Diss. ETH No. 14761

Generalized Communication and Security Models in Byzantine Agreement

A dissertation submitted to

ETH ZÜRICH
(SWISS FEDERAL INSTITUTE OF TECHNOLOGY)

for the degree of
Doctor of Technical Sciences

presented by

Matthias Fitzi
Dipl. Informatik-Ing. ETH

born April 29, 1970, in Basel
citizen of Gais AR, Switzerland

Prof. Dr. Ueli Maurer, examiner
Dr. Michael Waidner, co-examiner

2002

to my parents

Konrad and Renate

Acknowledgments

This thesis originated under the supervision of Ueli Maurer. I thank him for introducing me to a fascinating field of research, for his competent guidance, as well as for many inspiring discussions. I thank Michael Waidner for spontaneously accepting to be the co-referee of this thesis.

A lot of this research was done in close collaboration with Martin Hirt. I thank him for the good team work, a countless number of discussions, and a lot of support and inspiration.

It was a great pleasure to collaborate with my co-authors, Bernd Altmann, Juan Garay, Nicolas Gisin, Daniel Gottesman, Thomas Holenstein, Rafi Ostrovsky, Oliver von Rotz, Adam Smith, and Jürg Wullschleger. In addition, I thank Juan Garay and Nicolas Gisin for their kind invitations and their hospitality. I am also grateful to Matt Franklin for making some valuable suggestions to improve this text.

It was always enjoyable to work in the Cryptography and Information Security Research Group at ETH Zürich; I thank its numerous members, previous and present, for the always motivated yet relaxed environment. Among them are Masayuki Abe, Jan Camenisch, Ronald Cramer, Thomas Dübendorfer, Stefan Dziembowski, Serge Fehr, Clemens Holenstein, Olaf Keller, Reto Kohlas, Thomas Kühne, Julien Marcil, Lennart Meier, Krzystof Pietrzak, Bartosz Przydatek, Renato Renner, and Stefan Wolf. Additional thanks go to the people who appeared around IBM Zürich, including Anna Lysyanskaya, Reto Strobl, and many others.

Furthermore, I thank Beate Bernhard, Ortrud Milos, and Helena Sechser for their administrative support, the members of Stabstelle Software for their efficient way of solving the software problems, the members of Stabstelle Hardware for their less efficient way of solving the hardware ones, and the cleaning staff of IFW / RZ for their excellent service.

This research was partially supported by the Swiss National Science Foundation (SNF).

Abstract

Byzantine agreement (BA) is a primitive of fundamental importance for fault-tolerant distributed computing and cryptographic protocols. BA among a set of n players allows them to reach agreement about a value even if some of the players are malicious and try to prevent agreement among the non-faulty players by distributing false information.

Since the initial statement of the BA problem, a small number of widely accepted standard models have established, distinguishing between aspects such as what means of communication are given among the players or how powerful the faulty players are. Both in research on Byzantine agreement and its applications, these standard models are obstinately followed.

Besides a selective overview on some standard models in Byzantine agreement, this thesis gives a broader view on the problem by considering natural generalizations of these models and generalizations of the problem definition itself. Thereby the main focus is on synchronous networks and active adversaries. It turns out that some of these generalizations, without restricting the adversarial power, allow for BA protocols that achieve a level of security that is provably unachievable in their corresponding standard models. The main contributions are described in the following paragraphs whereby n denotes the number of players and t the number of cheaters among the players.

Security. Standard BA provides either unconditional or computational security. Unconditionally secure protocols for BA are provably secure but can only tolerate a relatively small number of cheaters, typically $t < n/3$. Computationally secure ones often tolerate any number of cheaters, $t < n$, but their security is based on unproven intractability assumptions. So far, every previous computationally secure protocol from the literature has the property that, in case its underlying intractability assumption is

false, it does not withstand one single cheater, $t = 0$. In contrast, we show that computational and unconditional security can be combined by presenting protocols computationally secure against some large number t_1 of cheaters but, at the same time, still unconditionally secure against some smaller number $t_0 > 0$ of cheaters. It is shown that BA of