of local broadcast channels among each subset of b parties one can implement broadcast among n parties if and only if $t < \frac{b-1}{b+1}n$. In the model with a Q-Flip primitive broadcast is achievable if $t < n/2$ [FGMvR02].

In the model where PKI can be partially compromised (by exposing up to $c > 0$ secret keys of correct parties to the adversary) it has been shown that broadcast is achievable if and only if $2t + \min(t, c) < n$ [GKKY10, GGBS10].

Fitzi et al. [FM98] have also studied for which general adversaries broadcast is achievable from bilateral channels. This result has later been extended in [AFM99] by characterizing under which general mixed adversaries broadcast is achievable from bilateral channels.

Extend Notions of Broadcast

Another line of research considered which variations of broadcast can be achieved when $t \geq n/3$.

In [FGH+02] the authors present a detectable version of broadcast. In detectable version the parties execute a protocol that either achieves broadcast or aborts. It is then guaranteed that if all players are correct, then broadcast is achieved and in case of misbehavior the correct parties agree whether the execution aborted or not. Additionally, it is required that if the sender is correct and the execution aborted then the adversary learns no information about the sender's input (fairness).

In [FHHW03] the authors present a two-threshold version of broadcast. In the two-threshold version of broadcast consistency and validity properties of broadcast are guaranteed to be achieved in the presence of up to t_c and t_v malicious parties respectively. It is then shown that one can implement two-threshold broadcast from bilateral channels if and only if $t = 0$ or $t + 2T < n$ (where $t = \min(t_c, t_v)$ and $T = \max(t_c, t_v)$). A special case of two-threshold broadcast called "weak broadcast"[3] has

[3] Not to be confused with a different weak broadcast notion given in Definition 2.3.

also been studied in [Lam83].

1.2.2 Broadcast Amplification

Consider a setting where the number of parties that can be corrupted is $t \geq n/3$. Implementing broadcast in such a setting can be possible only if, in addition to bilateral channels, some addition to the model is made (e.g., a trusted setup; see Section 1.2.1 for more protocols/assumptions).

The perhaps most natural choice of such an addition is the availability of some weaker broadcast primitives. We model two cases where the weaker broadcast primitive is amplified to a stronger one. In the first case we consider how the broadcast domain can be amplified, i.e., for which ℓ and ℓ' broadcasting ℓ bits can be extended to broadcasting $\ell' > \ell$ bits. Second, we investigate whether broadcast availability can be prolonged. We model this by assuming that the broadcast primitive is available during some fixed number of rounds together with bilateral channels, then, we investigate whether this is sufficient for generating broadcast setup that can be used later to simulate broadcast from bilateral channels only.

In both amplification models we assume that the given (weaker) broadcast primitive is given in a form of an oracle denoted with $\mathfrak{BO}$.

Another related line of research is the amplification of other primitives, like OT extension [Bea96, IKNP03] or coin-toss extension [HMQU06].

A domain-amplification protocol is an example of the construction of a consistency primitive from another consistency primitive as defined in [Mau04].

Domain Amplification

First, we consider the following natural question: Can the sender broadcast a message with domain size d by using bilateral channels and broadcasting only a *single* message with domain size $d' < d$?

Definition 1.2 *Let $\phi_n(d)$ denote the minimal d' such that d-broadcast can be constructed from d'-broadcast and bilateral channels in the setting with n parties where any $t < n$ parties can be corrupted.*

Trivially, $\phi_n(d) \leq d$, as d-broadcast can be constructed directly from d-broadcast.

The most natural generalization of the above question is the following:[4] If any party can broadcast short messages, what is the minimal total number of bits that need to be broadcast to construct an ℓ-bit broadcast? More precisely, since we consider arbitrary alphabet sizes (not just powers of 2), the question is to determine the quantity $\phi_n^*(d)$ defined below.

Definition 1.3 *Let $\phi_n^*(d)$ denote the minimal d' such that d-broadcast can be constructed from the k primitives d'_1-broadcast, ..., d'_k-broadcast and bilateral channels in the setting with n parties where any $t < n$ parties can be corrupted and $d' = \prod_i d'_i$.*

Note that $\log d' = \sum_i \log d'_i$ is the total number of bits of information[5] broadcast. It is therefore often natural to state results for the quantity $\log \phi_n^*(d)$. It is obvious that $\phi_n^*(d) \leq \phi_n(d)$.

A protocol that amplifies the domain of a broadcast, in the sense of the above two definitions, is called a *domain-amplification protocol*. A domain-amplification protocol for domain size d can be used to replace a call to a d-broadcast primitive within another protocol. Hence domain-amplification protocols can be constructed recursively.

One can call $\phi_n(d)$ and $\phi_n^*(d)$ the *intrinsic broadcast complexity* of domain size d, in the single-sender and in the general multi-sender model, respectively.[6]

Domain Amplification & Multi-Valued Broadcast Protocols. We consider multi-valued broadcast protocols that can be viewed as a construction that given broadcast of short messages and bilateral channels realizes broadcast of longer messages, i.e., as a domain-amplification protocol. Later, once broadcast for short messages is substituted with a concrete broadcast protocol (e.g., cf. Section 1.2.1) we obtain a concrete protocol for broadcasting longer messages.

Availability Amplification

Assume that we have a broadcast channel to be temporarily available in some predetermined fixed number of rounds. Our goal is to be able to

[4]More refined versions of this question exist but will not be considered.

[5]Not necessarily exactly the number of actual bits.

[6]One could also consider a single-sender multi-shot model, i.e., the model where the sender can broadcast with the underlying broadcast multiple times. Later we give a protocol for the single-sender setting which requires only a single call to the underlying broadcast and is optimal even in the multi-shot model.

broadcast some information at some later point in time when temporarily available broadcast is not accessible anymore. A solution to this problem is to use the temporarily available broadcast to prepare some resource that can be used later in order to simulate broadcast invocations. A natural choice of such a resource is a trusted setup (cf. Section 1.1.1).

Then the problem of availability amplification can be formally formulated as follows: The parties generate a trusted setup using bilateral channels and temporarily-available broadcast channels. Later, with the help of the setup generated, the broadcast channels can be simulated using the bilateral channels only.

We employ two measures of efficiency for the temporary broadcast used in the setup protocol: round complexity and bit complexity. Round complexity denotes the number of rounds in which temporary broadcast is used and bit complexity denotes the number of bits broadcast via it.

1.3 Contributions

First, we investigate the domain-amplification problem in Chapter 3. We show that for $n = 3$ parties, broadcast for any domain size is possible if only a single 3-broadcast is available, and broadcast of a single bit ($d' = 2$) is not sufficient, i.e., $\phi_3(d) = 3$ for any $d \geq 3$. In contrast, for $n > 3$ no domain amplification in the single-sender model is possible, i.e., $\phi_n(d) = d$ for any d. However, if other parties than the sender can also broadcast some short messages, then domain amplification is possible for *any* n. We show that broadcasting $8n \log n$ bits in total suffices, independently of d, and that at least $n-2$ parties, including the sender, must broadcast at least one bit. Hence $\min(\log d, n-2) \leq \log \phi_n^*(d) \leq 8n \log n$.

Second, we study multi-valued broadcast protocols in Chapter 4. All known protocols implementing broadcast of an ℓ-bit message from point-to-point channels and a trusted setup tolerating any $t < n$ active corruptions have communication complexity at least $\Omega(\ell n^2)$. We give cryptographically secure and information-theoretically secure protocols for $t < n$ that communicate $\mathcal{O}(\ell n)$ bits for sufficiently large ℓ. This matches the optimal communication complexity bound for any protocol allowing to broadcast ℓ-bit messages. While broadcast protocols with the optimal communication complexity exist in cases where $t < n/3$ or $t < n/2$ [FH06, LV11a, Pat11], we are first to present such protocols for $t < n$.

Third, we study availability amplification in Chapter 5. We focus on the complexity of setup generation for information-theoretic protocols using secure channels and temporarily-available broadcast channels. We optimize the number of rounds in which the temporary broadcast channels are used while minimizing the number of bits broadcast with them. We give the first information-theoretically secure broadcast protocol tolerating $t < n/2$ that uses the temporary broadcast channels during only 1 round in the setup phase. Furthermore, only $\mathcal{O}(n^3)$ bits need to be broadcast with the temporary broadcast channels during that round, independently of the security parameter employed. The broadcast protocol presented in this thesis allows to construct the first information-theoretically secure multi-party computation protocol which uses a broadcast channel during only one round. Additionally, the presented broadcast protocol supports refreshing, which allows to broadcast an a priori unknown number of times given a fixed-size setup.

1.4 Outline

In Chapter 2 we give the necessary preliminaries for the broadcast protocols we present. In Chapter 3 we study domain-amplification protocols, while in Chapter 4 we focus on multi-valued broadcast protocols. In Chapter 5 we study broadcast availability amplification.

Chapter 2

Tools

2.1 General Tools

2.1.1 Collision-Resistant Hash Functions

Consider a family of hash functions $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$ indexed with a key k from a set $\mathcal{K}$, where each function H_k is a mapping from a domain $\mathcal{X}$ to a range $\mathcal{Y}$. The family of functions $\mathcal{H}$ is called collision-resistant if there is no efficient algorithm that given a uniformly sampled key $k \in \mathcal{K}$ outputs $v, v' \in \mathcal{X}$ such that $H_k(v) = H_k(v')$.

In practice we often consider hash functions which are *keyless*, i.e., we consider a single hash function $H : \mathcal{X} \to \mathcal{Y}$. One also can extend the notion of collision-resistance from keyed hash functions to keyless [Rog06]. Informally, H is called collision-resistant, if there is *no known* efficient algorithm that outputs $v, v' \in \mathcal{X}$ such that $H(v) = H(v')$. We will denote a keyless collision-resistant hash function with `CRHash`.

2.1.2 Universal Hash Functions [CW79]

Consider a family of functions $\mathcal{U} = \{U_k\}_{k \in \mathcal{K}}$ indexed with a key k from a set $\mathcal{K}$, where each function U_k maps elements of some set $\mathcal{X}$ to a fixed set of bins $\mathcal{Y}$. The family $\mathcal{U}$ is called ε-universal if for any two distinct messages $v_1, v_2 \in X$,

$$\frac{|\{k \in \mathcal{K} \mid U_k(v_1) = U_k(v_2)\}|}{|\mathcal{K}|} \leq \varepsilon.^{7}$$

A ε-universal hash function can for example be constructed as follows: Let $\mathcal{X} = \{0,1\}^{\ell}$, $\mathcal{K} = \mathcal{Y} = \mathrm{GF}(2^{\kappa})$, and any value $v \in \{0,1\}^{\ell}$ be interpreted as a polynomial f_v over $\mathrm{GF}(2^{\kappa})$ of degree $\lceil \ell/\kappa \rceil - 1$. The hash function is defined as $U_k(v) = f_v(k)$. We know that two distinct polynomials of degree $\lceil \ell/\kappa \rceil - 1$ can match in at most $\lceil \ell/\kappa \rceil - 1$ points. Hence, for any two distinct $v_1, v_2 \in \{0,1\}^{\ell}$,

$$\frac{|\{k \in \{0,1\}^{\kappa} \mid U_k(v_1) = U_k(v_2)\}|}{2^{\kappa}} \leq \frac{\lceil \ell/\kappa \rceil - 1}{2^{\kappa}} \leq 2^{-\kappa}\ell.$$

So, $\{U_k\}_{k \in \{0,1\}^{\kappa}}$ is a family of $(2^{-\kappa}\ell)$-universal hash functions. We will denote a ε-universal hash function with `ITHash`.

2.1.3 Identifying Predicates

An identifying predicate allows to identify a specific element v from some small subset $S \subseteq \mathcal{X}$, where $\mathcal{X}$ is a potentially large domain. To our knowledge, this concept has been first introduced in [HR13b].

Definition 2.1 *A c-*identifying predicate *for domain $\mathcal{X}$ is a family of functions $Q_{k \in \mathcal{K}} : \mathcal{X} \to \{0,1\}$ such that for any $S \subseteq \mathcal{X}$ with $|S| \leq c$ and any value $v \in S$ there exists a key $k \in \mathcal{K}$ with $Q_k(v) = 1$ and $Q_k(v') = 0$ for all $v' \in S \setminus \{v\}$. We say that such v is* uniquely identified *in S by Q_k.*

Note that any identifying predicates Q_k achieve monotonicity in the following sense:

Lemma 2.1 *If v is uniquely identified in S by Q_k, then v is uniquely identified in any $S' \subseteq S$ with $v \in S'$.*

The designer's goal when constructing an identifying predicate family is to have $|\mathcal{K}|$ as small as possible given c and $|\mathcal{X}|$. We give two constructions of c-identifying predicates below. The first construction is very simple and helps to develop an intuition about identifying predicates, while the second one is more involved and achieves better efficiency.

[7] This is a combinatorial definition of a universal hash function, usually the latter condition is written probabilistically as $\Pr[k \xleftarrow{\$} \mathcal{K} : U_k(v_1) = U_k(v_2)] \leq \varepsilon$.

Position-based Construction

Assume now $\mathcal{X} = \{0,1\}^\ell$, then Q can be constructed as following: for a set S and a value v, we find $c-1$ bit positions $p_1, \ldots, p_{c-1}$ such that v differs from any other value $v' \in S$ in some position p_i. Then the key k for a set S and a value v is defined by the positions $p_1, \ldots, p_{c-1}$ and the bits $b_1, \ldots, b_{c-1}$ which v has at these positions. Hence, for this Q the key space $\mathcal{K} = \{0,1\}^{(c-1)(\lceil \log \ell \rceil + 1)}$.

Polynomial-based Construction

Let $\ell = \log|\mathcal{X}|$. For $\kappa \in \mathbb{N}$, let any value $v \in \mathcal{X}$ be interpreted as a polynomial f_v over $\mathrm{GF}(2^\kappa)$ of degree $\lceil \ell/\kappa \rceil - 1$. We find a point $x \in \mathrm{GF}(2^\kappa)$ such that $f_v(x)$ is different from all other values $f_{v'}(x)$ for $v' \in S \setminus \{v\}$. For such a point x to always exist we need that the total number of points in the field is larger than the number of points in which f_v may coincide with other polynomials $f_{v'}$, i.e., $2^\kappa > (c-1)(\lceil \ell/\kappa \rceil - 1)$. To satisfy this condition, it is enough to choose $\kappa := \lceil \log(c\ell) \rceil$. The key for the identifying predicate is defined as $k = (x, f_v(x))$, which is encoded using $2\lceil \log(c\ell) \rceil$ bits.[8] The predicate is defined as follows:

$$Q_{(x,y)}(v) = \begin{cases} 1, \text{ if } f_v(x) = y; \\ 0, \text{ otherwise.} \end{cases}$$

Lemma 2.2 *The polynomial-based construction gives a c-identifying predicate Q with domain $\mathcal{X}$ and key space $\mathcal{K}_c^{\mathcal{X}} = \{0,1\}^{2\lceil \log(c \log|\mathcal{X}|) \rceil}$.*

2.1.4 Resolution Functions

An identifying predicate allows to identify a specific element v from some set S of potentially a large domain $\mathcal{X}$. Resolution functions provide a different way of choosing one of the values in S while explicitly not choosing the others.

Definition 2.2 *A c-*resolution function* for domain $\mathcal{X}$ and range $\mathcal{Y}$ is a family of functions $F_{k \in \mathcal{K}} : \mathcal{X} \to \mathcal{Y}$ such that for any $S \subseteq \mathcal{X}$ with $|S| \le c$ there exists a key $k \in \mathcal{K}$ with $F_k(v) \ne F_k(v')$ for any $v \ne v'$ from S. Such a key k is said to* resolve *the set S.*

[8]Such a point x can be efficiently found by random sampling elements in $\mathrm{GF}(2^\kappa)$. Indeed, for $\kappa = \lceil \log(c\ell) \rceil$ more than half of the elements in $\mathrm{GF}(2^\kappa)$ are points where f_v is different from all other $f_{v'}$.

It can be easily seen that any resolution function induces identifying predicate. Take any c-resolution function family F_k with key space $\mathcal{K}$, domain $\mathcal{X}$ and range $\mathcal{Y}$. A pair $(k,y) \in \mathcal{K} \times \mathcal{Y}$ can be used to induce identifying predicate Q based on F as following:

$$Q_{(k,y)}(v) = \begin{cases} 1, \text{ if } F_k(x) = y; \\ 0, \text{ otherwise.} \end{cases}$$

Analogously to identifying predicates, we say that v is *identified* in S by a pair (k,y) if $F_k(v) = y$ (trivially, only one value can be identified if k resolves the set S). The goal of constructing such a function F is to have $|\mathcal{K}|$ and $|\mathcal{Y}|$ as small as possible given c and $|\mathcal{X}|$. We give a construction of a c-resolution function with domain $\mathcal{X}$ below.

Polynomial-based Construction

This construction is very similar to the polynomial-based construction for identifying predicates presented before. Let $\ell = \log|\mathcal{X}|$. Consider any set $S \subseteq \mathcal{X}$ with $|S| \leq c$. For $\kappa \in \mathbb{N}$, let any value $v \in \mathcal{X}$ be interpreted as a polynomial f_v over GF(2^κ) of degree $\lceil \ell/\kappa \rceil - 1$. We find a point $x \in$ GF(2^κ) such that $f_v(x) \neq f_{v'}(x)$ for any two $v \neq v' \in S$. For such a point x to always exist we need that the total number of points in the field is larger than the number of points in which any f_v may coincide with any $f_{v'}$, i.e., $2^\kappa > \frac{c(c-1)}{2}(\lceil \ell/\kappa \rceil - 1)$. To satisfy this condition, it is enough to choose $\kappa := \lceil \log(c^2\ell) \rceil$. The resolution function is defined as $F_x(v) = f_v(x)$ with the key space and range space being GF(2^κ). So, the key and the value of the resolution function can be encoded using $\lceil \log(c^2\ell) \rceil$ bits.

Lemma 2.3 *The polynomial-based construction gives a c-resolution function F for domain $\mathcal{X}$ with key and range spaces $\mathcal{K}_c^{\mathcal{X}} = \mathcal{Y}_c^{\mathcal{X}} = \{0,1\}^{\lceil \log(c^2 \log|\mathcal{X}|) \rceil}$.*

2.2 Additional Distributed Primitives

We now turn to broadcast-type distributed primitives which will become useful when constructing our protocols in a modular way.

2.2.1 Weak Broadcast

Weak broadcast (a.k.a. "Crusader agreement" [Dol82]) is a weak form of broadcast, where the recipients either decide on the value sent by the sender or on a special symbol $\perp$ indicating that the sender is malicious.

Definition 2.3 *A protocol achieves weak broadcast if it allows the sender P_1 to distribute a value $v \in \mathcal{X}$ among parties $\mathcal{R}$ with every party P_i outputting a value $v_i \in \mathcal{X} \cup \{\perp\}$ such that:*

VALIDITY: *If the sender P_1 is correct, then every correct $P_i \in \mathcal{R}$ outputs $v_i = v$.*

WEAK CONSISTENCY: *If a correct $P_i \in \mathcal{R}$ outputs $v_i \neq \perp$, then every correct $P_j \in \mathcal{R}$ outputs $v_j \in \{v_i, \perp\}$.*

TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

2.2.2 Graded Broadcast

Graded broadcast (a.k.a. gradecast) was introduced by Feldman and Micali [FM88, FM97]. It allows to broadcast a value among the set of recipients but with weaker consistency guarantees. In addition to the value v_i each recipient P_i also outputs a grade g_i describing the level of agreement reached by the players. Below we give a slightly weakened version of the original definition given in [FM97].

Definition 2.4 *A protocol achieves graded broadcast if it allows the sender P_1 to distribute a value $v \in \mathcal{X}$ among parties $\mathcal{R}$ with every party P_i outputting a value $v_i \in \mathcal{X}$ with a grade $g_i \in \{0, 1\}$ such that:*

VALIDITY: *If the sender P_1 is correct, then every correct $P_i \in \mathcal{R}$ outputs $(v_i, g_i) = (v, 1)$.*

GRADED CONSISTENCY: *If a correct $P_i \in \mathcal{R}$ outputs (v_i, g_i) with $g_i = 1$, then every correct $P_j \in \mathcal{R}$ outputs (v_j, g_j) with $v_j = v_i$.*

TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

An equivalent gradecast definition that uses grades from the set $\{0, 1, 2\}$ is employed in [KK06a, KK06b]:

Definition 2.5 *A protocol achieves graded broadcast if it allows the sender P_1 to distribute a value $v \in \mathcal{X}$ among parties $\mathcal{R}$ with every party P_i outputting a value $v_i \in \mathcal{X}$ with a grade $g_i \in \{0, 1, 2\}$ such that:*

VALIDITY: *If the sender P_1 is correct, then every correct $P_i \in \mathcal{R}$ outputs $(v_i, g_i) = (v, 2)$.*

GRADED CONSISTENCY: *If a correct $P_i \in \mathcal{R}$ outputs (v_i, g_i) with $g_i = 2$, then every correct $P_j \in \mathcal{R}$ outputs (v_j, g_j) with $v_j = v_i$ and $g_j \geq 1$.*

TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

One can easily see that the two definitions are equivalent. First, given any $\{0,1\}$-grade protocol that satisfies Definition 2.4 we can construct a $\{0,1,2\}$-grade protocol that satisfies Definition 2.5 by updating the final grades as following: $0 \to 1$ and $1 \to 2$. Second, given any $\{0,1,2\}$-grade protocol that satisfies Definition 2.5 we can construct a $\{0,1\}$-grade protocol that satisfies Definition 2.4 by updating the final grades as following: $0,1 \to 0$ and $2 \to 1$.

We detail now why Definition 2.5 (and analogously Definition 2.4) is weaker than the gradecast definition in [FM97]. The graded consistency property in the Definition 2.5 guarantees that correct players agree on the same value only if one of them has grade 2, while in the definition of [FM97] these agreement must be reached if one of them has grade 1 or 2.

2.2.3 Verifiable Secret Sharing

Verifiable secret sharing [CGMA85] (VSS) is a classical cryptographic primitive for secure sharing of a secret s (assumed to be from some finite field $\mathbb{F}$) held by one of the parties (called dealer and denoted by D) that can be later reconstructed by the parties. It lies in a core of many protocols for multi-party computation and is used in various applications. VSS is useful as a distributed commitment for the dealer D. Thus, we may say that a dealer who completes sharing has committed to s, and that upon completing reconstruction he decommits .

Definition 2.6 *A verifiable secret-sharing scheme is a pair of protocols* (VSS-Share, VSS-Rec) *for a set of players $\mathcal{P}$, one of whom, the dealer D, holds input $s \in \mathbb{F}$. VSS must satisfy the following security guarantees:*

CORRECTNESS: *At the end of* VSS-Share *there exists a fixed value $s^* \in \mathbb{F}$, defined by the joint view of the correct parties, such that all correct parties will output s^* in* VSS-Rec. *If D is correct, then $s^* = s$.*

PRIVACY: *If D is correct, then prior to* VSS-Rec *the adversary gains no information on s (i.e., his view is statistically independent of s).*

TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

We say that a secret s is *verifiably shared* among the parties if each correct party holds some state such that, when correct parties invoke VSS-Rec on that joint state, they will reconstruct s.

We call a VSS scheme linear if it meets the following additional condition.

LINEARITY: *The parties can verifiably share any linear combination of already verifiably shared values by locally computing the linear combination of their shares.*

Chapter 3

Domain Amplification

In this chapter we formalize the concept of domain amplification and prove a number of results, both feasibility results in terms of protocols as well as infeasibility results in terms of impossibility proofs.

This chapter is based on [HMR14].

3.1 Contributions

We first study the setting where the sender uses a single broadcast primitive of smaller domain. For the case of three parties ($n = 3$), the smallest non-trivial case, we show the quite surprising result that broadcast for any domain size d is possible if only a single 3-broadcast ($d' = 3$) is available. Moreover broadcast of a single bit ($d' = 2$) is not sufficient. In other words, $\phi_3(d) = 3$ for any $d \geq 3$.

In contrast, for $n > 3$ no domain amplification is possible, i.e., $\phi_n(d) = d$ for any d.

If not only the sender, but also other parties can broadcast some short messages, then (strong) domain amplification is possible for *any* n. We show that broadcasting $8n \log n$ bits of information in total suffices, independently of d, i.e., $\log \phi_n^*(d) \leq 8n \log n$. On the negative side, we show that at least $n - 2$ parties must broadcast at least one bit, i.e., $\min(\log d, n - 2) \leq \log \phi_n^*(d)$.

The protocol that uses $8n \log n$ bits to broadcast a value of domain size d communicates exponentially many messages over point-to-point channels. We give an optimized version of this protocol which communicates a polynomial number of messages over point-to-point channels but needs to broadcast $\mathcal{O}(n^2 \log\log d)$ bits with the underlying broadcast primitive.

Furthermore, we study the round complexity of domain-amplification protocols. We show that any non-trivial construction (where the underlying broadcast primitive is not used to broadcast the message directly) cannot be constant-round.

3.2 Domain-Amplification Setting

A domain-amplification protocol consists of the programs $\pi_1, \ldots, \pi_n$ that the players $P_1, \ldots, P_n$ use. Each program π_i is a randomized algorithm (which takes an input from domain $\mathcal{X}$ in case of the sender's program π_1) and produces an output. The program π_i has $n-1$ interfaces to point-to-point channels to communicate with the other programs and additional interfaces to access a special oracle $\mathfrak{BO}$ providing access to the underlying broadcast channels given as "black-box".

We now describe how the programs interact with $\mathfrak{BO}$. First, we extend the notion of d-broadcast given in Section 1.2.2. Let (r, P_i, d)-broadcast be a broadcast channel available in round r which allows P_i to broadcast one single value from a domain of size d among the parties. We assume that each program π_j has an interface to each (r, P_i, d)-broadcast channel. Whenever we say that the parties broadcast with $\mathfrak{BO}$, we mean that they actually access the corresponding broadcast channel by explicitly giving input/asking for an output on that channel's interface.

The protocol must ensure that the correct parties agree on which (r, P_i, d)-broadcast channels to invoke, that is, on r, P_i and d.[9] We say that a (r, P_i, d)-broadcast channel is *used* if the correct parties access it, i.e., in round r correct parties expect an output provided by P_i of a domain of the size d (in case of a correct P_i, he provides the corresponding input). Note that which channels are used by the protocol may not be necessarily fixed a priori and may depend on the execution. We say that a domain-amplification protocol has a *static* $\mathfrak{BO}$ usage pattern if the broadcast channels used are fixed beforehand. As opposed to the static case, protocols with a *dynamic* $\mathfrak{BO}$ usage pattern allow to broadcast with $\mathfrak{BO}$

[9]This requirement stems from the observation that the broadcast channel may be implemented via a different protocol and hence in order to employ it all correct parties must start its execution together while agreeing on the broadcasting party and the domain of the broadcast value. Note that without this requirement, the $\mathfrak{BO}$ could be abused to reach agreement on "hidden" information, e.g., one could broadcast an ℓ-bit message v with using $\mathfrak{BO}$ only for a single bit (in round v).

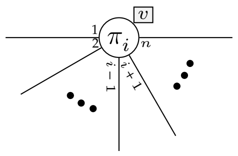

Figure 3.1: Drawing of a program π_i. It has $n - 1$ interfaces to bilateral channels with other players $\mathcal{P} \setminus \{P_i\}$ labeled accordingly. The program π_i is given v as input.

adaptively to the execution, where of course still agreement on which broadcast channels to use is required among the correct parties.

Depending on which channels are used we distinguish the following models.

Definition 3.1 *The **single-sender** model allows for protocols where only (r, P_1, d)-broadcast channels are used, i.e., only P_1 broadcasts with $\mathfrak{BD}$ (If only one channel is used then such a single-sender model is called* single-shot*; otherwise, it is called* multi-shot.*) The **multi-sender** model does not put any limitations on the broadcast channels used.*

The costs d' of $\mathfrak{BD}$ usage of a domain-amplification protocol with a static $\mathfrak{BD}$ pattern is defined to be $\prod_i d_i$, where d_i's are the domain sizes of the broadcast channels used. The protocols with a dynamic $\mathfrak{BD}$ usage pattern have costs d' to be computed as the maximum of $\prod_i d_i$ among all possible executions.

A domain-amplification protocol that uses $\mathfrak{BD}$ to broadcast a message directly (i.e., its costs d' are equal to the broadcast value domain $d = |\mathcal{X}|$) is called *trivial*. We say that a domain-amplification protocol is *non-trivial* if $d' < d$.

3.3 Impossibility Proofs of Domain Amplification

We prove lower-bounds on the $\mathfrak{BD}$ usage by employing a standard indistinguishability argument that is used to prove that certain security goals cannot be achieved by any protocol in the Byzantine environment [PSL80]. Such a proof goes by contradiction, i.e., by assuming that the security goals can be satisfied by means of some protocol $(\pi_1, \ldots, \pi_n)$. Then the programs π_i are used to build a *configuration* with contradictory behavior. The configuration consists of multiple copies of π_i connected with bilateral channels and given admissible inputs. A pictorial drawing of a program in such a configuration is shown in Figure 3.1. When describing a configuration we will often use such a drawing accompanied with a textual description. If in the drawing an interface to a bilateral channel is not depicted then it is connected to a "null" device which simulates the program sending no messages. The interfaces to $\mathfrak{BO}$ are never drawn. Once the configuration is built, one simultaneously starts all the programs in the configuration and analyzes the outputs produced by the programs locally. By arguing that the view of some programs π_i and π_j in the configuration is indistinguishable from their view when run by the corresponding players P_i and P_j (while the adversary corrupts the remaining players in $\mathcal{P} \setminus \{P_i, P_j\}$) one deduces consistency conditions on the outputs by π_i and π_j that lead to a contradiction.

The main novelty of the proofs presented is that we consider an extended communication model where in addition to bilateral channels players are given access to $\mathfrak{BO}$. While following the path described above, one needs to additionally define the $\mathfrak{BO}$ behavior in the configuration.

In the following impossibility proofs we assume that the $\mathfrak{BO}$ usage pattern is static. Later in Section 3.8 we show how to adapt the impossibility proofs given to include protocols with a dynamic $\mathfrak{BO}$ usage pattern. Furthermore, the lower bounds are given only for perfectly-secure protocols, i.e., those that fail with probability 0.

3.4 Existing Domain-Amplification Protocols

Since broadcast can be implemented from bilateral channels for $t < n/3$ [BGP92], one can trivially see this protocol as domain-amplification protocol that does not employ $\mathfrak{BO}$ protocol.

All known protocols for efficient multi-valued broadcast [TC84, FH06, LV11a, Pat11] can be interpreted as domain-amplification protocols, as they actually employ an underlying broadcast scheme for short messages (besides the point-to-point channels). These protocols tolerate only $t < n/3$ or $t < n/2$.[10]

Another method for domain amplification can be derived from existing signature-based broadcast protocols [DS83, PW96]. One can use

[10]Note that in [LV11a] the authors state that their broadcast protocol can be extended to tolerate $t \geq n/3$. We detail in Appendix A why this extension is only possible for $t < n/2$.

the available black-box broadcast oracle $\mathfrak{BO}$ to generate an appropriate setup (e.g., a PKI) and then use the corresponding protocol over point-to-point channels to broadcast the ℓ-bit message. Thus we obtain domain-amplification protocols for $t < n$ with cryptographic and statistical security.

The following table summarizes the complexity costs of the existing approaches to domain amplification.

Thr.	Security	$\mathfrak{BO}$	Bil. Chann.	Ref.
$t < n/3$	perfect	0	$\mathcal{O}(\ell n^2)$	[BGP92]
		$\mathcal{O}(n)$	$\mathcal{O}(\ell n^2)$	[TC84]
		$\mathcal{O}(\sqrt{\ell} n^2 + n^4)$	$\mathcal{O}(\ell n)$	[LV11a]
		$\mathcal{O}(n^2)$	$\mathcal{O}(\ell n)$	[Pat11]
$t < n/2$	inf.-th.	$\mathcal{O}(n^2 + n\kappa)$	$\mathcal{O}(\ell n + n^3\kappa)$	[FH06]
$t < n$	perfect	$\ell\mathcal{B}(1)$	0	Trivial
		$8n \log n$	$\mathcal{O}(n^{2n+3}\ell)$	Sec. 3.6.1
		$\mathcal{O}(n^2 \log \ell)$	$\mathcal{O}(n^3\ell)$	Sec. 3.6.2
	crypto	$\mathcal{O}(n\kappa)$	$\mathcal{O}(\ell n^2 + n^3\kappa)$	[DS83]
		$\mathcal{O}(n^2 + n\kappa)$	$\mathcal{O}(\ell n)$	Sec. 4.2
	inf.-th.	$\mathcal{O}(n^8\kappa(\log \ell)^2)$	$\mathcal{O}(\ell n^2 + n^6\kappa)$	[PW96]
		$\mathcal{O}(n^4 + n^3\kappa)$	$\mathcal{O}(\ell n)$	Sec. 4.3

Table 3.1: The overview of existing domain-amplification protocols. The columns labeled "$\mathfrak{BO}$" and "**Bil. Chann.**" represent the communication costs over $\mathfrak{BO}$ and bilateral channels, respectively. The table omits the protocol given in Section 3.5.1 that can be used only for $n = 3$.

3.5 Single-Sender Model

In this section we consider a single-sender model, that is, only the sender is allowed to use the $\mathfrak{BO}$ oracle. First, we completely investigate the situation for $n = 3$, that is, we show that 3-broadcast is enough to simulate any d-broadcast while 2-broadcast is not. On the negative side, we prove that for any $n > 3$ perfectly secure domain amplification is not possible, showing that $n = 3$ is a peculiar case in the context of domain-amplification protocols.

3.5.1 Domain Amplification for 3 Parties

We construct a domain-amplification protocol for three parties that allows the sender to broadcast a value v from domain $\mathcal{X}$ of size d, where the sender uses $\mathfrak{BC}$ to broadcast one value from a domain $\mathcal{X}'$ of size $d' = 3$. For ease of presentation, we assume that $\mathcal{X} = [d]$ and $\mathcal{X}' = [3]$.

The protocol works recursively. For $d = 3$, v is broadcast directly via $\mathfrak{BC}$. For $d \geq 4$, the sender transmits v to both recipients, who then exchange the received values and forward the exchanged values back to the sender. Finally, the sender broadcasts a hint h from domain $[d-1]$, which allows each recipient to decide which of the values he holds is the right one. Broadcasting the hint is realized via recursion.[11]

The crucial trick in this protocol is the computation of the hint h. Very generically, this computation is expressed as a special function which takes as input three values (the original value v and the two values sent back to the sender) and outputs h. Given the hint h, the recipients decide on the value received from the sender if it is consistent with h. Otherwise, if the other recipient's value (as received in the exchange phase) is consistent with h, then that value is taken. Finally, if no value is consistent with h then some default value (say $\perp$) is taken.

More formally, denote the value of the sender by v; the values received by the recipients P_2 and P_3 by v_2 and v_3, respectively; the values received by the recipients in the exchange phase by v_{32} and v_{23}, respectively; and the values sent back to the sender by v_{321} and v_{231}, respectively. The sender computes the hint h to be $g_d(v, v_{321}, v_{231})$, where g_d is a hint-producing function mapping triples of values from $[d] \times [d] \times [d]$ into the hint domain $[d-1]$. Once the hint h is computed, the sender broadcasts it. Recipient P_2 outputs v_2 if $h = g_d(v_2, v_{32}, \widetilde{v_{231}})$ for some $\widetilde{v_{231}} \in [d]$. Otherwise, P_2 outputs v_{32} if $h = g_d(v_{32}, \widetilde{v_{321}}, v_2)$ for some $\widetilde{v_{321}} \in [d]$. Otherwise, P_2 outputs $\perp$. P_3 decides analogously. Clearly, this protocol guarantees validity. Consistency is achieved as long as

$$\forall v_2, v_3, \widetilde{v_{231}}, \widetilde{v_{321}} \in [d] : \; v_2 \neq v_3 \;\Rightarrow\; g_d(v_2, v_3, \widetilde{v_{231}}) \neq g_d(v_3, \widetilde{v_{321}}, v_2). \quad (3.1)$$

[11]As we see later, the recursion can be made much more efficient with the help of so-called identifying predicates. We focus on the feasibility results and hence do not optimize the protocols.

For $d \geq 4$, the function $g_d(x, y, z)$ can be constructed as follows: For $x \leq d-1$, let $g_d(x, y, z) = x$ (for any y, z). For $x = d$, let $g_d(x, y, z) = \min([d-1] \smallsetminus \{y, z\})$. One can easily verify that g_d satisfies (3.1).

```
Protocol AmplifyBC_3(d, v)
  1. If $d = 3$ then broadcast $v$ using the $\mathfrak{BO}$.
  2. Otherwise:
      2.1 $P_1$ sends $v$ to $P_2$ and $P_3$. Denote the values received with
          $v_2$ and $v_3$, respectively.
      2.2 $P_2$ sends $v_2$ to $P_3$ and $P_3$ sends $v_3$ to $P_2$. Denote the values
          received by $P_2$ and $P_3$ with $v_{32}$ and $v_{23}$, respectively.
      2.3 $P_2$ sends $v_{32}$ to $P_1$ and $P_3$ sends $v_{23}$ to $P_1$. Denote the values
          received by $v_{321}$ and $v_{231}$, respectively.
      2.4 $P_1$ computes $h = g_d(v, v_{321}, v_{231})$.
          Parties invoke AmplifyBC_3$(d-1, h)$.
      2.5 $P_2$: If there exists $\widetilde{v_{231}}$ such that $h = g_d(v_2, v_{32}, \widetilde{v_{231}})$ decide
          on $v_2$. Else if there exists $\widetilde{v_{321}}$ such that $h = g_d(v_{32}, \widetilde{v_{321}}, v_2)$
          decide on $v_{32}$. Otherwise decide on $\perp$.
      2.6 $P_3$: If there exists $\widetilde{v_{321}}$ such that $h = g_d(v_3, \widetilde{v_{321}}, v_{23})$ decide
          on $v_3$. Else if there exists $\widetilde{v_{231}}$ such that $h = g_d(v_{23}, v_3, \widetilde{v_{231}})$
          decide on $v_{23}$. Otherwise decide on $\perp$.
```

Lemma 3.1 *The protocol* AmplifyBC$_3$ *achieves broadcast for any $t < n = 3$. The sender P_1 broadcasts one value from domain $[3]$ via $\mathfrak{BO}$.*

Proof: We prove by induction that the broadcast properties are satisfied. For $d = 3$, broadcast is achieved by assumption of $\mathfrak{BO}$. Now consider $d \geq 4$:

VALIDITY: If the sender is correct, then P_2 and P_3 receive $h = g_d(v, v_{321}, v_{231})$ as output from the recursive call to AmplifyBC$_3$. As $h = g_d(v_2, v_{32}, \widetilde{v_{231}})$ for $\widetilde{v_{231}} = v_{231}$, a correct P_2 decides on $v_2 = v$. Analogously, $h = g_d(v_3, \widetilde{v_{321}}, v_{23})$ for $\widetilde{v_{321}} = v_{321}$, a correct P_3 decides on $v_3 = v$.

CONSISTENCY: This property is non-trivial only if both P_2 and P_3 are correct, hence $v_{23} = v_2$ and $v_{32} = v_3$. Due to the Consistency property of the recursive call to AmplifyBC$_3$ both P_2 and P_3 receive the same hint h. If $v_2 = v_3$, then by inspection of the protocol both parties decide on the same value (namely on v_2 if $h \in \{g_d(v_2, v_2, \cdot), g_d(v_2, \cdot, v_2)\}$ and on $\perp$ otherwise). If $v_2 \neq v_3$, then (3.1) implies that if P_2 decides on

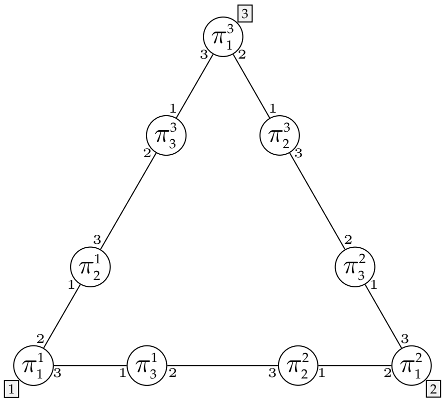

Figure 3.2: The configuration for $n = 3$ to show the impossibility of domain amplification with broadcasting 1 bit only via $\mathfrak{BO}$

v_2 (i.e., $h = g_d(v_2, v_{32}, \widetilde{v_{231}})$), then P_3 does not decide on v_3 (i.e., $h \neq g_d(v_3, \widetilde{v_{321}}, v_{23})$), but decides on $v_{23} = v_2$ (i.e., $h = g_d(v_{23}, v_3, \widetilde{v_{231}})$). Analogously, if P_3 decides on v_3, then P_2 decides on v_3 as well.

TERMINATION: Follows by inspection. ■

3.5.2 Lower Bounds in the Single-Sender Model

Lemma 3.2 *There is no perfectly-secure protocol among 3 parties achieving domain amplification for domain $\mathcal{X}$ with $|\mathcal{X}| \geq 3$ by broadcasting only 1 bit via $\mathfrak{BO}$ and tolerating any $t < 3$ corruptions.*

Proof: Assume towards a contradiction that there is such a protocol (π_1, π_2, π_3). Without loss of generality, assume that $\mathcal{X} = [d]$ for some $d \geq 3$.

We consider the following configuration: For $i = 1, 2, 3$ and $j = 1, 2, 3$ let π_i^j be an instance of π_i. For $j = 1, 2, 3$ let π_1^j be given input j. We construct the configuration by connecting programs π_i^j as shown in Figure 3.2. Now we execute the programs. Whenever any program π_1^j broadcasts a bit with $\mathfrak{BO}$ this bit is given to programs π_2^j and π_3^j.

Since there are 3 programs $\pi_1^1, \pi_1^2, \pi_1^3$ broadcasting 1 bit only, there exist two of them π_1^i and π_1^j broadcasting the same bit. Without loss of

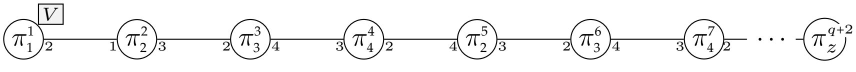

Figure 3.3: The configuration for $n = 4$ to show the impossibility of non-trivial domain amplification with only the sender broadcasting

generality, assume that π_1^1 and π_1^2 broadcast the same bit. The configuration can be interpreted in three different ways, which lead to contradicting requirements on the outputs of the programs. (i) P_1 holds input 1 and executes π_1^1, P_3 executes π_3^1, and P_2 is corrupted and executes the remaining programs in the configuration. Due to the validity property, π_3^1 must output 1. (ii) P_1 holds input 2 and executes π_1^2, P_2 executes π_2^2, and P_3 is corrupted and executes the remaining programs in the configuration. Due to the validity property, π_2^2 must output 2. (iii) P_3 executes π_3^1, P_2 executes π_2^2, and P_1 is corrupted and executes the remaining programs in the configuration. Due to the consistency property, π_3^1 and π_2^2 must output the same value. These three requirements cannot be satisfied simultaneously, hence whatever output the programs make, the protocol (π_1, π_2, π_3) is not a perfectly-secure domain-amplification protocol. ∎

Lemma 3.3 *There is no perfectly-secure protocol among $n \geq 4$ parties achieving non-trivial domain amplification in the single-sender multi-shot model tolerating any $t < n$ corruptions.*

Proof: We first prove the lemma for $n = 4$, then reduce the case of arbitrary $n > 4$ to $n = 4$.

(Case $n = 4$) Assume towards a contradiction that there exists a perfectly-secure protocol $(\pi_1, \pi_2, \pi_3, \pi_4)$ achieving non-trivial domain-amplification in the single-sender model in q rounds (for some $q \in \mathbb{N}$). On the highest level our proof consists of three steps. (i) we define a configuration. (ii) we show that all programs in the configuration must output the same value v. (iii) we use an information flow argument to prove that there is a program in the configuration that does not have enough information to output v with probability 1 (this argument is inspired by [Lam83]).

(i) We consider the following configuration: Let π_i^j denote an instance of the program π_i. Consider a chain of $q+2$ programs $\pi_1^1, \pi_2^2, \pi_3^3, \pi_4^4, \pi_2^5, \pi_3^6, \ldots, \pi_z^{q+2}$ connected as shown in Figure 3.3. The chain is built starting with a program π_1^1 and then by repeatedly alternating copies of the programs π_2, π_3 and π_4 until the chain has $q+2$ programs. To simplify the notation we will sometimes refer to the programs in the chain without the subscript, i.e., as to $\pi^1, \pi^2, \ldots, \pi^{q+2}$. Let π_1^1 be given as input a uniform random variable V chosen from domain $\mathcal{X}$. Now we execute the programs. Whenever π_1^1 uses $\mathfrak{BO}$ to broadcast some x, the value x is given to all recipients' programs in the configuration (i.e., to all π_i^j such that $i \in \{2,3,4\}$).

(ii) First, we prove that any pair of connected recipients' programs (π_i^a, π_j^{a+1}) $(a \geq 2)$ in the chain output the same value. One can view the configuration as the player P_i running the program π_i^a and P_j running π_j^{a+1} while the adversary corrupting $\{P_1, P_2, P_3, P_4\} \smallsetminus \{P_i, P_j\}$ is simulating the programs $\pi^1, \ldots, \pi^{a-1}$ and $\pi^{a+2}, \ldots, \pi^{q+2}$. Due to the consistency property, π_i^a and π_j^{a+1} must output the same value. Since every connected pair of the recipients' programs in the chain outputs the same value, then the programs $\pi^2, \ldots, \pi^{a+1}$ in the configuration output the same value. Moreover, the configuration can be viewed as P_1 executing π_1^1, P_2 executing π_2^2 while the adversary who corrupts $\{P_3, P_4\}$ is simulating the remaining chain. Due to the validity property, π_2^2 must output V. Finally, each recipient's program $\pi^2, \ldots, \pi^{q+2}$ in the chain outputs V.

(iii) Let S_i^r be a random variable denoting the state of the program π^i in the chain after r rounds of the protocol execution. By state we understand the input that the program has, the set of all messages that the program received up to the r^{th} round over point-to-point channels and on the $\mathfrak{BO}$'s interface together with the random coins it has used. Let B^r be a random variable denoting the list of the values that have been broadcast with $\mathfrak{BO}$ up to the r^{th} round.

After r rounds only programs $\pi^1, \pi^2, \ldots, \pi^{r+1}$ can receive full information about V. The remaining programs in the chain $\pi^{r+2}, \pi^{r+3}, \ldots, \pi^{q+2}$ can receive only the information that was distributed with $\mathfrak{BO}$, i.e., the information contained in B^r. That is, one can verify by induction that for any r and for all $i \geq r+2$ holds $I(V; S_i^r | B^r) = 0$. Hence, for the last program in the chain π^{q+2} after q rounds of computation it holds that $I(V; S_{q+2}^q | B^q) = 0$ and hence $I(V; S_{q+2}^q) \leq H(B^q)$. Because we assumed that the protocol achieves non-trivial domain-amplification we have that $H(B^q) < H(V)$. Combining these facts we get that $I(V; S_{q+2}^q) < H(V)$. Hence, the last program π^{q+2} cannot output V with probability one, a contradiction.

(Case $n > 4$) Assume towards a contradiction that there is a protocol $(\pi_1, \pi_2, \pi_3, \ldots, \pi_n)$ allowing to do domain amplification in the single-

sender model. One particular strategy of the adversary is to corrupt parties $P_5, \dots, P_n$ and make them not execute their corresponding programs $\pi_5, \dots, \pi_n$. Still, the remaining protocol $(\pi_1, \pi_2, \pi_3, \pi_4)$ must achieve broadcast, which contradicts the first case. ■

3.5.3 Summary

Theorem 3.1 *If* $n = 3$ *then* $\forall d \geq 3$ $\phi_3(d) = 3$; *if* $n > 3$ *then* $\forall d$ $\phi_n(d) = d$.

The first statement follows from combining Lemma 3.1 and Lemma 3.2. The second statement follows from Lemma 3.3.

3.6 Multi-Sender Model

As we have seen in the previous section in the single-sender model no domain amplification is achievable for $n \geq 4$. In this section we consider a generalization of this model by allowing recipients to broadcast with $\mathfrak{BO}$ as well. In such a model we show that domain amplification is achievable for any n. Moreover, we prove that in order to achieve a non-trivial domain amplification for arbitrary n essentially all recipients must broadcast with $\mathfrak{BO}$.

3.6.1 Domain Amplification for n Parties

In this section we present a domain-amplification protocol for n parties, where the parties broadcast with $\mathfrak{BO}$ at most $8n \log n$ bits in total. First, we present a protocol for multi-graded broadcast, which achieves only a relaxed variant of broadcast, but only requires the sender to use $\mathfrak{BO}$. Then, we give the main domain-amplification protocol, which uses multi-graded broadcast and $\mathfrak{BO}$ (by each party) to achieve broadcast.

While the presented protocol is very efficient in terms of the $\mathfrak{BO}$ usage (it broadcasts via $\mathfrak{BO}$ only $8n \log n$ bits to achieve broadcast of any ℓ bits), it communicates exponentially many messages over authenticated channels. We then show how to optimize this protocol such that it communicates only a polynomial (in n) number of messages at the expense of a higher $\mathfrak{BO}$ usage.

Multi-Graded Broadcast

We extend the original gradecast primitive (cf. Section 2.2.2) with a more flexible grading system and define a new primitive called multi-graded broadcast (a similar grading extension "Proxcast" is given in [CFF⁺05]).

Definition 3.2 *A protocol achieves multi-graded broadcast if it allows the sender P_1 to distribute a value v among parties $\mathcal{R}$ with every party P_i outputting a value v_i with a grade $g_i \in [n]$ such that:*

VALIDITY: *If the sender P_1 is correct, then every correct $P_i \in \mathcal{R}$ outputs $(v_i, g_i) = (v, 1)$.*

GRADED CONSISTENCY: *If a correct $P_i \in \mathcal{R}$ outputs (v_i, g_i) with $g_i < n$, then every correct $P_j \in \mathcal{R}$ outputs (v_j, g_j) with $v_j = v_i$ and $g_j \leq g_i + 1$.*

TERMINATION: *Every correct party in $\mathcal{P}$ terminates.*

Intuitively, the grade can be understood as the consistency level achieved. The "strongest" grade $g_i = 1$ means that from the point of view of P_i, the sender "looks correct". Grade $g_i = 2$ means that P_i actually knows that the sender is incorrect; however, there might be an honest P_j for whom the sender looks correct. Grade $g_i = 3$ means that P_i knows that the sender is incorrect and every honest P_j knows so, too; however, there might be an honest P_k who does not know that every honest P_j knows that the sender is incorrect. And so on till the "weakest" grade $g_i = n$.

The protocol proceeds as follows: The sender sends the value v he wants to broadcast to all parties, who then exchange the received value(s) during $2n$ rounds. That is, in every round each party sends the set of values received so far to every other party. In this way each recipient P_i forms a growing sequence of sets $M_i^1 \subseteq M_i^2 \subseteq \cdots \subseteq M_i^{2n}$ (the set M_i^r represents the set of all messages received by P_i up to the round r). Finally, the sender distributes a hint consisting of the key k for an identifying predicate Q_k that should identify v among the values that the recipients hold. Then each recipient P_i computes his grade g_i to be the smallest number in $[n]$ such that both $M_i^{g_i}$ and $M_i^{2n-g_i}$ contain a uniquely identified message. There could be only one value v_i uniquely identified in both sets since $M_i^{g_i} \subseteq M_i^{2n-g_i}$. Then P_i outputs v_i with the grade g_i. Clearly, if the sender is correct, then each correct recipient outputs $g_i = 1$. Otherwise, since for every pair P_i, P_j of correct recipients it holds that $M_i^{g_i} \subseteq M_j^{g_i+1}$ and $M_j^{2n-(g_i+1)} \subseteq M_i^{2n-g_i}$ we have $g_j \leq g_i + 1$.

Let us detail the step when the sender distributes his hint k. While it can be done directly with the help of $\mathfrak{BC}$ (which would lead to a less efficient construction), we let the parties invoke gradecast recursively to distribute the k. Once each player P_i outputs a key k_i with a grade g'_i he uses k_i as a hint. Then the final grade is computed by P_i as the maximum of two grades g_i and g'_i, i.e., it is computed as the "weakest" grade among the two.

```
Protocol Multi-GradedBC($P_1, \mathcal{X}, v$)
  1. If $|\mathcal{X}| \le |\mathcal{K}^{\mathcal{X}}_{n^{2n}}|$ then $P_1$ broadcasts $v$ using $\mathfrak{BD}$, and every $P_i \in \mathcal{R}$
     outputs $(v, 1)$.
  2. Otherwise:
    2.1 Sender $P_1$: Set $M_1^0 := \{v\}$. $\forall P_i \in \mathcal{R}$: Set $M_i^0 := \varnothing$.
    2.2 For $r = 0, \ldots, 2n-1$:
          $\forall P_i \in \mathcal{P}$: Send $M_i^r$ (of size at most $n^r$) to all $P_j \in \mathcal{P}$, $P_j$
          denotes the union of the received sets with $M_j^{r+1}$, i.e.,
          $M_j^{r+1} = \bigcup_i M_i^r$.
    2.3 Sender $P_1$: Choose a key $k$ for the $n^{2n}$-identifying predicate
        $Q$ with domain $\mathcal{X}$, the set of values $M_1^{2n}$ and the value $v$.
    2.4 Players $\mathcal{P}$ invoke Multi-GradedBC($P_1, \mathcal{K}^{\mathcal{X}}_{n^{2n}}, k$) recursively.
        Let $(k_i, g'_i)$ denote the output of $P_i \in \mathcal{R}$.
    2.5 $\forall P_i \in \mathcal{R}$: Let $g$ be the smallest number in $[n]$ such that there
        exists $u$ which is uniquely identified by $Q_{k_i}$ in $M_i^g$ and in
        $M_i^{2n-g}$. Output $(v_i, g_i) = (u, \max(g, g'_i))$. If such $g$ does not
        exist output $(v_i, g_i) = (\perp, n)$.
```

Lemma 3.4 *The protocol* Multi-GradedBC *achieves multi-graded broadcast for $t < n$ while requiring only the sender to use $\mathfrak{BD}$ to broadcast one value of at most $\lceil 7n \log n \rceil$ bits and communicating $\mathcal{O}(n^{2n+3} \log |\mathcal{X}|)$ bits over bilateral channels.*

Proof: We prove by induction that multi-graded broadcast is achieved. For $|\mathcal{X}| \le |\mathcal{K}^{\mathcal{X}}_{n^{2n}}|$, multi-graded broadcast is achieved by assumption of $\mathfrak{BD}$. For $|\mathcal{X}| > |\mathcal{K}^{\mathcal{X}}_{n^{2n}}|$:

VALIDITY: If the sender is correct then he selects a key k for the n^{2n}-identifying predicate Q_k such that only his value v is identified by Q_k in M_1^{2n}. All correct players get $(k, 1)$ as output from the recursive call to Multi-GradedBC (due to the Validity property of the recursive Multi-GradedBC). Since for every correct player P_i it holds that $M_i^1 \subseteq M_i^{2n-1} \subseteq M_1^{2n}$ and $v \in M_i^1$ this implies that v is uniquely identified in M_i^1 and in M_i^{2n-1}. Hence P_i computes $v_i = v$ and $g_i = 1$.

GRADED CONSISTENCY: Let P_i denote a correct recipient outputting the smallest grade g_i. If $g_i = n$ then Graded Consistency holds trivially. Now assume that $g_i < n$, and hence $g_i' < n$. Consider any other correct recipient P_j. Due to the Graded Consistency property of the recursive Multi-GradedBC, the fact that $g_i' < n$ implies that P_i and P_j have the same keys k_i and k_j which we denote with k. Observe that $M_i^{g_i} \subseteq M_j^{g_i+1} \subseteq M_j^{2n-(g_i+1)} \subseteq M_i^{2n-g_i}$. The value v_i is uniquely identified by Q_k in both $M_i^{g_i}$ and $M_i^{2n-g_i}$, hence v_i is uniquely identified in both $M_j^{g_i+1}$ and $M_j^{2n-(g_i+1)}$. Hence the grade $g_j \in \{g_i, g_i+1\}$. If $g_j = g_i + 1$ then $v_j = v_i$. If $g_j = g_i$ then, since $M_j^{g_i} \subseteq M_j^{g_i+1}$ and v_i is uniquely identified in $M_j^{g_i+1}$, the only value that can be uniquely identified by Q_k in $M_j^{g_i}$ is v_i. This implies that $v_j = v_i$.

TERMINATION: Follows by inspection.

It remains to prove the stated communication complexity bounds. We denote the logarithm of broadcast domain size at the r^{th} recursive level to be ℓ_r. We have that $\ell_0 = \log|\mathcal{X}|$ and ℓ_{i+1} is defined recursively to be $2\lceil\log(n^{2n}\ell_i)\rceil$. It can be verified that $\ell_{i+1} < \ell_i$ for any $\ell_i > 7n\log n$. Note that $\mathfrak{BO}$ is only used at the deepest recursion level. Hence, the sender P_1 broadcasts with $\mathfrak{BO}$ at most $\lceil 7n\log n\rceil$ bits.

The communication costs of one iteration of Multi-GradedBC over bilateral channels consist of distributing sets M_i^r (of size at most n^r) for each $i \in \{1, \ldots, n\}$ and $r \in \{0, \ldots, 2n-1\}$. Hence, the total number of messages send in each iteration is at most n^{2n+1}. Each message at the i^{th} iteration is of ℓ_i bits. It can also be verified that $\ell_{i+1} < \ell_i/2$ for any $\ell_i > 14n\log n$. Hence the total number of bits communicated over bilateral channels is at most $\mathcal{O}(n^{2n+3}\log|\mathcal{X}|)$.

■

Main Protocol

The domain-amplification protocol first invokes multi-graded broadcast. Then, each party broadcasts his grade (using $\mathfrak{BO}$), and decides depending on the grades broadcast whether to use the output of multi-graded broadcast or to use some default value (say $\perp$) as output.

The core idea of the protocol lies in the analysis of the grades broadcast. Denote the set of all grades by $G = \{g_i\}_i$. As $|\mathcal{R}| = n-1$, there exists a grade $g \in [n]$ with $g \notin G$. Consider the smallest grade g_i of an honest party P_i. If $g_i > g$, then clearly the grade g_j of each honest party P_j is

$g_j > g$. On the other hand, if $g_i < g$, then by the definition of multi-graded broadcast, the grade g_j of any honest party P_j is $g_j \leq g_i + 1$, hence $g_j < g$. In other words, either the grades of all honest parties are below g, or the grades of all honest parties are above g. In the former case, every honest party P_i has $g_i < n$ and hence all values v_i are equal (and are a valid output of broadcast). In the latter case, no honest party P_i has grade $g_i = 1$, hence the recipients can output some default value $\perp$.

Protocol $\mathsf{AmplifyBC}_n(P_1, \mathcal{X}, v)$

1. Players $\mathcal{P}$ invoke $\mathsf{Multi\text{-}GradedBC}(P_1, \mathcal{X}, v)$, let (g_i, v_i) denote the output of P_i.
2. $\forall P_i \in \mathcal{R}$: Broadcast g_i using $\mathfrak{BD}$. Let G denote the set of all g_i broadcast.
3. $\forall P_i \in \mathcal{R}$: Let $g = \min([n] \setminus G)$. If $g_i < g$, then decide on v_i, otherwise decide on $\perp$.

Lemma 3.5 *The protocol* $\mathsf{AmplifyBC}_n$ *achieves broadcast for* $t < n$ *and requires the sender to broadcast with* $\mathfrak{BD}$ *one value of at most* $\lceil 7n \log n \rceil$ *bits and each of the recipients to broadcast one value from domain* $[n]$*. In total at most* $8n \log n$ *bits need to be broadcast via* $\mathfrak{BD}$ *and* $\mathcal{O}(n^{2n+3} \log |\mathcal{X}|)$ *bits need to be communicated over bilateral channels.*

Proof: We show that each of the broadcast properties are satisfied:

VALIDITY: If the sender is correct then all correct parties get $(v, 1)$ as an output from $\mathsf{Multi\text{-}GradedBC}$ and decide on v.

CONSISTENCY: Let $g = \min([n] \setminus G)$, and let P_i denote a correct recipient outputting the smallest grade g_i. If $g_i < g$, then clearly $g_i < n$, and all honest parties P_j hold the same value $v_j = v_i$ and grade $g_j \leq g_i + 1$. As $g_j \neq g$, it follows $g_j < g$. Hence, every honest party P_j outputs $v_j = v_i$. On the other hand, if $g_i > g$, then every honest party P_j holds $g_j > g$ and outputs $\perp$.

TERMINATION: Follows by inspection.

It remains to prove the stated communication complexity of the protocol. The protocol $\mathsf{AmplifyBC}_n$ requires the sender to broadcast one value of at most $\lceil 7n \log n \rceil$ bits during the $\mathsf{Multi\text{-}GradedBC}$ invocation (cf. Lemma 3.4). Furthermore, each recipient broadcasts the grade (of domain $[n]$) using $\mathfrak{BD}$. This sums up to at most $8n \log n$ bits overall. The protocol $\mathsf{AmplifyBC}_n$ uses bilateral channels only during the invocation of $\mathsf{Multi\text{-}GradedBC}$. Hence, $\mathsf{AmplifyBC}_n$ communicates at most $\mathcal{O}(2^{2n+3} \log |\mathcal{X}|)$ bits over bilateral channels. ■

3.6.2 A Protocol with Polynomial Communication

The main disadvantage of the protocol AmplifyBC$_n$ is that the underlying gradecast protocol Multi-GradedBC requires exponential message communication. Here we briefly sketch how one can achieve polynomial communication complexity in Multi-GradedBC at the cost of higher $\mathfrak{BO}$ usage. The main idea of the new protocol Multi-GradedBC$^+$ is to allow recipients to use $\mathfrak{BO}$ such that they can filter out messages from the sets M_i^r. Roughly speaking, if a recipient holds a set of messages M_i^r then he broadcasts a "challenge" forcing the sender in his response to invalidate at least all but one values in M_i^r. After each of the recipients has his set M_i^r filtered, recipients continue exchanging sets consisting of at most one element.

Lemma 3.6 *The protocol* AmplifyBC$_n$ *with the underlying gradecast implementation by* Multi-GradedBC$^+$ *achieves broadcast of messages from $\mathcal{X}$ for $t < n$ while broadcasting $\mathcal{O}(n^2 \log\log|\mathcal{X}|)$ bits with $\mathfrak{BO}$ and communicating $\mathcal{O}(n^3 \log|\mathcal{X}|)$ bits over point-to-point channels.*

The Protocol Multi-GradedBC$^+$

The protocol proceeds as follows: The sender sends the value v he wants to broadcast among all recipients $\mathcal{R}$, who then exchange the received value(s) during $n-1$ rounds. In each round r each party P_i sends the value it currently holds (denoted with v_i^r) to every other party and forms the set of at most n received values. Then each party broadcasts a key k_i for an n-resolution function which resolves the set of the values received. In the end of the round the sender broadcasts the values $y_1, \ldots, y_n$ of the resolution function for the keys $k_1, \ldots, k_n$ so that each of the recipients keeps at most one value identified by all (k_i, y_i). Finally, each recipient P_i decides on the grade g_i to be the first "stable" round starting from which the value he holds remain unchanged, i.e., $v_i^{g_i} = v_i^{g_i+1} = \cdots = v_i^n$.

Protocol Multi-GradedBC$^+(P_1, \mathcal{X}, v)$:

```
1.  Sender $P_1$: Send $v$ to every $P_i \in \mathcal{R}$.
    $\forall P_i \in \mathcal{R}$: Denote the message received from the sender by $v_i^1$.
r.  In each step $r = 2, \ldots, n$, execute the following sub-steps:
    r.1  $\forall P_i \in \mathcal{R}$: Send the value $v_i^{r-1}$ to all $P_j \in \mathcal{R}$, $P_j$ denotes the
         set of the received values with $M_j^r$.
```

$r.2$ $\forall P_i \in \mathcal{R}$: Choose a key k_i^r for an n-resolution function F with domain $\mathcal{X}$, that resolves the set of values S_i^r. Broadcast the key k_i^r using the $\mathfrak{BO}$.
$r.3$ Sender P_1: Broadcast a list of values $(F_{k_2^r}(v), \dots, F_{k_n^r}(v))$ using the $\mathfrak{BO}$. Denote the list broadcast with $(y_2^r, \dots, y_n^r)$.
$r.4$ $\forall P_i \in \mathcal{R}$: Select v_i^r to be some $u \in M_i^r$ such that u is identified by (k_j^r, y_j^r) for all j; set v_i^r to $\perp$ if no such u exists.
$n{+}1.$ $\forall P_i \in \mathcal{R}$: Compute g_i to be the smallest step r such that $v_i^r = v_i^{r+1} = \cdots = v_i^n$. Output (v_i^n, g_i).

Lemma 3.7 *The protocol* Multi-GradedBC$^+$ *achieves multi-graded broadcast for $t < n$ while requiring $\mathcal{O}(n^2 \log\log|\mathcal{X}|)$ bits to be broadcast with $\mathfrak{BO}$ and communicating $\mathcal{O}(n^3 \log|\mathcal{X}|)$ bits over point-to-point channels.*

Proof: We show that each of the multi-graded broadcast properties is satisfied:

VALIDITY: If the sender is correct then for any key k he broadcasts $y = F_k(v)$ such that only his value v is chosen by correct recipients at every iteration. Hence each correct P_i computes $v_i = v$ and $g_i = 1$.

GRADED CONSISTENCY: Let P_i denote a correct recipient outputting the smallest grade g_i. If $g_i = n$ then Graded Consistency holds trivially. Now assume that $g_i < n$. Consider any other correct recipient P_j. Observe that $v_i^{g_i} \in M_j^{g_i+1}$. Since P_i kept the value $v_i^{g_i}$ till the round n it implies that $v_i^{g_i}$ is identified in S_i^r by all pairs (k_a^r, y_a^r) for all a and $r \geq g_i$. Hence, $v_i^{g_i}$ is identified in $M_j^{g_i+1}, M_j^{g_i+2}, \dots, M_j^n$ by all pairs (k_a^r, y_a^r) for all a and $r \geq g_i + 1$. Moreover, since P_j chose the keys k_j^r for a resolution function faithfully, only a unique value can be identified in $M_j^{g_i+1}, M_j^{g_i+2}, \dots, M_j^n$. Since only a unique value can be identified then P_j outputs $v_j = v_i^{g_i}$ with the grade $g_j \leq g_i + 1$.

TERMINATION: Follows by inspection.

It remains to prove the stated communication complexity of the protocol. Let $\ell = \log|\mathcal{X}|$. At each step $r = 2, \dots, n$ of the protocol Multi-GradedBC$^+$, every recipient P_i broadcasts a key k_i^r for a family of n-resolution functions and the sender broadcasts a list of n values of the function. If the polynomial-based construction of the resolution function is used then each key and a value of function consists of $\lceil \log(n^2\ell) \rceil$ bits. Since in total the protocol works in $n - 1$ rounds we broadcast $2(n-1)^2 \lceil \log(n^2\ell) \rceil$ bits with $\mathfrak{BO}$. This expression can be rewritten as $\mathcal{O}(n^2(\log n + \log \ell))$. We can assume that $\ell > n$, since for $\ell \leq n$ it is easier to run the trivial algorithm that broadcasts a message bit by bit. Summing up the analysis above, we have that the total number of the $\mathfrak{BO}$ invocations during the protocol run is $\mathcal{O}(n^2 \log \ell)$.

The communication costs of $\mathsf{Multi\text{-}GradedBC}^+$ over bilateral channels consist of distributing $n-1$ ℓ-bit messages during Step 1 and exchanging of $(n-1)^2(n-1)$ ℓ-bit messages during Steps $2, \dots, n$. Hence, the total number of bits that need to be communicated is $\mathcal{O}(n^3\ell)$. ■

3.6.3 Lower Bounds in the Multi-Sender Model

Based on the approach presented in Section 3.3 we investigate the lower bounds on the domain-amplification protocols in the multi-sender model. As it was shown for the single-sender model there is no domain-amplification possible when only the sender uses $\mathfrak{BO}$ for $n \geq 4$. We extend this result by showing that the sender and at least all but 2 recipients are required to broadcast some information via $\mathfrak{BO}$ to achieve non-trivial domain-amplification.

Lemma 3.8 *Every perfectly-secure domain-amplification protocol tolerating $t < n$ requires the sender P_1 to broadcast at least 1 bit via $\mathfrak{BO}$.*

Proof: We first prove the theorem for $n = 3$, then reduce the case of arbitrary $n > 3$ to $n = 3$.

(Case $n = 3$) Assume towards a contradiction that there is a protocol (π_1, π_2, π_3) allowing the parties P_1, P_2, P_3 to do domain amplification, where the sender does not broadcast with $\mathfrak{BO}$. We consider the following configuration: Let π_1^u and π_1^v denote two instances of the program π_1, where π_1^u is given input u and π_1^v is given input v for $u \neq v$ chosen from the broadcast input domain $\mathcal{X}$. We connect programs π_1^u, π_2, π_3 and π_1^v with bilateral channels as shown in Figure 3.4. Now we execute the programs. Whenever π_2 or π_3 use $\mathfrak{BO}$ to broadcast some x, the value x is given to all recipients' programs, i.e., to programs π_1^u, π_1^v, π_3 when π_2 broadcasts, and to programs π_1^u, π_1^v, π_2 when π_3 broadcasts .

The configuration can be interpreted in three different ways, which lead to contradicting requirements on the outputs of the programs. (i) P_1 holds input u and executes π_1^u, P_2 executes π_2, and P_3 is corrupted and executes π_3 and π_1^v. Due to the validity property, π_2 must output u. (ii) P_1 holds input v and executes π_1^v, P_3 executes π_3, and P_2 is corrupted and executes π_2 and π_1^u. Due to the validity property, π_3 must

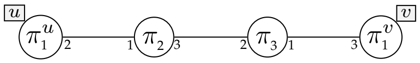

Figure 3.4: The configuration for $n = 3$ to show that the sender must use $\mathfrak{BD}$ to broadcast at least one bit

output v. (iii) P_2 executes π_2, P_3 executes π_3, and P_1 is corrupted and executes π_1^u and π_1^v. Due to the consistency property, π_2 and π_3 must output the same value. These three requirements cannot be satisfied simultaneously, hence whatever output the programs make, the protocol (π_1, π_2, π_3) is not a perfectly-secure domain-amplification protocol.

(Case $n > 3$) Assume towards a contradiction that there is a protocol $(\pi_1, \pi_2, \pi_3, \ldots, \pi_n)$ allowing to do domain amplification where the sender does not broadcast with $\mathfrak{BD}$. One particular strategy of the adversary is to corrupt parties $P_4, \ldots, P_n$ and make them not execute their corresponding programs $\pi_4, \ldots, \pi_n$. Still, the remaining protocol (π_1, π_2, π_3) must achieve broadcast, which contradicts the first case. ∎

Lemma 3.9 *Every perfectly-secure non-trivial domain-amplification protocol for $t < n$ requires that at least all but 2 of the recipients broadcast at least 1 bit with $\mathfrak{BD}$.*

Proof: Assume towards a contradiction that there is a protocol $(\pi_1, \pi_2, \pi_3, \ldots, \pi_n)$ allowing to do non-trivial domain amplification with three recipients' programs not broadcasting with $\mathfrak{BD}$. Without loss of generality, assume that these programs are π_2, π_3, π_4.[12] One particular strategy of the adversary is to corrupt parties $P_5, \ldots, P_n$ and make them not execute their corresponding programs $\pi_5, \ldots, \pi_n$. The programs $\pi_1, \pi_2, \pi_3, \pi_4$ of the remaining honest players can then put the values sent and broadcast by the corrupted parties to some default value (say $\perp$). The remaining protocol $(\pi_1, \pi_2, \pi_3, \pi_4)$ achieves non-trivial domain amplification, which contradicts Lemma 3.3. ∎

3.6.4 Summary

The following theorem summarizes results obtained in this section (The proof of this theorem follows from Lemmas 3.5, 3.8 and 3.9.)

[12]Such not broadcasting programs' indices are fixed because we considered protocols with static $\mathfrak{BD}$ usage pattern.

Theorem 3.2 *For all n, d we have* $8n \log n \geq \log \phi_n^*(d) \geq \min(\log d, n-2)$.[13]

Additionally, we gave an efficient protocol that allows to broadcast an ℓ-bit value while broadcasting $\mathcal{O}(n^2 \log \ell)$ bits with $\mathfrak{BO}$ and communicating $\mathcal{O}(n^3 \ell)$ bits over point-to-point channels.

3.7 On The Round Complexity of Domain-Amplification Protocols

While the primary goal of this chapter is to minimize the $\mathfrak{BO}$ usage, one often optimizes the protocols with respect to another measure of the protocols' efficiency, namely, number of rounds employed by a protocol. According to this measure there are two principal classes of protocols: constant-round and non-constant round. In the following we investigate whether it is possible to obtain non-trivial constant-round domain-amplification protocols.

Theorem 3.3 *There is no perfectly-secure constant-round non-trivial domain-amplification protocol for $t < n$ in the multi-sender model.*

Proof: Take any non-trivial domain-amplification protocol $(\pi_1, \ldots, \pi_n)$ which terminates in some constant q number of rounds. Consider n such that $q < n-1$ (e.g., set $n := q+2$). On the highest level our proof consists of three steps. (i) we define a configuration (inspired by [GKKO07]). (ii) we show that all recipients' programs in the configuration must output the same value v. (iii) we use an information flow argument to prove that there is a program in the configuration that does not have enough information to output v with probability 1.
(i) Consider a chain of n programs $\pi_1, \pi_2, \pi_3, \ldots, \pi_n$ connected with bilateral channels as shown in Figure 3.5. In this configuration only programs that are connected communicate, i.e., π_1 communicates only with π_2 and receives no messages from parties in $\mathcal{P} \setminus \{P_1, P_2\}$. Let π_1 be given as input a uniform random variable V chosen from the broadcast input domain $\mathcal{X}$. Now we execute the programs. Whenever any program π_i broadcasts any value using the $\mathfrak{BO}$ this value is delivered to all recipients' programs in the configuration (i.e., to all π_j such that $j \neq i$).
(ii) First, we prove that any pair of connected recipients' programs (π_i, π_{i+1}) $(i \geq 2)$ in the chain outputs the same value. One can view

[13]The last inequality combines the facts that any non-trivial domain amplification protocol broadcasts at least $n-2$ bits, whereas the trivial protocol always uses $\log d$ bits.

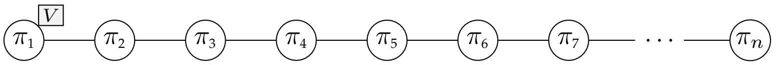

Figure 3.5: The configuration to show the impossibility of non-trivial construction

the configuration as the player P_i running the program π_i and P_{i+1} running π_{i+1} while the adversary corrupting $\mathcal{P} \setminus \{P_i, P_{i+1}\}$ is simulating the programs $\pi_1, \ldots, \pi_{i-1}$ and $\pi_{i+2}, \ldots, \pi_n$. Due to the consistency property, π_i and π_{i+1} must output the same value. Since every connected pair of recipients' programs in the chain outputs the same value, then all the recipients' programs in the configuration output the same value. Moreover, the configuration can be viewed as P_1 executing π_1 and P_2 executing π_2 while the adversary corrupts $\mathcal{P} \setminus \{P_1, P_2\}$ and simulates the remaining programs. Due to the validity property, π_2 must output V. Finally, the recipients' programs $\pi_2, \ldots, \pi_n$ output V.
(iii) Let S_i^r be a random variable denoting the state of the program π_i in the chain after r rounds of the protocol execution. By state we understand the input that the program has, the set of all messages that the program received up to the r^{th} round over point-to-point channels and via the underlying broadcast procedure together with the random coins it has used. Let B^r be a random variable denoting the list of the values that have been broadcast with the broadcast procedure up to the r^{th} round.
After r rounds only programs $\pi_1, \pi_2, \ldots, \pi_{r+1}$ can receive full information about V. The remaining programs in the chain $\pi_{r+2}, \pi_{r+3}, \ldots, \pi_n$ can receive only the information that was distributed with $\mathfrak{BO}$, i.e., the information contained in B^r. That is, one can verify by induction that for any r and for all $i \geq r+2$ holds $I(V; S_i^r | B^r) = 0$. Hence, for the last program in the chain π_n after q rounds of computation it holds that $I(V; S_n^q | B^q) = 0$ and hence $I(V; S_n^q) \leq H(B^q)$. Because we assumed that the construction is non-trivial, we have that $H(B^q) < H(V)$. Combining these facts we get that $I(V; S_n^q) \leq H(V)$. Hence, the last program π_n cannot output V with probability one, a contradiction. ∎

3.8 Impossibility Proofs for Dynamic Usage Pattern

Dynamic usage patterns allow for protocols that adaptively decide during the protocol execution which of the underlying broadcast channels are used. In the single-sender model dynamic usage patterns allow to adaptively decide on the round where the sender broadcasts with $\mathfrak{BO}$ and on the domain of the value broadcast. In the multi-sender model dynamic usage patterns allow to decide additionally on the party using $\mathfrak{BO}$ as sender.

In this section we show that the lower bounds we proved for the case of the static usage pattern hold for the case of dynamic usage pattern as well.[14] We note that all the domain-amplification protocols we gave have static usage patterns. Whether or not dynamic usage patterns allow for a more efficient domain-amplification protocols is an open question.

3.8.1 Single-Sender Model

The impossibility proofs in the single-sender model for protocols with static usage patterns (Lemmas 3.2 and 3.3) directly apply to the protocols with dynamic usage patterns by the following observation: It can be easily seen that all programs in the configurations considered in the proofs agree on the $\mathfrak{BO}$ usage pattern.

3.8.2 Multi-Sender Model

In the multi-sender model, the impossibility proofs for the protocols with static usage patterns (Lemmas 3.8 and 3.9) do not directly carry over to the dynamic usage patterns setting. Lemmas 3.8 and 3.9 state that in any non-trivial domain-amplification protocol (i) the sender must broadcast with $\mathfrak{BO}$ and (ii) at least all but 2 recipients must broadcast with $\mathfrak{BO}$. There are two shortcomings when reusing the static usage pattern proofs of these statements for protocols with dynamic usage patterns. First, one needs to show only that there is *some* execution where the corresponding statement holds (i.e., the sender broadcasts with $\mathfrak{BO}$, or all but 2 recipients broadcast with $\mathfrak{BO}$). Second, if we prove these two lemmas

[14]Because the costs of $\mathfrak{BO}$ usage for the protocols with dynamic usage patterns are computed as the highest $\mathfrak{BO}$ usage among all executions, there could still exist protocols with dynamic usage patterns which have $\mathfrak{BO}$ usage costs smaller than the lower bounds in some executions. Proving lower bounds for all executions is left as an open question.

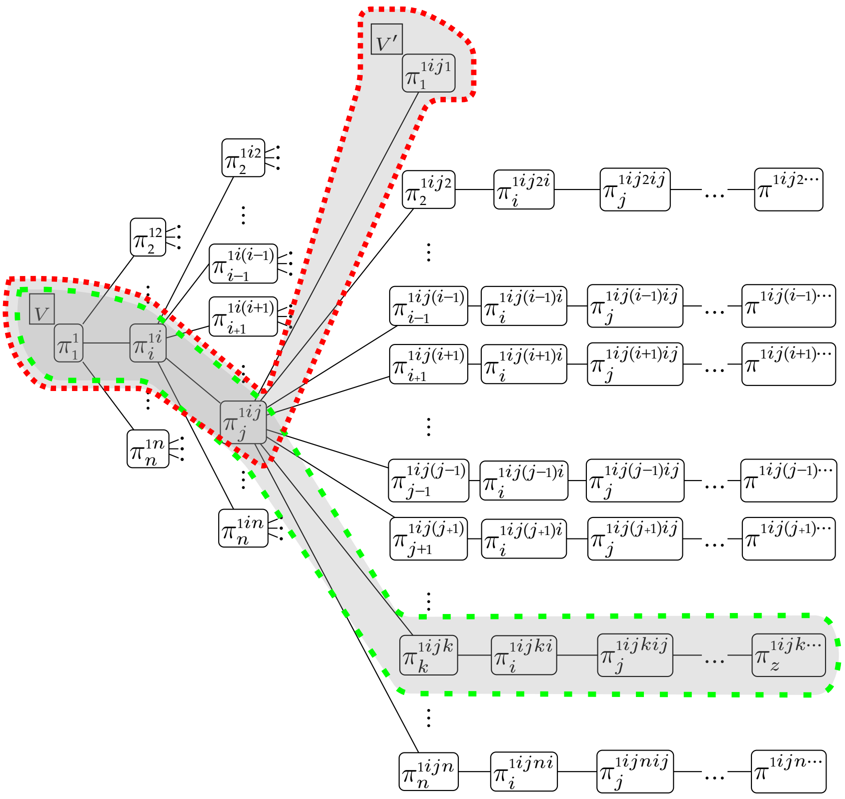

Figure 3.6: The configuration to show that at least all but 2 parties must use $\mathfrak{BD}$ to broadcast at least one bit. The region 🟢 describes a part of the configuration responsible for proving that at least all but 2 recipients must broadcast with $\mathfrak{BD}$ (a clone of the configuration from Figure 3.5). The region 🔴 describes a part of the configuration responsible for proving that the sender must broadcast with $\mathfrak{BD}$ (a clone of the configuration from Figure 3.4). The labels of the bilateral channels' interfaces on the picture are not drawn to simplify the picture (Each pair of the copies of the programs π_i and π_j is connected on interfaces j and i, respectively.)

individually, we show that any protocol achieving non-trivial domain-amplification must have *some* executions where the sender broadcasts with $\mathfrak{BO}$ and *some* executions where all but 2 recipients broadcast with $\mathfrak{BO}$. Combining these two statements does not necessarily imply that there is an execution where the sender *and* all but 2 recipients broadcast with $\mathfrak{BO}$. In the following we prove that there exists an execution where all but 2 parties broadcast with $\mathfrak{BO}$.

Lemma 3.10 *Every perfectly-secure non-trivial domain-amplification protocol for $t < n$ requires that at least all but 2 parties broadcast with $\mathfrak{BO}$.*

Proof: Assume towards a contradiction that there exists a perfectly-secure protocol $(\pi_1, \dots, \pi_n)$ achieving non-trivial domain-amplification in the multi-sender model in q rounds (for some $q \in \mathbb{N}$) where in every execution at most all but 3 parties broadcast with $\mathfrak{BO}$. Our proof consists of five steps: First, we define a configuration of the given programs. Second, we define the execution of the configuration and introduce the notion of *surviving* programs, i.e., programs that terminate properly. Third, we show that all surviving programs must output the same value v. Fourth, we show that the sender must use $\mathfrak{BO}$ in every execution of the configuration. Fifth, we use an information flow argument to prove that there is a surviving program in the configuration that does not have enough information to output v with probability 1.
(Defining the configuration) We consider the following configuration (see Figure 3.6): Let π_i^w (where w is the sequence of integers) denote an instance of the program π_i. The programs in the configuration are put in a tree as follows.

1. In the root of the tree we put program π_1^1.
2. On the second level of the tree we put programs π_i^{1i} $(i \in [n] \setminus \{1\})$ connected via bilateral channels to the root program π_1^1 on the corresponding interfaces.
3. Each program π_i^{1i} from the second level is connected via bilateral channels to the programs π_j^{1ij} $(j \in [n] \setminus \{1, i\})$ on the third level on the corresponding interfaces.
4. Each program π_j^{1ij} from the third level is connected via bilateral channels to the programs π_k^{1ijk} $(k \in [n] \setminus \{1, i, j\})$ on the fourth level on the corresponding interfaces. Furthermore, each program π_j^{1ij} is connected to new a copy of the sender's program π_1^{1ij1} on the corresponding interface.

5. Each program π_k^{1ijk} is connected via bilateral channel to a chain of programs $\pi_i^{1ijki}, \pi_j^{1ijkij}, \ldots, \pi_z^w$ (where w is the integer sequence of the length $q+3$ formed by concatenation of 1 and i, j, k repeatedly; and z is the last element of w).

In such a tree we will refer to the programs π_j^w with $|w| = l$ and $j \neq 1$ as to the programs on the l^{th} level of the tree.

(Executing the configuration and survival set structure) The configuration is executed by giving π_1^1 as input a uniform random variable V chosen from domain $\mathcal{X}$. For a random variable V let V' be a random variable such that $\Pr[V \neq V'] = 1$ (e.g., if $\mathcal{X} = [d]$ then $V' = V + 1 \bmod d$). Let each program π_1^{1ij1} be given V' as input. In order to define the configuration's behavior in each round we introduce the set of the survived programs T^r up to the round r.[15] We execute the configuration during q rounds as follows:

1. In the beginning T^1 is defined to be the set of all programs in the configuration.

2. In round r (for $r = 1, \ldots, q$) we let all the programs in T^r execute and prepare inputs for point-to-point channels and $\mathfrak{BD}$. Whenever a program from T^r inputs a message into a point-to-point channel the message is delivered to the program at the other end of the channel. If (r, P_i, d)-broadcast is accessed by some of the survived programs, then depending on P_i the following happens: If $P_i = P_1$ then all the programs in T^r receive the value which was input to $\mathfrak{BD}$ by the program π_1^1; otherwise, all the programs in T^r receive the default value from the broadcast domain (say $\perp$) as a result of the $\mathfrak{BD}$ call.[16]

3. The set T^{r+1} is defined to be the set T^r modified by excluding some of the programs as follows. If (r, P_i, d)-broadcast was accessed by the survived programs with $P_i = P_1$ then the programs π_1^{1ij1} ($\forall i, j \in [n] \setminus \{1\}$) are excluded from T^{r+1}; otherwise, if $P_i \neq P_1$ then the programs π_j^u with u containing i are excluded from T^{r+1}.

By construction we have that (i) the survival set T^r is a subtree of the initial configuration with the program π_1^1 in its root and (ii) the programs in T^r (except π_1^1) have not used $\mathfrak{BD}$ up to the round r.[17] Now consider

[15]Note that T^r is a random variable depending on the execution.

[16]We show later that all the programs in T^r agree on which (r, P_i, d)-broadcast channels to use, that is, they agree on the $\mathfrak{BD}$ usage pattern.

[17]It may be that some programs in T^r use $\mathfrak{BD}$ in round r to broadcast some values, but this values will be delivered only in round $r+1$.

any pair of programs π_i^w and π_j^u in T^r with $u = w||j$. Up to round r the configuration can be seen as P_i executing π_i^w, P_j executing π_j^u and the adversary corrupting $\mathcal{P} \smallsetminus \{P_i, P_j\}$ simulating the remaining programs in the configuration. This simulation is possible because either $P_1 \notin \{P_i, P_j\}$ and due to (ii) P_i and P_j do not broadcast with $\mathfrak{BO}$; or $P_1 \in \{P_i, P_j\}$ and the broadcast(s) of P_1 have been correctly delivered to the other party in $\{P_i, P_j\}$. Since P_i and P_j must agree on the $\mathfrak{BO}$ usage pattern so do programs π_i^w and π_j^u. Hence, because of (i) all programs in the survival set T^r must agree on the same $\mathfrak{BO}$ usage pattern up to the round r.
(All programs in T^q must output V) Consider any program π_i^w in the set T^q. We prove by induction on the program's level $|w|$ that it must output V.

(Base $|w| = 2$) Consider any program π_i^{1i} on the second level of the tree which belongs to T^q. Then, the configuration can be viewed as P_1 executing π_1^1 and P_i executing π_i^{1i} while the adversary who corrupts $\mathcal{P} \smallsetminus \{P_1, P_i\}$ is simulating the remaining programs. Due to the validity property, π_i^{1i} must output V.

(Induction step $|w| \to |w|+1$) Consider any program π_j^u with $|u| = |w|+1$. Since T^q is a subtree there exists π_i^w from T_q with $u = w||j$. One can view the configuration as players P_i and P_j running π_i^w and π_j^u while the adversary corrupting $\mathcal{P} \smallsetminus \{P_i, P_j\}$ is simulating the remaining programs. Due to the consistency property, π_i^w and π_j^u must output the same value. Due to the induction hypothesis, π_i^w outputs V and hence π_j^u outputs V as well.
(Sender must broadcast with $\mathfrak{BO}$) Assume there exists an execution where the sender does not use $\mathfrak{BO}$. We assumed that at most all but 3 parties broadcast with $\mathfrak{BO}$, hence, there are two recipients P_i and P_j who did not broadcast with $\mathfrak{BO}$. Consider the programs π_j^{1ij} and π_1^{1ij1}. The configuration can be interpreted as follows: P_1 holds input V' and executes π_1^{1ij1}, P_j executes π_j^{1ij}, and $\mathcal{P} \smallsetminus \{P_1, P_j\}$ are corrupted and simulate the remaining programs in the configuration. Due to the validity property, π_j^{1ij} must output V'. Since we assumed that P_1, P_i and P_j do not broadcast with $\mathfrak{BO}$, we have that π_j^{1ij} belongs to T^q. As shown above, this implies that π_j^{1ij} must also output $V \neq V'$, a contradiction.
(Information flow argument) Since we assumed that at least all but 3 parties do not broadcast with $\mathfrak{BO}$ and as we showed above the sender broadcasts with $\mathfrak{BO}$ in all executions, there are three recipients P_i, P_j and P_k who did not broadcast with $\mathfrak{BO}$. This implies that in every execution the set T^q contains a program π_z^w with $w = 1ijkijki\cdots$ of the length $q+3$ and z being the last integer in the sequence w. Below we show that such

a program π_z^w does not have enough information to output V.

Let S_l^r be a random variable denoting the joint view of the programs on the l^{th} level after r rounds of the protocol execution which are still in the survival set T^r (the description of the set T^r is included in S_l^r). The joint view includes the set of all messages that the programs received up to the round r over point-to-point channels and on the $\mathfrak{BO}$'s interface together with the random coins they have used. Let B^r be a random variable denoting the list of the values that have been broadcast with $\mathfrak{BO}$ up to the round r.

After r rounds only the programs on the levels $1, \ldots, r+2$ can receive full information about V (via bilateral channels).[18] The remaining programs on the levels $r+3, \ldots, q+3$ can receive only the information that was distributed with $\mathfrak{BO}$, i.e., the information contained in B^r. That is, one can verify by induction that for any r and for all $l \geq r+3$ holds $I(V; S_l^r | B^r) = 0$. Hence, for the last level of the programs in the configuration it holds that $I(V; S_{q+3}^q | B^q) = 0$ and, consequently, $I(V; S_{q+3}^q) \leq H(B^q)$. Because we assumed that the protocol achieves non-trivial domain-amplification we have that $H(B^q) < H(V)$. Combining these facts we get that $I(V; S_{q+3}^q) < H(V)$. Hence, survived programs on the last level cannot output V with probability 1, a contradiction. ■

3.8.3 Round Complexity Theorem

The impossibility proof given in Theorem 3.3 is valid in both static and dynamic usage pattern models. This can be easily seen since all programs in the configuration considered must agree on the $\mathfrak{BO}$ usage pattern.

[18]We need to consider $r+2$ levels (as opposed to $r+1$ levels in Lemma 3.3), because the programs at level $r+1$ might learn partial information on V from π_1^{1ij1} (namely, that $V' \neq V$).

Chapter 4

Communication Optimal Multi-Valued Broadcast for $t < n$

In this chapter we construct two protocols tolerating $t < n$ that achieve communication optimal multi-valued broadcast of ℓ-bit messages from bilateral channels and a trusted setup with cryptographic and information-theoretic security, respectively. We build each of the protocols in two steps. First, we construct a domain-amplification protocol that for broadcasting an ℓ-bit message communicates $\mathcal{O}(\ell n)$ bits over bilateral channels and broadcasts a "small" number of bits via $\mathfrak{BO}$. Second, in order to obtain a concrete protocol for multi-valued broadcast we instantiate the $\mathfrak{BO}$ oracle employed in the domain-amplification protocol with a protocol tolerating $t < n$ that achieves binary broadcast from bilateral channels and a trusted setup. As a result of the two protocols composition we obtain a protocol that achieves multi-valued broadcast with optimal communication complexity from bilateral channels and a trusted setup.

Our first multi-valued protocol is build on top of the cryptographically secure domain-amplification construction that broadcasts $\mathcal{O}(n^2 + n\kappa)$ bits with $\mathfrak{BO}$, where κ is a security parameter. After substituting $\mathfrak{BO}$ with [DS83] we obtain a concrete multi-valued broadcast protocol that communicates $\mathcal{O}(\ell n)$ bits in order to broadcast a sufficiently long message.

Our second multi-valued protocol is build on top of the information-theoretically secure domain-amplification construction that broadcasts $\mathcal{O}(n^4 + n^3\kappa)$ bits with $\mathfrak{BO}$, where κ is a security parameter. After substituting $\mathfrak{BO}$ with [PW96] we obtain a concrete multi-valued broadcast protocol that communicates $\mathcal{O}(\ell n)$ bits in order to broadcast a sufficiently long message.

The contents of this chapter are based on [HR13a].

4.1 Protocols Overview

We present cryptographically and information-theoretically secure domain-amplification protocols. Both constructions are built over point-to-point channels and $\mathfrak{BO}$.

On the highest level both constructions broadcast the long message block by block, where each block is broadcast using a special protocol for block broadcast. This block broadcast protocol achieves optimal communication complexity over bilateral channels only in *good* executions, while in *bad* executions more bits need to be communicated. We select the number of blocks in such a way that good executions outnumber bad ones and that the total communication complexity is optimal. Whether an execution is good or bad is determined using the *Dispute Control Framework* [BH06, BHR07]. Dispute control is a technique which keeps track of disputes (also called conflicts) between players and ensures that occurred disputes cannot show up again. Intuitively, an execution is good if it is dispute-free, and bad otherwise.

We employ the dispute control framework as follows. We consider a set of unordered pairs of parties Δ, where $\{P_i, P_j\} \in \Delta$ represents the fact that parties P_i and P_j accuse each other of being Byzantine. Parties start a protocol by setting Δ to be the empty set. Then during the protocol run they add new disputes to Δ when they learn about new accusations. We ensure that Δ always remains *valid*, meaning that if $\{P_i, P_j\} \in \Delta$ then at least one of the players P_i, P_j is Byzantine.

4.2 Cryptographically Secure Construction

First, we present a protocol CryptoBlockBC for broadcasting blocks that makes use of a set of disputes Δ. Then we plug CryptoBlockBC in the protocol CryptoBC, which broadcasts an ℓ-bit message block by block q times. In each invocation of CryptoBlockBC we will use the same global variable Δ with the disputes among the players. This means that if parties P_i and P_j conflict during some block broadcast, then they conflict in all later invocations of CryptoBlockBC. Then, we count the communication complexity of the resulting construction and select q which makes its optimal.

4.2.1 Block Broadcast Protocol CryptoBlockBC

The protocol CryptoBlockBC employs a collision-resistant hash function `CRHash`. In the beginning of the protocol the sender broadcasts a hash $h = \texttt{CRHash}(v)$ of the value it holds with $\mathfrak{BO}$. The goal of the protocol is to ensure that all correct players learn v (or collectively abort by agreeing on $\perp$). All parties during the protocol run are divided into two sets: H and $\overline{H}$. The set H consists of happy players who have already learned v_s, and $\overline{H}$ consists of the remaining players. At each iteration of CryptoBlockBC we try to move a player from $\overline{H}$ to H. We select a pair of players P_x, P_y such that $P_x \in H$ and $P_y \in \overline{H}$. Then P_x sends the value it holds to P_y. This procedure is meaningless if parties P_x, P_y are in the dispute, so the pair is chosen such that $\{P_x, P_y\} \notin \Delta$. Once P_y receives a value from P_x it verifies that its hash is h; in the positive case P_y is included in H and in the negative case a conflict between P_x and P_y is found. Hence at each iteration we either include one player into H or we discover a new conflict between a pair of players.

```
Protocol CryptoBlockBC($P_1, v, \Delta$):
  1. Parties initialize happy set $H$ to be $\{P_1\}$.
  2. Sender $P_1$: set $v_1 := v$ and broadcast $h := \texttt{CRHash}(v)$ with $\mathfrak{BO}$.
  3. While $\exists\, P_x, P_y \in \mathcal{P}$ s.t. $P_x \in H$ and $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$ do
       $r.1$ $P_x$: Send $v_x$ to player $P_y$. Denote received value by $v_y$.
       $r.2$ $P_y$: If $h = \texttt{CRHash}(v_y)$ broadcast 1, else broadcast 0 with $\mathfrak{BO}$.
       $r.3$ If $P_y$ broadcasted 1 then parties add $P_y$ to $H$, otherwise they add $\{P_x, P_y\}$ to $\Delta$.
  4. $\forall P_i \in \mathcal{R}$: If $P_i \in H$ decide on $v_i$, otherwise decide on $\perp$.
```

Lemma 4.1 *Given that the initial dispute set Δ_s is valid and `CRHash` is a collision-resistant hash function, then the protocol* CryptoBlockBC *achieves broadcast (of v) and terminates with a valid dispute set Δ_e. Furthermore, the protocol terminates in $\mathcal{O}(n+d)$ rounds communicating at most $(|h|+n+d)$ bits* *with $\mathfrak{BD}$ and $(n+d)|v|$ bits over bilateral channels, where $d = |\Delta_e| - |\Delta_s|$, $|h|$ is the output length of `CRHash`, and $|v|$ is the block length.*

Proof: First, we prove that at each iteration of the while loop all correct players in H always hold the same value u such that $\texttt{CRHash}(u) = h$. A player is included into H under condition that he broadcasts 1 at Step r.2, which he does only if he holds a value u with $\texttt{CRHash}(u) = h$. Hence for any two correct players $P_i, P_j \in H$ it must hold that $\texttt{CRHash}(v_i) = h$ and $\texttt{CRHash}(v_j) = h$. Since `CRHash` is collision-resistant it implies that $v_i = v_j$.[19]
(Validity of Δ_e) We show that whenever P_x and P_y are correct then $\{P_x, P_y\}$ is not added to Δ at Step r.3. A correct $P_x \in H$ holds v_x with $\texttt{CRHash}(v_x) = h$ and sends $v_x = v_y$ to P_y at Step r.1, who successfully verifies that $\texttt{CRHash}(v_y) = h$ and broadcasts 1 at Step r.2, hence $\{P_x, P_y\}$ is not added to Δ at Step r.3.
(Termination) At each iteration of the while loop either the happy set H or the dispute set Δ grows. $|H|$ is limited by n and $|\Delta|$ is limited by n^2, hence the number of iterations is limited.
(Consistency) We prove that in the end of the protocol all correct players belong either to H or to $\overline{H}$. As shown above Δ remains valid in all iterations, hence for correct players P_x and P_y the pair $\{P_x, P_y\} \notin \Delta$. This implies that if some $P_x \in H$ and $P_y \in \overline{H}$ then the while loop makes another iteration.

If all correct recipients belong to H, then they hold the same value u and hence decide the same. If all correct recipients belong to $\overline{H}$, then they decide on $\perp$.
(Validity) Assume that the sender P_1 is correct. Since P_1 is always in H, it means that all other correct players belong to H in the end of the protocol execution (as shown when proving Consistency property). The sender P_1 holds $v_1 = v$. As shown above all correct players in H hold the same value, hence all correct recipients hold v in the end of the protocol execution and decide on it.
(Complexity analysis) At each iteration of the while loop either H or Δ grows. Hence, the total number of iterations of the while loop is upper bounded by $n + d$ where d is $|\Delta_e| - |\Delta_s|$. This implies that the number of rounds the construction employs is $\mathcal{O}(n + d)$. Hence, the total number of bits broadcast with $\mathfrak{BD}$ is $(|h| + n + d)$ and the total number of bits communicated via bilateral channels is $(n + d)|v|$. ■

[19] More formally, when an adversary can provoke two correct players to hold colliding values for `CRHash` with non-negligible probability, then this adversary can be used to construct an efficient collision-finding algorithm for `CRHash`.

4.2.2 Constructing Broadcast for Long Messages

Now we plug in CryptoBlockBC in the protocol CryptoBC which broadcasts a message block by block.

Protocol CryptoBC(P_1, v, q):

1. Parties initialize dispute set Δ with the empty set.
2. Sender P_1: Cut v in q pieces $v^1, \ldots, v^q$ (add padding if required).
3. For $r = 1, \ldots, q$ invoke CryptoBlockBC(P_1, v^r, Δ), denote the output of party P_i by v_i^r.
4. $\forall P_i \in \mathcal{R}$: If one of $v_i^r = \bot$ then output $\bot$, otherwise output $v_i^1 \| \cdots \| v_i^q$.

Since block broadcast is invoked q times, due to Lemma 4.1 the total communication complexity over bilateral channels is

$$\sum_{i=1}^{q}\left[(n + d_i)\ell/q\right] = (qn + \sum_{i=1}^{q} d_i)\ell/q$$

bits. We know that the sum of d_i is upper bounded by the total number of possible disputes n^2. Hence we have that communication complexity via bilateral channels is upper bounded by $(qn + n^2)\ell/q$. By setting $q = n$ we get that the total communication over bilateral channels is at most $2\ell n$ bits. Furthermore, only $\mathcal{O}(n^2 + n\kappa)$ bits need to be broadcast with $\mathfrak{BD}$.

The number of rounds the construction employs is $\sum_{i=1}^{q} r_i$, where each $r_i \in \mathcal{O}(n + d_i)$. Hence, for $q = n$ we have that the total number of rounds is $\mathcal{O}(n^2)$.

The following theorem summarizes the cryptographically secure construction presented in this section:

Theorem 4.1 *In the setting with $t < n$, the construction* CryptoBC *with $q = n$ achieves cryptographically secure broadcast of ℓ-bit messages in $\mathcal{O}(n^2)$ rounds by communicating $\mathcal{O}(\ell n)$ bits over bilateral channels and $\mathcal{O}(n^2 + n\kappa)$ bits with $\mathfrak{BD}$ (where κ is a security parameter).*

In order to obtain a concrete multi-valued broadcast protocol we instantiate CryptoBC with the protocol [DS83]:

Theorem 4.2 *Instantiating the construction* CryptoBC *with $q = n$ and [DS83] in place of $\mathfrak{BD}$ results in a cryptographically secure multi-valued broadcast protocol for $t < n$ with communication complexity $\mathcal{O}(\ell n + n^5\kappa)$ (where κ is a security parameter).*

4.3 Information-Theoretically Secure Construction

This section is organized similar to the cryptographic case. First, we present a protocol ITBlockBC for broadcasting blocks like CryptoBlockBC, but with the difference that it relies on a universal hash function instead of a collision-resistant one. As in the cryptographic case we then plug ITBlockBC in the ITBC protocol which broadcasts a message block by block q times. Then, we count the communication complexity of the resulting protocol ITBC and select the number of blocks q which makes it optimal.

4.3.1 Block Broadcast Protocol ITBlockBC

Similar to the cryptographic case all parties during the protocol ITBlockBC run are divided into two sets: H and $\overline{H}$. The set H consists of happy players who have already learned v, and $\overline{H}$ consists of the remaining players. The difference to the cryptographic case is that the set H is not monotonically growing—it may happen that the same player may be added and removed from H several times. At each iteration of ITBlockBC we try to move a player from $\overline{H}$ to H. We select a pair of players P_x, P_y such that $P_x \in H$, $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$. Then P_x sends the value it holds to P_y. Now player P_y needs to verify that the value received from P_x is the value that correct parties in H hold. In order to do so, P_y broadcasts with $\mathfrak{BD}$ a key k for an ε-universal hash function `ITHash`, and then P_1 broadcasts with $\mathfrak{BD}$ a hash h for this key. As long as P_y honestly chooses k uniformly at random, with overwhelming probability correct players will obtain different hashes if they hold different values. If a party in $(H \cup \{P_y\}) \setminus \{P_1\}$ holds a value with a hash h, then he broadcasts 1 with $\mathfrak{BD}$, and 0 otherwise (the sender P_1 does not broadcast because if he is correct he can broadcast only 1). If every party broadcasts 1 with $\mathfrak{BD}$, then the iteration was successful and P_y is added to H. Otherwise, some of the parties in $H \cup \{P_y\}$ do not hold the right value and we search for new disputes.

An important difference from the cryptographic case is that disputes may occur not only between P_x and P_y, but between any two parties in H. In order to find such disputes, one must be able to reason about the history of how H was formed. We will keep a history set T which will contain pairs of players (P_x, P_y) such that P_y learned the value it holds from P_x.

```
Protocol ITBlockBC($P_1, v, \Delta_s$):
  1. Parties initialize happy set $H$ to be $\{P_1\}$ and history set $T$ to be $\varnothing$.
     Sender $P_1$: set $v_1 := v$.
  2. While $\exists\, P_x, P_y \in \mathcal{P}$ s.t. $P_x \in H$ and $P_y \in \overline{H}$ and $\{P_x, P_y\} \notin \Delta$ do
     $r$.1 $P_x$: Send $v_x$ to player $P_y$. Denote received value by $v_y$. Add $(P_x, P_y)$ to $T$.
     $r$.2 $P_y$: Generate random $k \in \mathcal{K}$ and broadcast it with $\mathfrak{BO}$.
           Sender $P_1$: Broadcast $h := \texttt{ITHash}_k(v_1)$ with $\mathfrak{BO}$.
     $r$.3 $\forall P_i \in (H \cup \{P_y\}) \setminus \{P_1\}$: If $h = \texttt{ITHash}_k(v_i)$ then broadcast 1, otherwise 0 with $\mathfrak{BO}$.
     $r$.4 If all parties broadcasted 1
             - Add $P_y$ to $H$.
           else
             - For all $(P_i, P_j) \in T$ s.t. $P_i$ broadcasted 1 (resp. $P_i = P_1$) and $P_j$ broadcasted 0, add $\{P_i, P_j\}$ to $\Delta$.
             - Set $H$ to $\{P_1\}$, $T$ to $\varnothing$.
  3. $\forall P_i \in \mathcal{P}$: If $P_i \in H$ decide on $v_i$, otherwise decide on $\perp$.
```

Lemma 4.2 *Given that the initial dispute set Δ_s is valid and `ITHash` is a universal hash function, protocol* ITBlockBC *achieves broadcast (of v) and terminates with a valid dispute set Δ_e (except with negligible probability). Furthermore, the protocol terminates in $\mathcal{O}(n + nd)$ rounds communicating at most $(n+nd)|v|$ bits over bilateral channels and $(n+nd)(|h|+|k|+n)$ bits with $\mathfrak{BO}$, where $d = |\Delta_e| - |\Delta_s|$, $|h|$ is the output length of `ITHash`, $|k|$ is the key length of `ITHash`, and $|v|$ is the block length.*

Proof: First, we prove that at each iteration of the while loop all correct players in H always hold the same value u. More precisely, we need to show that if a correct player P_y is added to H, then, given that all correct players in H hold the same value u, it holds that $v_y = u$. We have that all parties in $(H \cup \{P_y\}) \setminus \{P_1\}$ broadcast 1 at Step r.3. This implies that P_y successfully verifies that $\texttt{ITHash}_k(v_y) = h$, and all correct parties in H verify that $\texttt{ITHash}_k(u) = h$. Due to the fact that P_y is correct, the key k is chosen uniformly at random, so given that $\texttt{ITHash}_k(v_y) = \texttt{ITHash}_k(u)$, it must hold with overwhelming probability $1 - \varepsilon$ that $v_y = u$.
Second, we show that if the condition at Step r.4 is false then at least one new conflict is found. We have that not all players in $(H \cup \{P_y\}) \setminus \{P_1\}$ broadcasted 1. Consider two possible cases:

(Exists $P_z \in H \setminus \{P_1\}$ which broadcasts 0 at step r.3) For P_z to be included

in H there must exist a sequence of players $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ in H such that $P_{i_1} = P_1, P_{i_k} = P_z$ and $(P_{i_j}, P_{i_{j+1}}) \in T$ for all $j = 1, \ldots, k-1$ (see illustration in Figure 4.1). In the r^{th} iteration some of the players in H stayed happy (P_1 and those who broadcasted 1) and some become unhappy (broadcasted 0). We know that P_1 stayed happy and P_z became unhappy. Hence in a row $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ there are players of both types. Then we have that there exist two players $P_{i_a}, P_{i_{a+1}}$ such that P_{i_a} stays happy and $P_{i_{a+1}}$ becomes unhappy. By construction of T, $(P_{i_a}, P_{i_{a+1}}) \in T$ implies that $\{P_{i_a}, P_{i_{a+1}}\}$ is not yet in Δ. Consequently, the pair $\{P_{i_a}, P_{i_{a+1}}\}$ will be identified as having a conflict and will be added to Δ.

(Each $P_i \in H \setminus \{P_1\}$ broadcasts 1 at step r.3) It means that P_x broadcasts 1 (or $P_x = P_s$) and P_y broadcasts 0. Hence the new dispute $\{P_x, P_y\}$ will be added to Δ.

Now we proceed with the proof of the current lemma.
(Validity of Δ_e) We show that whenever P_i and P_j are correct then $\{P_i, P_j\}$ is never added to Δ. The pair $\{P_i, P_j\}$ is added to Δ only when P_i sent some u to P_j (i.e., $(P_i, P_j) \in T$), and they disagree for some key k whether $\texttt{ITHash}_k(u)$ equals h. Hence, at least one of P_i and P_j is corrupted.
(Termination) There can be at most n successive iterations where the set H grows (condition at Step r.4 is true). As shown above whenever condition at Step r.4 is false a new conflict is found. The number of conflicts is limited and so must be the number of the while loop iterations.
(Consistency) We prove that in the end of the protocol all correct players belong either to H or to $\overline{H}$. As shown above Δ remains valid in all iterations, hence for any two correct players P_x, P_y, the pair $\{P_x, P_y\} \notin \Delta$. Hence, if $P_x \in H$ and $P_y \in \overline{H}$ then the while loop must make another iteration.

If all correct recipients belong to H, then they hold the same value u and hence decide the same. If all correct recipients belong to $\overline{H}$, then they decide on $\perp$.
(Validity) Assume that the sender P_1 is correct. Since P_1 is always in H, it means that all other correct players belong to H in the end of the protocol execution (as shown when proving Consistency property). The sender P_1 holds $v_1 = v$. As shown above all correct players in H hold the same value, hence all correct recipients hold v in the end of the protocol execution and decide on it.
(Complexity analysis) There can be at most n consecutive iterations, where no conflict is found, hence the total number of iterations is at most

$n + nd$, where $d = |\Delta_e| - |\Delta_s|$. This implies that the number of rounds the construction employs is $\mathcal{O}(n + nd)$. Furthermore, since the communication costs of each iteration over bilateral channels are $|v|$ bits and over $\mathfrak{BO}$ are $|h| + |k| + n$, we have that the total communication costs of the protocol over bilateral channels are upper bounded by $(n + nd)|v|$ and over $\mathfrak{BO}$ are upper bounded by $(n + nd)(|h| + |k| + n)$. ■

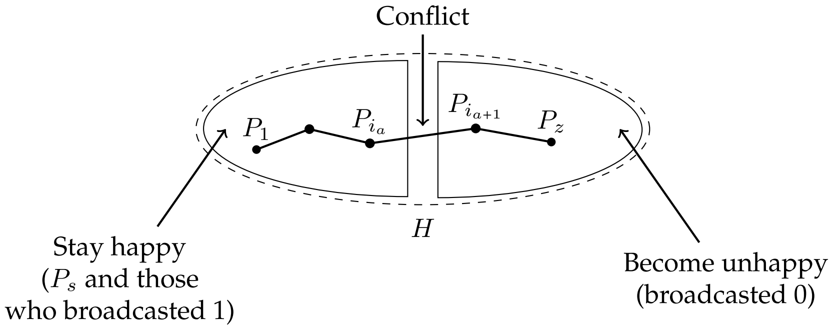
Conflict
P_1
P_{i_a}
$P_{i_{a+1}}$
P_z
H
Stay happy (P_s and those who broadcasted 1)
Become unhappy (broadcasted 0)

Figure 4.1: Conflict finding in ITBlockBC

4.3.2 Constructing Broadcast for Long Messages

Similarly to the cryptographic case, we plug ITBlockBC in the protocol ITBC which simply broadcasts a message block by block. The protocol ITBC is a copy of the protocol CryptoBC with the only difference that CryptoBlockBC is substituted with ITBlockBC.

Protocol ITBC(P_1, v, q):
1. Parties initialize dispute set Δ to be an empty set.
2. Sender P_1: Cut v_1 in q pieces $v^1, \ldots, v^q$ (add padding if required).
3. For $r = 1, \ldots, q$ invoke ITBlockBC(P_1, v^r, Δ), denote the output of party P_i by v_i^r.
4. $\forall P_i \in \mathcal{R}$: If one of $v_i^r = \bot$ then output $\bot$, otherwise output $v_i^1 || \cdots || v_i^q$.

Due to Lemma 4.2 the total communication complexity over bilateral channels of ITBC is at most

$$\sum_{i=1}^{q}\left\lceil (n+d_i n)\ell/q \right\rceil = n(q+\sum_{i=1}^{q} d_i)\ell/q.$$

This expression is bound by $n(q+n^2)\ell/q$. By setting $q=n^2$ we have that communication costs over bilateral channels are at most $2\ell n$. Furthermore, the protocol needs to broadcast $\mathcal{O}(n^4+n^3\kappa)$ bits with $\mathfrak{BO}$.

The number of rounds the construction employs is $\sum_{i=1}^{q} r_i$, where each $r_i \in \mathcal{O}(n+nd_i)$. Hence, for $q=n^2$ we have that the total number of rounds is $\mathcal{O}(n^3)$.

The following theorem summarizes the information-theoretically secure construction presented in this section:

Theorem 4.3 *In the setting with $t<n$, the construction* ITBC *with $q=n^2$ achieves information-theoretically secure broadcast of ℓ-bit messages in $\mathcal{O}(n^3)$ rounds by communicating $\mathcal{O}(\ell n)$ bits over bilateral channels and $\mathcal{O}(n^4+n^3\kappa)$ bits with $\mathfrak{BO}$ (where κ is a security parameter).*

In order to obtain a concrete multi-valued broadcast protocol we instantiate ITBC with the protocol [PW96]:

Theorem 4.4 *Instantiating the construction* ITBC *with $q=n^2$ and [PW96] in place of $\mathfrak{BO}$ results in an information-theoretically secure multi-valued broadcast protocol for $t<n$ with communication complexity $\mathcal{O}(\ell n+n^{10}\kappa)$ (where κ is a security parameter).*

4.4 Instantiating $\mathfrak{BO}$ in Domain-Amplification Protocols with Concrete Bit Broadcast Protocols

In order to obtain a concrete protocol for multi-valued broadcast one takes any domain-amplification protocol (cf. Table 3.1) and composes them with the existing protocols for a bit broadcast (e.g., [BGP92, DS83, PW96]). The security of the composed protocol is then the "minimal" security provided by the domain-amplification protocol and the bit broadcast protocol employed. For example, when composing information-theoretical construction for $t<n/2$ [FH06] with cryptographically secure protocol for $t<n$ [DS83] we obtain multi-valued broadcast protocol with cryptographic security tolerating $t<n/2$ and communication complexity $\mathcal{O}(\ell n+n^4(n+\kappa))$. Further protocols for multi-valued broadcast are given in the following table:

Thr.	Security	Bits Communicated	Literature
$t < n/3$	perfect	$\mathcal{O}(\ell n^2)$	Trivial with [BGP92]
		$\mathcal{O}(\ell n + \sqrt{\ell} n^4 + n^6)$	[LV11a] with [BGP92]
		$\mathcal{O}(\ell n + n^4)$	[Pat11] with [BGP92]
$t < n/2$	inf.-theor.	$\mathcal{O}(\ell n + n^7 \kappa)$	[FH06] with [PW96]
	crypto	$\mathcal{O}(\ell n + n^4(n + \kappa))$	[FH06] with [DS83]
$t < n$	inf.-theor.	$\mathcal{O}(\ell n^2 + n^6 \kappa)$	[PW96]
		$\mathcal{O}(\ell n + n^{10} \kappa)$	This with [PW96]
	crypto	$\mathcal{O}(\ell n^2 + n^3 \kappa)$	[DS83]
		$\mathcal{O}(\ell n + n^5 \kappa)$	This with [DS83]

We note that all multi-valued constructions are only *asymptotically* optimal in ℓ, i.e., they only outperform the trivial construction when relatively long messages are broadcast. Such long messages appear, for example, in voting protocols [CGS97] (where the set of authorities agree on the set of ballots), or in multi-party computation protocols [GMW87] (when all gates on a particular level of the circuit are evaluated in parallel).

Chapter 5

Availability Amplification

In this chapter we first formalize the availability-amplification setting. Then, we give a broadcast availability-amplification protocol for $n = 3$ players which tolerates any number of corruptions (Section 5.3). Finally, in Section 5.4 we show how to use the protocol for three players to obtain a protocol for arbitrary number n of players tolerating $t < n/2$ corruptions.

The contents of this chapter are based on [HR13b].

5.1 Availability-Amplification Setting

Assume we have a broadcast channel to be temporarily available in some predetermined fixed number of rounds. Our goal is to be able to broadcast some information at some later point of time when the temporarily available broadcast is not accessible anymore. A solution to this problem is to use the temporarily available broadcast in order to prepare some resource that can be used later in order to simulate broadcast invocations. A natural choice of such a resource is a trusted setup (cf. Section 1.1.1). We consider two-stage protocols that allow to achieve availability amplification of broadcast. We call such a two-stage protocol a *broadcast scheme*. Formally, a broadcast scheme is a pair of protocols (Setup, BCfromSetup), where Setup generates the parties' secret states with which they start the execution of BCfromSetup. The Setup protocol makes uses of *temporary* broadcast channels (available via $\mathfrak{BO}$) and point-to-point communication channels, while BCfromSetup employs point-to-point channels only. It is required that the combination of Setup and BCfromSetup achieves

broadcast and Setup is independent of the sender's input v provided in BCfromSetup.
We study the efficiency of broadcast schemes, in particular the usage of $\mathfrak{BO}$ employed in the setup protocol. We employ two measures of efficiency in terms of $\mathfrak{BO}$ usage: round complexity and bit complexity. Round complexity denotes the number of rounds in which temporary broadcast is used and bit complexity denotes the number of bits broadcast by it.

5.2 Contributions

If computational security suffices, then in the setup phase it is enough to consistently generate a *Public-Key Infrastructure* (PKI) and then employ [DS83] to broadcast. Generating a PKI requires only 1 round of temporary broadcast, in which each party broadcasts his public key. In case of information-theoretic security known solutions require $\Omega(n^2)$ rounds of temporary broadcast while broadcasting $\Omega(n^8\kappa)$ bits, where κ is a security parameter [PW96]. Motivated by the gap between computational and information-theoretically secure protocols, Garay et al. [GGO12] studied information-theoretically secure broadcast schemes for $t < n/2$.[20] They focused on optimizing the temporary broadcast round complexity in the setup phase and gave a broadcast scheme which requires only 3 rounds of temporary broadcast. The number of bits broadcast by their protocol in the setup phase is $\Omega(n^6\kappa)$. Another construction by Hirt et al. [BHR07] yields a broadcast scheme for $t < n/2$ which needs $\Omega(n)$ rounds of temporary broadcast with $\Omega(n\kappa)$ bits broadcast in the setup phase.[21]
In this chapter we give the first information-theoretically secure broadcast scheme that requires only 1 round of temporary broadcast in the setup phase for $t < n/2$, which is trivially optimal. This result not only improves the broadcast round complexity of all existing broadcast schemes, but allows to construct MPC protocols that use only one round of broadcast (existence of such schemes was unresolved before as stated in [KK07, GGO12]).[22] Furthermore, our protocol needs to broadcast only

[20]Their original motivation was to optimize the number of broadcast rounds needed for information-theoretically secure MPC. The broadcast scheme they give is used as a core component of an MPC protocol.

[21]One can view this construction as a broadcast scheme by interpreting the refresh protocol presented in this chapter as the setup protocol.

[22]Additionally, this shows that the lower bound of at least 2 broadcast rounds for MPC protocols given in [GGO12] is wrong.

$\mathcal{O}(n^3)$ bits in the setup phase *regardless* of how long the security parameter κ is. To our knowledge, this is the first protocol with such a property. Additionally, we give an efficient refresh protocol which allows to broadcast many values given a fixed setup. The table below summarizes the existing broadcast schemes and compares them to the scheme presented in this chapter.

Security	**Thresh.**	$\mathfrak{BO}$ **rounds**	$\mathfrak{BO}$ **bits**	**Ref.**
comp.	$t < n$	1	$\Omega(n\kappa)$	[DS83]
inf.-theor.	$t < n$	$\Omega(n^2)$	$\Omega(n^8\kappa)$	[PW96]
	$t < n/2$	$\Omega(n)$	$\Omega(n\kappa)$	[BHR07]
		3	$\Omega(n^6\kappa)$	[GGO12]
		1	$\mathcal{O}(n^3)$	Sec. 5.4

5.3 The Broadcast Scheme for $n = 3$ and $t \leq 3$

In this section we consider a setting consisting of only *three* players and show how one can prepare a setup that allows a designated player to broadcast one bit. The broadcast scheme is based on a series of reductions among modifications of well known cryptographic primitives. On the highest level we show that: (1) temporary broadcast allows to construct information checking, (2) information checking allows to construct verifiable secret sharing, and (3) verifiable secret sharing allows to construct a setup protocol.
Additionally, we present an optimization which allows players to efficiently generate many setups in parallel. This reduces the number of bits broadcast in the setup phase while still requiring only one round of a temporary broadcast.

5.3.1 Detectable Information Checking

Information checking (IC) is an information-theoretically secure method for authenticating data among three players D, I, R. The dealer D holds a secret value s from some finite field $\mathbb{F}$ which he sends to an intermediary I together with an authentication information. The intended recipient

R gets a verification information. Later I sends s' and some authentication information to R, who uses the verification information to check whether $s = s'$. We propose a non-robust modification of IC which we call *Detectable Information Checking* (DIC), where the authentication phase may either *succeed* or *abort*. If it aborts then the parties output a *dispute* Δ, which is a pair of players (i.e., $\Delta \subseteq \{D, I, R\}$), at least one of them being Byzantine.

Formally, a DIC scheme is a pair[23] of protocols $(\mathsf{ICSetup}, \mathsf{ICReveal})$, where in ICSetup D inputs s and then parties either abort with a dispute or succeed by saving a local state. If ICSetup succeeds then parties invoke ICReveal such that R outputs s' or $\perp$. A DIC scheme must satisfy the following security properties:

COMPLETENESS: If D, I and R are correct, then ICSetup succeeds and R will output $s' = s$ in ICReveal.

NON-FORGERY: If D and R are correct and ICSetup succeeds, then R will output $s' = s$ or $s' = \perp$ in ICReveal.

COMMITMENT: If I and R are correct and ICSetup succeeds, then at the end of ICSetup I knows a value s' such that R will output s' in ICReveal.

PRIVACY: If D and I are correct, then R obtains no information on s during ICSetup.

DETECTION: In case ICSetup aborts every correct party outputs the same dispute Δ.

TERMINATION: Every correct party terminates ICSetup, respectively ICReveal.

We say that a DIC scheme is information-theoretically secure if the properties above are guaranteed with overwhelming probability.

The Protocol

In this section we present a protocol for DIC based on [CDD+99]. The secret and the verification information will lie on a line (a polynomial of degree 1), where the secret will be the value in 0. Intermediary I gets to know the line, while the recipient R learns a value on this line in some secret point α unknown to I. In the reveal phase, R accepts a line from I only if the evaluation of this line in α is correct. In order to ensure

[23]Sometimes (as in [CDD+99, GGO12]) IC is presented as a triple of protocols Distr, AuthVal, RevealVal where Distr and AuthVal is a more fine-grained representation of ICSetup.

commitment property I verifies whether D distributed consistent information to him and to R. This is done during one broadcast round in ICSetup. As an outcome of the broadcast parties may succeed in ICSetup or abort with a dispute.
Let $s, y, z, \alpha \in \mathbb{F}$. We say that the triple (s, y, z) is 1_α-consistent provided that three points $(0, s)$, $(1, y)$ and (α, z) lie on a line in $\mathbb{F}$ (that is, for some $L(x) = bx + c$ over $\mathbb{F}$, we have $L(0) = s$, $L(1) = y$ and $L(\alpha) = z$).

Protocol ICSetup(s):

1. Dealer D chooses a random value $\alpha \in \mathbb{F} \setminus \{0, 1\}$ and additional random $y, z \in \mathbb{F}$ such that (s, y, z) is 1_α-consistent. In addition it chooses a random 1_α-consistent vector (s', y', z'). D sends s, s', y, y' to I and α, z, z' to R.
2. Intermediary I chooses a random $d \in \mathbb{F}$. I sends d to D and $d, s' + ds, y' + dy$ to R. Let d_1 denote the value received by D and d_2, s'', y'' the values received by R.
3. Every player **broadcasts** with $\mathfrak{BO}$ the following (in parallel):
   - 3.1 Dealer broadcasts triple $T_D = (d_1, s' + d_1 s, y' + d_1 y)$.
   - 3.2 Intermediary broadcasts triple $T_I = (d, s' + ds, y' + dy)$.
   - 3.3 Recipient broadcasts triple $T_R = (d_2, s'', y'')$ and a bit b, where b is 1 if $(s'', y'', z' + d_2 z)$ is 1_α-consistent and 0 otherwise.
4. Every player checks for abortion:
   If $T_D \neq T_I$ then abort with $\Delta = \{D, I\}$.
   If $T_R \neq T_I$ then abort with $\Delta = \{R, I\}$.
   If $T_D = T_R$ and $b = 0$ then abort with $\Delta = \{D, R\}$.
   Otherwise, the protocol succeeds and D stores nothing, I stores (s, y) and R stores (α, z).

Protocol ICReveal(s):

1. Intermediary I sends s, y to R. If (s, y, z) is 1_α-consistent then R decides on s, otherwise on $\perp$.

Lemma 5.1 *The pair of protocols* (ICSetup, ICReveal) *achieves DIC (except with probability $\mathcal{O}(1/|\mathbb{F}|)$). Furthermore,* ICSetup *uses the underlying broadcast channel during one predetermined round (where each player broadcasts $\mathcal{O}(\log|\mathbb{F}|)$ bits) and* ICReveal *does not use broadcast at all.*

Proof: First, we show that each DIC property is satisfied:

COMPLETENESS: It is clear that if all parties are honest, then ICSetup succeeds and R outputs s in ICReveal.

NON-FORGERY: If D and R are correct and ICSetup succeeds, then prior to ICReveal I has no information on α except that it is different from 0 and 1. For I making R output in ICReveal any value different from s is equivalent to guessing α, which he can do with probability at most $1/(|\mathbb{F}|-2)$.

COMMITMENT: If I and R are correct and ICSetup succeeds, then $(s'+ds, y'+dy, z'+dz)$ is 1_α-consistent with a randomly chosen d. R accepts s if (s,y,z) is 1_α-consistent. The probability that (s,y,z) is not 1_α-consistent, given that $(s'+ds, y'+dy, z'+dz)$ is 1_α-consistent, is at most $1/|\mathbb{F}|$.

PRIVACY: Here D and I are correct. In ICSetup R observes only $\alpha, z, z', d, s'+ds, y'+dy$. R knows that $(s'+ds, y'+dy, z'+dz)$ is 1_α-consistent, hence his knowledge is equivalent to $\alpha, z, z', d, s'+ds$ which is independent of s.

DETECTION: Since parties compute Δ based only on the information that was broadcast they decide on the same Δ. We are left to show that Δ forms a dispute. If D and I are correct then it must be that $T_D = T_I$, analogously if I and R are correct then $T_I = T_R$. If $T_D = T_R$ and D is correct then R must hold a 1_α-consistent triple $(s'', y'', z'+d_2 z)$.

TERMINATION: Due to their specifications, protocols ICSetup and ICReveal always terminate.

Finally, the protocol ICSetup uses broadcast during only one round at Step 3 where each party broadcasts $\mathcal{O}(\log|\mathbb{F}|)$ bits, while the protocol ICReveal does not broadcast. ■

5.3.2 Detectable Verifiable Secret Sharing

In this section we consider a very restricted setting for VSS (cf. Section 2.2.3) consisting of only three players D, P_1, P_2. The dealer D holds a secret value s from some finite field $\mathbb{F}$ and shares it among two recipients P_1, P_2, such that individually they have no information about s. Later in the reconstruction phase all correct recipients reconstruct the same s' which equals s if the dealer is correct. We consider a non-robust version of VSS which we call *Detectable Verifiable Secret Sharing* (or short DVSS), where the sharing phase can *abort* in the presence of malicious behavior. Formally, a DVSS scheme is a pair of protocols (DVSS-Share, DVSS-Rec), where in DVSS-Share D inputs s and then parties either abort with a dispute Δ or succeed by saving a local state. If DVSS-Share succeeds then the parties invoke DVSS-Rec such that the recipients reconstruct the shared secret. A DVSS scheme must satisfy the following security properties:

CORRECTNESS: If DVSS-Share succeeds, then there exists a fixed value $s' \in \mathbb{F}$ which will be reconstructed as a result of DVSS-Rec by every correct recipient. If D is correct then $s' = s$.

PRIVACY: If D is correct, then corrupted parties obtain no information on s in DVSS-Share.

DETECTION: If D, P_1 and P_2 are correct then DVSS-Share always succeeds. In case DVSS-Share aborts every correct party outputs the same dispute Δ.

TERMINATION: Every correct party terminates DVSS-Share, respectively DVSS-Rec.

We say that a DVSS scheme is information-theoretically secure if the properties above are guaranteed with overwhelming probability. Furthermore, by t-DVSS we denote a DVSS scheme tolerating $\leq t$ corruptions, and by t-DVSS$^+$ we denote a scheme which is t-DVSS but Detection and Termination hold for arbitrary number of corruptions.

The Protocol

In this section we present an implementation of 1-DVSS$^+$ based on DIC. In order to share a secret s the dealer generates two random values s_1, s_2 such that $s_1 + s_2 = s$. Then the dealer authenticates s_1 by invoking ICSetup(s_1) where P_1 acts as intermediary and P_2 as recipient. In parallel, the dealer authenticates s_2 by invoking ICSetup(s_2) where P_2 acts as intermediary and P_1 as recipient. If one of the ICSetup invocations aborts then DVSS-Share aborts as well. The reconstruction consists of running ICReveal among P_1, P_2 and the dealer sending the secret s to both recipients. If a correct recipient obtains non-$\bot$ value in ICReveal, then it reconstructs $s' = s_1 + s_2$, otherwise it outputs $s' = s$ received from the dealer.

Protocol DVSS-Share(s):

1. The dealer D chooses random $s_1, s_2 \in \mathbb{F}$ such that $s = s_1 + s_2$.
2. Then D, P_1, P_2 execute in parallel ICSetup(s_1) where P_1 is intermediary and P_2 is recipient and ICSetup(s_2) where P_2 is intermediary and P_1 is recipient.

3. Every player checks for abortion:
   If any of ICSetup aborts, then DVSS-Share aborts as well with a dispute Δ output by one of ICSetup (in case both aborts, then output the dispute from the first).
   Otherwise, the protocol succeeds and each player saves the states obtained from both ICSetup invocations.

Protocol DVSS-Rec(s):

1. The dealer D sends s to both recipients.
2. Players P_1, P_2 invoke ICReveal(s_1) and ICReveal(s_2). If P_i obtains a share $s_j \neq \perp$ from the other recipient P_j then it decides on $s_i + s_j$, otherwise it decides on the value s received from the dealer.

Lemma 5.2 *The pair of protocols* (DVSS-Share, DVSS-Rec) *achieves* 1*-DVSS*$^+$ *(except with probability* $\mathcal{O}(1/|\mathbb{F}|)$*). Furthermore,* DVSS-Share *uses the underlying broadcast channel in only one predetermined round (where each player broadcasts* $\mathcal{O}(\log|\mathbb{F}|)$ *bits) and* DVSS-Rec *does not use broadcast at all.*

Proof: First, we show that each 1-DVSS$^+$ property is satisfied:

CORRECTNESS: We prove that Correctness holds when the number of corrupted parties $t \leq 1$. If all parties are correct ($t = 0$) then, due to the Completeness property of DIC, correct recipients will output $s_1 + s_2 = s$ in DVSS-Rec. Assume now only one party is corrupted ($t = 1$). Consider two cases: (1) *the dealer is corrupted* and (2) *one of the recipients is corrupted*. In case (1), due to the Commitment property of DIC, player P_1 knows s_1 such that P_2 outputs s_1 in ICReveal, and P_2 knows s_2 such that P_1 outputs s_1 in ICReveal (except with probability $\mathcal{O}(1/|\mathbb{F}|)$). Hence, since both P_1 and P_2 are correct they will both output $s' = s_1 + s_2$ in DVSS-Rec. In case (2), wlog assume that a correct recipient is P_1. Due to the Non-Forgery property of DIC, player P_1 may obtain in ICReveal only s_2 or $\perp$ (except with probability $\mathcal{O}(1/|\mathbb{F}|)$). In both cases P_1 outputs $s' = s_1 + s_2$ or $s' = s$, which is the same for a correct D.

PRIVACY: Wlog assume that a corrupted recipient is P_2 and P_1 is correct. Due to the Privacy property of DIC, player P_2 who acts as a recipient in ICSetup(s_1) has no information about s_1 in the end of ICSetup. Hence, after DVSS-Share P_2's view contains only s_2 which is independent of s.

DETECTION: If all parties are correct, then, due to the Completeness property of DIC, both $\mathsf{ICReveal}(s_1)$ and $\mathsf{ICReveal}(s_2)$ succeed and hence DVSS-Share succeeds. Parties abort if and only if one of ICSetup aborts. Due to the Detection property of ICSetup parties output the same dispute Δ.

TERMINATION: Due to their specifications, protocols DVSS-Share and DVSS-Rec always terminate.

Finally, the protocol DVSS-Share uses the underlying broadcast channel only at Step 2 where ICSetup uses it, while the protocol DVSS-Rec does not broadcast. Since two instances of ICSetup are invoked in parallel, according to Lemma 5.1 there is only one predetermined round where the underlying broadcast channel is invoked and each player broadcasts $\mathcal{O}(\log|\mathbb{F}|)$ bits. ■

5.3.3 Broadcast Scheme

We consider three players $\{D, P_1, P_2\}$, where D is the sender (also called the dealer) and P_1, P_2 are the recipients. In the setup phase parties execute protocol Setup_3 which makes use of a temporary broadcast channel. The setup generation may abort by outputting a dispute Δ. Later the dealer D can broadcast a bit value with the protocol $\mathsf{BCfromSetup}_3$ using the setup created. This is done differently depending on whether the preceding Setup_3 aborted or succeeded.

The Protocol

The protocol for generating a setup uses 1-DVSS$^+$ together with *Message Authentication Codes* (MACs). We employ a MAC scheme which uses a preshared key. In the setup phase, the dealer shares a random key from some finite field $\mathbb{F}$ using DVSS-Share. In order to broadcast a message later, the dealer sends the message together with its authentication information to recipients. Then the recipients exchange the messages they received from the dealer. It is guaranteed that if both recipients are correct then they hold the same set of at most two messages. Then the parties invoke DVSS-Rec and learn the key for the MAC scheme used. Recipients decide on the message with a valid MAC, respectively on $\perp$ if no/both messages match.

In order to construct an information-theoretically secure MAC scheme,

we employ a trivially unforgeable authentication scheme for bits. To authenticate a message consisting of one bit b the dealer sends to the recipients P_1, P_2 either the left part of the preshared key ($b = 0$) or its right part ($b = 1$).
For the simplicity of the presentation, let $\overline{P_1}$ denote P_2 and $\overline{P_2}$ denote P_1. Additionally, let $\mathbb{F}$ be a finite field of $2^{2\kappa}$ elements interpreted as bit strings of length 2κ.

Protocol Setup_3:

1. The dealer generates random $k_0||k_1 \in \{0,1\}^{2\kappa}$.
   Parties execute $\mathsf{DVSS\text{-}Share}(k_0||k_1)$. If $\mathsf{DVSS\text{-}Share}$ aborts, then abort as well and output the dispute Δ.

Protocol $\mathsf{BCfromSetup}_3(b)$:

```
1. If $\mathsf{Setup}_3$ succeeded
    1.1 The dealer sends $k_b$ to both $P_1, P_2$. Denote the values received by the players $P_1$ and $P_2$ with $a_1$ and $a_2$, respectively.
    1.2 The recipients exchange $a_1$ and $a_2$ and form a set of authenticators $A = \{a_1, a_2\}$.
    1.3 The players execute $\mathsf{DVSS\text{-}Rec}$ and get $k_0||k_1$ as an output.
    1.4 The recipients decide on 0 if $k_0 \in A$ and on 1 otherwise.
2. else $\mathsf{Setup}_3$ aborted with $\Delta$ then
    2.1 If $\Delta = \{D, P_i\}$ then the dealer sends $b$ to $\overline{P_i}$ who forwards it to $P_i$. All parties decide on the values received.
    2.2 If $\Delta = \{P_1, P_2\}$ then the dealer sends $b$ to $P_1$ and $P_2$. All parties decide on the values received.
```

Lemma 5.3 *The protocol* $\mathsf{BCfromSetup}_3$ *achieves broadcast (of b) given that* Setup_3 *has been executed before (except with probability $\mathcal{O}(2^{-\kappa})$). Furthermore,* Setup_3 *is independent of b and uses the temporary available broadcast channels (via $\mathfrak{BD}$) in only one predetermined round (where each party broadcasts $\mathcal{O}(\kappa)$ bits).*

Proof: First, we prove that broadcast properties are satisfied. If Setup_3 outputted Δ, then by inspection of the cases it is clear that $\mathsf{BCfromSetup}_3$ achieves broadcast. Assume now that Setup_3 succeeded. We consider four possible cases of player corruption:

(All players are correct) Due to the Correctness and Detection properties of DVSS^+, $\mathsf{DVSS\text{-}Share}$ succeeds and all players reconstruct dealer's

$k_0||k_1$ in DVSS-Rec. Hence, all players decide on dealer's b in BCfromSetup$_3$.

(Two players are correct) Consider two possibilities: (1) *both recipients are correct*, and (2) *the dealer and one of the recipients are correct*. In both cases Termination clearly holds.
In case (1) Validity clearly holds. Due to the Correctness property of DVSS$^+$ for $t = 1$, both recipients reconstruct the same $k_0||k_1$ (except with probability $\mathcal{O}(1/|\mathbb{F}|)$). Since they hold the same set A and use the same key $k_0||k_1$, the recipients decide on the same value.
Consider case (2). Wlog assume that P_1 is correct and P_2 is Byzantine. Due to the Correctness property of DVSS$^+$ for $t = 1$, player P_1 obtains the key $k_0||k_1$ shared by the dealer (except with probability $\mathcal{O}(1/|\mathbb{F}|)$). Due to the Privacy property of DVSS$^+$ for $t = 1$, player P_2 does not learn the authentication information for the bit which was not sent by the dealer. The probability that P_2 forges a MAC for an unknown bit is the same as guessing the corresponding part of the key which is at most $2^{-\kappa}$. Hence, a correct recipient decides on the value send by the correct sender (except with probability $\mathcal{O}(2^{-\kappa})$).

(One player is correct) If a single recipient or the dealer is correct, then broadcast properties are clearly satisfied.

(No player is correct) No properties are needed to be proven here.

The protocol Setup$_3$ uses temporary broadcast only during the DVSS-Share invocation at Step 1. Due to Lemma 5.2 DVSS-Share uses broadcast in only one predetermined round (where each player broadcasts $\mathcal{O}(\kappa)$ bits) and so does Setup$_3$. ■

5.3.4 Efficient Parallel Setup

In the previous section we have shown how to generate a setup for one bit broadcast. An obvious approach for generating a setup for an ℓ bit message is to prepare ℓ such setups. In this section we show how to efficiently parallelize ℓ setup invocations such that the temporary broadcast is used only small number of times. As a key ingredient of such a parallelization, we use the protocol AmplifyBC$_3$ (cf. Section 3.5.1) which given an opportunity to broadcast 2 bits (in fact even opportunity to broadcast a single value from domain of size 3) allows to broadcast a message of arbitrary fixed length.

Now we turn to describing how to efficiently parallelize many invocations of Setup$_3$. We run many instances of Setup$_3$ in parallel such that each player consolidates the values it needs to broadcast in one string which is broadcast with $\mathsf{AmplifyBC}_3$. The protocol $\mathsf{AmplifyBC}_3$ uses temporary broadcast as its underlying primitive for broadcasting 2 bits. The following lemma summarizes the properties achieved by this construction:

Lemma 5.4 *The broadcast scheme described above generates ℓ setups. Furthermore, the temporary broadcast is used in one predetermined round only where each player broadcasts at most 2 bits.*

5.4 Broadcast Scheme for any n and $t < n/2$

Broadcast is achievable from point-to-point channels without a trusted setup if and only if $t < n/3$. Fitzi and Maurer [FM00] proposed a construction of a broadcast protocol for $t < n/2$ from the broadcast channels among every triple of players. We use following theorem from [FM00]:

Theorem 5.1 *In the model where broadcast is available among every triple of players there is a protocol that implements broadcast among n players tolerating $t < n/2$ corruptions. This protocol invokes underlying three-party broadcast channels at most n times for each triple (P_i, P_j, P_k), where P_i is the sender and P_j, P_k are the recipients.*

The protocol we present generates n setups for each triple of players where P_i is the sender and P_j, P_k are the recipients. It is done by invoking in parallel procedure Setup_3.

Protocol Setup_n:

1. For each possible sender P_i and recipients P_j, P_k (i, j, k are all different): Parties P_i, P_j, P_k invoke Setup_3 n times in parallel.

Protocol $\mathsf{BCfromSetup}_n(b)$:

1. Use protocol by [FM00], whenever P_i needs to broadcast 1 bit among P_j, P_k use the protocol $\mathsf{BCfromSetup}_3$ with the prepared setup.

Theorem 5.2 *The protocol* $\mathsf{BCfromSetup}_n$ *achieves broadcast (of b) for $t < n/2$ given that* Setup_n *has been executed before (except with probability $\mathcal{O}(n^4 2^{-\kappa})$).*

Furthermore, Setup_n *is independent of b and uses temporary broadcast procedure in only one predetermined round (where players needs to broadcast $\mathcal{O}(n^3)$ bits).*

Proof: Due to their specifications, protocols Setup_n and $\mathsf{BCfromSetup}_n$ always terminate. Since $\mathsf{BCfromSetup}_3$ has failure probability $\mathcal{O}(2^{-\kappa})$ then due to Theorem 5.1 the probability that $\mathsf{BCfromSetup}_n$ does not achieve broadcast is bounded by $\mathcal{O}(n^4 2^{-\kappa})$. The protocol Setup_n uses the temporary broadcast only during the parallel invocation of Setup_3 at Step 1. Due to Lemma 5.3 Setup_3 uses broadcast only in one predetermined round and so does Setup_n. Due to Lemma 5.4, each party in a triple needs to broadcasts 2 bits in order to generate n setups. There are $\binom{n}{3}$ triples in total, so players broadcast $\mathcal{O}(n^3)$ bits in one round of the setup phase. ■

5.4.1 Refreshing the Setup

A broadcast scheme, as we defined it, works in the following way: first we run the Setup_n protocol for the setup generation and then execute $\mathsf{BCfromSetup}_n$ to broadcast a value. The invocation of the $\mathsf{BCfromSetup}_n$ protocol actually "consumes" the setup, i.e., one setup may be used for broadcasting one value only. A concept of a *refresh* protocol allows to extend a fixed setup for many setups [PW96].
A refresh protocol for the presented scheme with arbitrary number of players n runs refresh protocol for each triple of players. A refresh procedure for a triple is sketched below. Let us denote the three players with P_1, P_2, P_3. We assume that the initial setup allows each player P_1, P_2, P_3 to broadcast 2 bits. We construct a setup allowing each party to broadcast any number of ℓ bits. For each possible sender (P_1,P_2 or P_3) the players run in parallel ℓ instances of Setup_3 where the values each player P_i is supposed to broadcast in every instance are consolidated in one string s_i of $\mathcal{O}(\ell\kappa)$ bits. In order to broadcast s_i, each P_i uses the protocol $\mathsf{AmplifyBC}_3$ where 2 initial setups are used.

5.4.2 Application to MPC

One of the most important applications of broadcast schemes is a secure MPC [GMW87]. In the settings with $t < n/2$ MPC is achievable from point-to-point communication only when a broadcast channel is available additionally. Such a broadcast channel is usually simulated with a broadcast scheme which tolerates $t < n/2$. The broadcast scheme presented in this chapter shows that there is an information-theoretically secure MPC protocol which uses a broadcast channel during only one round. This is achieved by running Setup_n sufficiently many times in parallel (this uses one round of temporary broadcast) to prepare enough setups, and simulating all invocations to broadcast in the MPC protocol by $\mathsf{BCfromSetup}_n$. Furthermore, if it is not known beforehand how many times the broadcast channel will be used, the refresh protocol is used.

5.5 A More Efficient Scheme for any n ant $t < n/2$

In this Section we give an optimized version of the $\mathsf{BCfromSetup}_n$ protocol which we denote with $\mathsf{BCfromSetup}^+_n$. In particular we optimize the round complexity of the protocol. The broadcast protocol for $t < n/2$ by Fitzi and Maurer [FM00] takes $\Theta(t)$ rounds where in each round three-party broadcast channels among some triples of players are used. Hence, if each call to a three-party broadcast channel is substituted with $\mathsf{BCfromSetup}_3$ procedure, which takes constant number of rounds, then the total number of rounds the protocol $\mathsf{BCfromSetup}_n$ takes is $\Theta(t)$. In the following we sketch the protocol $\mathsf{BCfromSetup}^+_n$ which uses three-party broadcast channels as well as the $\mathsf{BCfromSetup}_n$ protocol, but simulates broadcast among n players in expected constant number of rounds.

We start by introducing necessary technicalities in Section 5.5.1 and then proceed to the main protocol $\mathsf{BCfromSetup}^+_n$ in Section 5.5.2.

5.5.1 Additional Preliminaries

In this section we show how one can implement gradecast among n parties tolerating $t < n/2$ given three-party broadcast channels. Our construction consists of two steps. First, we construct weak broadcast (cf. Section 2.2.1) given three-party broadcast channels. Then we construct gradecast (cf. Section 2.2.2) given weak broadcast. Our constructions are based on those given in [Fit03] but are extended to support gradecasting of the values from arbitrary domains efficiently.

We first give a weak broadcast protocol for arbitrary domains $\mathcal{X}$ instead of the binary domain only (it is based on the Protocol 5.1 in [Fit03]):

Protocol WeakBroadcast$(P_1, \mathcal{X}, v)$:

1. Sender P_1 uses three-party broadcast to send v to every pair of players in $\mathcal{R}$.
2. $\forall P_i \in \mathcal{R}$: If all values received in the previous step are the same (equal to some $u \in \mathcal{X}$) then output $v_i = u$, otherwise output $v_i = \perp$.

Lemma 5.5 *The protocol* WeakBroadcast *achieves weak broadcast for $t < n/2$.*

Proof: We show that each of the weak broadcast properties is satisfied:

VALIDITY: If P_1 is correct then he always broadcasts v among each pair of recipients in $\mathcal{R}$. Hence, each recipient $P_i \in \mathcal{R}$ receives only v and outputs $v_i = v$.

WEAK CONSISTENCY: Take any correct $P_i \in \mathcal{R}$ outputting v_i. Consider any other $P_j \in \mathcal{R}$. Since P_i outputs v_i then all the values P_i received from P_1 via three-party broadcast are equal to v_i. In particular, this means that P_1 used three-party broadcast to send v_i to P_i and P_j. This implies that one of the values that P_j received is v_i, consequently, P_j can output only on $v_j = v_i$ or $\perp$.

TERMINATION: Follows from the protocol inspection. ■

We now present a gradecast protocol allowing for arbitrary domains $\mathcal{X}$ (it is based on the Protocol 4.9 in [Fit03] which achieves binary gradecast only):

Protocol Gradecast$(P_1, \mathcal{X}, v)$:

1. Sender P_1: Weak broadcast v. Denote output of each player $P_i \in \mathcal{R}$ with w_i, and let $w_1 := v$.
2. $\forall P_i \in \mathcal{P}$: Weak broadcast w_i. Denote output of each player P_j with w_{ij}, and let $w_{ii} := w_i$.
3. $\forall P_i \in \mathcal{R}$: $\forall u \in \mathcal{X}$ let $T_i^u = \{P_j \in \mathcal{P} \mid w_{ji} = u\}$. Let v_i be u with maximal $|T_i^u|$ (break ties arbitrarily); if $|T_i^{v_i}| > n/2$ then $g_i = 1$, otherwise $g_i = 0$. Output (v_i, g_i).

Lemma 5.6 *The protocol* Gradecast *achieves gradecast for $t < n/2$.*

Proof: We show that each of the gradecast properties is satisfied:

VALIDITY: Let H denote the set of correct players. Consider any correct player $P_j \in H$. If the sender P_1 is correct then every correct $P_i \in H$ has $w_i = v$. Every correct $P_i \in H$ weak broadcasts $w_i = v$, hence $H \subseteq T_j^v$. Since $|H| > n/2$, then $|T_j^v| > n/2$ and P_j outputs values $v_j = v$ with grade 1.

GRADED CONSISTENCY: Consider any correct player P_i outputting (v_i, g_i) with $g_i = 1$. We have that $|T_i^{v_i}| > n/2$. Hence, there is at least one correct player P_j who weak-broadcasts v_i at Step 2. This implies that $w_j = v_i$ and any other correct player P_k has $w_k = v_i$ or $\perp$. Therefore, at Step 2 all correct players weak-broadcast v_i or $\perp$. Let H^{v_i} denote the set of correct players weak-broadcasting v_i and $H^\perp$ weak-broadcasting $\perp$. Then $M^{v_i} = T_i^{v_i} \setminus H^{v_i}$ is the set of corrupted parties who weak-broadcast v_i and $M^* = \mathcal{P} \setminus (H^{v_i} \cup H^\perp \cup M^{v_i})$ is the set of the remaining malicious players. We prove that $|M^*| < |H^{v_i}|$. Since $|T_i^{v_i}| > n/2$ we have that $|H^{v_i}| + |M^{v_i}| > n/2$. Since the majority of players are correct $|H^{v_i}| + |H^\perp| > n/2$. Combining two inequalities we have that $|H^{v_i}| + (|H^{v_i}| + |M^{v_i}| + |H^\perp|) > n$. Finally, we substitute the second term in the left part of the last inequality with $(n - |M^*|)$ and prove the claim.

In order to prove that Graded Consistency is satisfied we need to show that for any correct player P_j holds $|T_j^{v_i}| > |T_j^u|$ for any $u \neq v_i$. Since no player in $H^{v_i} \cup M^{v_i} \cup H^\perp$ weak-broadcasts u we have that $T_j^u \subseteq M^*$, hence $|T_j^u| \leq |M^*|$. Since all players in H^{v_i} are correct and weak-broadcast v_i we have that $|H^{v_i}| \leq |T_j^{v_i}|$. So, we have that $|T_j^u| \leq |M^*|$, $|H^{v_i}| \leq |T_j^{v_i}|$ and $|M^*| < |H^{v_i}|$ implies $|T_j^u| < |T_j^{v_i}|$.

TERMINATION: Follows from the protocol inspection. ■

5.5.2 A Sketch of the Protocol $\mathsf{BCfromSetup}_n^+$

In [KK06a, KK06b] the authors construct an expected constant-round broadcast protocol for $t < n/2$ based on PKI. In the following we show how to achieve the same result by substituting PKI with a setup allowing for three-party broadcast among every triple of players.

The protocol of [KK06a, KK06b] consists of the following four steps:

1. Given PKI and secure channels, construct gradecast [KK06b, Appendix A.2].

2. Given gradecast and secure channels, construct moderated VSS [KK06b, Theorem 5]. "Moderated VSS" is a variant of VSS

whose security guarantees depend on the correctness of one of the players (called moderator).

3. Given moderated VSS construct "Oblivious Leader Election" primitive which allows players to fairly elect a leader among them [KK06b, Theorem 8].

4. Given oblivious leader election and gradecast, construct an expected constant-round broadcast protocol [KK06b, Theorem 13].

We construct the protocol $\mathsf{BCfromSetup}_n^+$ along the lines of the protocol of [KK06a, KK06b], where in the first step the gradecast implementation of [KK06b, Appendix A.2] is substituted with the gradecast implementation from Section 5.5.1.

Appendices

Appendix A

On the Constructions of Liang and Vaidya [LV11a, LV10a, LV10b]

Liang and Vaidya have four papers based on the similar techniques which relate to the construction of multi-valued broadcast and consensus protocols:

1. The first paper [LV10a] presents a perfectly secure multi-valued broadcast protocol which tolerates $t < n/3$.

2. The second paper [LV10b] proposes an information-theoretically secure modification of the broadcast protocol from the first paper which tolerates $t < n/3$.

3. The third paper [LV11a] constructs a perfectly secure multi-valued consensus protocol for $t < n/3$.

4. The fourth paper [LV11b] is an archive version of the third paper containing the same protocol.

In the third and the fourth paper the authors explain how their protocols can be modified to tolerate $t \geq n/3$ by substituting the employed procedure for 1-bit broadcast with a protocol that tolerates $t \geq n/3$ (e.g., [DS83, PW96]). This modification can be applied in all four papers since they share similar structure and are based on similar techniques. In the following we clarify why even with this modification the protocols from the cited above papers cannot tolerate $t \geq n/2$.

In the first and the second paper the presented broadcast protocols describe communication between players based on their trust to each other. While the trust concept applies for any $t < n$, the broadcast protocols put a limitation on the trust relation—it is required that any two players either trust each other or there is another player whom both players trust (see Section V.B in [LV10a] and Section 3 in [LV10b]). Such a mutually trusted player exists only for $t < n/2$ and hence the protocols cannot progress when $t \geq n/2$, i.e., termination is not guaranteed.

In the third and the fourth paper a consensus protocol for $t < n/3$ is presented. As consensus is not achievable for $t \geq n/2$, the proposed modification can at most construct consensus for $t < n/2$, which in turn provides broadcast for the same bound only.

Bibliography

[AFM99] Bernd Altmann, Matthias Fitzi, and Ueli M. Maurer. Byzantine agreement secure against general adversaries in the dual failure model. In Prasad Jayanti, editor, *DISC*, volume 1693 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 1999.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2007.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC '96)*, pages 479–488. ACM, 1996.

[BGP92] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Bit optimal distributed consensus. In *Computer Science Research*, pages 313–322. Plenum Publishing Corporation, New York, NY, USA, 1992.

[BH06] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference — TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer-Verlag, March 2006.

[BHR07] Zuzana Beerliova-Trubiniova, Martin Hirt, and Micha Riser. Efficient Byzantine agreement with faulty minority. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 393–409, 2007.

[BOPV06] Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in o(log n) rounds. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 179–186, New York, NY, USA, 2006. ACM.

[CDD$^+$99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 311–326, May 1999.

[CFF$^+$05] Jeffrey Considine, Matthias Fitzi, Matthew Franklin, Leonid A. Levin, Ueli Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, July 2005.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *FOCS*, pages 383–395. IEEE Computer Society, 1985.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[CW92] Brian A. Coan and Jennifer L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97:61–85, March 1992.

[Dol82] Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.

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

[FGH+02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. Detectable byzantine agreement secure against faulty majorities. In *Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing*, PODC '02, pages 118–126, New York, NY, USA, 2002. ACM.

[FGMvR02] Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 482–501. Springer, 2002.

[FH06] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued Byzantine agreement. In *Proceedings of the 26th annual ACM symposium on Principles of distributed computing, PODC '06*, pages 163–168, New York, NY, USA, 2006. ACM.

[FHHW03] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 51–67, Berlin, Heidelberg, 2003. Springer-Verlag.

[FHM98] Matthias Fitzi, Martin Hirt, and Ueli Maurer. Trading correctness for privacy in unconditional multi-party computation. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 121–136. Springer-Verlag, August 1998. Corrected proceedings version.

[Fit03] Matthias Fitzi. *Generalized Communication and Security Models in Byzantine Agreement*. PhD thesis, ETH Zurich, March 2003. Reprint as vol. 4 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-853-3, Hartung-Gorre Verlag, Konstanz, 2003.

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 148–161, New York, NY, USA, 1988. ACM.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, August 1997.

[FM98] Matthias Fitzi and Ueli M. Maurer. Efficient byzantine agreement secure against general adversaries. In Shay Kutten, editor, *DISC*, volume 1499 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 1998.

[FM00] Matthias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In Frances Yao, editor, *Proc. 32nd ACM Symposium on Theory of Computing — STOC 2000*, pages 494–503. ACM, May 2000.

[FWW04] Matthias Fitzi, Stefan Wolf, and Jürg Wullschleger. Pseudo-signatures, broadcast, and multi-party computation from correlated randomness. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 562–578. Springer, 2004.

[GGBS10] Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. Authenticated byzantine generals in dual failure model. In *Proceedings of the 11th International Conference on Distributed Computing and Networking*, ICDCN'10, pages 79–91, Berlin, Heidelberg, 2010. Springer-Verlag.

[GGO12] Juan A. Garay, Clint Givens, and Rafail Ostrovsky. Broadcast-efficient secure multiparty computation. *IACR Cryptology ePrint Archive*, 2012:130, 2012.

[GGOR13] Juan A. Garay, Clint Givens, Rafail Ostrovsky, and Pavel Raykov. Broadcast (and round) efficient verifiable secret sharing. In Carles Padró, editor, *ICITS*, volume 8317 of *Lecture Notes in Computer Science*, pages 200–219. Springer, 2013.

[GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *Proceedings of the 48th Annual*

IEEE Symposium on Foundations of Computer Science, FOCS '07, pages 658–668, Washington, DC, USA, 2007. IEEE Computer Society.

[GKKY10] S. Dov Gordon, Jonathan Katz, Ranjit Kumaresan, and Arkady Yerukhimovich. Authenticated broadcast with a partially compromised public-key infrastructure. In Shlomi Dolev, Jorge Arturo Cobb, Michael J. Fischer, and Moti Yung, editors, *SSS*, volume 6366 of *Lecture Notes in Computer Science*, pages 144–158. Springer, 2010.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[HLM13] Martin Hirt, Christoph Lucas, and Ueli Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 203–219. Springer, 2013.

[HM97] Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In James E. Burns and Hagit Attiya, editors, *PODC*, pages 25–34. ACM, 1997.

[HMP00] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 143–161. Springer-Verlag, December 2000.

[HMQU06] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. On the (Im-)Possibility of extending coin toss. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 504–521. Springer, 2006.

[HMR14] Martin Hirt, Ueli Maurer, and Pavel Raykov. Broadcast amplification. In Yehuda Lindell, editor, *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 419–439. Springer, 2014.

[HR13a] Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: the $t < n$ case. Cryptology ePrint Archive, Report 2013/553, 2013. `http://eprint.iacr.org/`.

[HR13b] Martin Hirt and Pavel Raykov. On the complexity of broadcast setup. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP (1)*, volume 7965 of *Lecture Notes in Computer Science*, pages 552–563. Springer, 2013.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[KK06a] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *In Advances in Cryptology—Crypto '06*, pages 445–462. Springer-Verlag, 2006.

[KK06b] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. Cryptology ePrint Archive, Report 2006/065, 2006. `http://eprint.iacr.org/`.

[KK07] Jonathan Katz and Chiu-Yuen Koo. Round-efficient secure computation in point-to-point networks. In *EUROCRYPT '07*, volume 4515 of *LNCS*, pages 311–328, 2007.

[KKK08] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of vss in point-to-point networks. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510. Springer, 2008.

[KS11] Valerie King and Jared Saia. Breaking the $o(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. *J. ACM*, 58(4):18, 2011.

[Lam83] Leslie Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.

[LV10a] Guanfeng Liang and Nitin Vaidya. Complexity of multi-value byzantine agreement. Technical report, University of Illinois at Urbana-Champaign, 2010. Available at `http://www.crhc.illinois.edu/wireless/papers/ba_sum_capacity_0729.pdf`.

[LV10b] Guanfeng Liang and Nitin Vaidya. Short note on complexity of multi-value byzantine agreement. *CoRR*, abs/1007.4857, 2010.

[LV11a] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with Byzantine failures. In *Proceedings of the 30th annual ACM symposium on Principles of distributed computing*, PODC '11, pages 11–20, New York, NY, USA, 2011. ACM. The arxiv version is available at http://arxiv.org/abs/1101.3520.

[LV11b] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. *CoRR*, abs/1101.3520, 2011.

[Mau04] Ueli M. Maurer. Towards a theory of consistency primitives. In Rachid Guerraoui, editor, *DISC*, volume 3274 of *Lecture Notes in Computer Science*, pages 379–389. Springer, 2004.

[Pat11] Arpita Patra. Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity. In *Proceedings of the 15th international conference on Principles of Distributed Systems*, OPODIS '11, pages 34–49. Springer, 2011.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[PW96] Birgit Pfitzmann and Michael Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. Technical report, IBM Research, 1996.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.

[TC84] Russell Turpin and Brian A. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[Yao79] Andrew C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC '79, pages 209–213, New York, NY, USA, 1979. ACM.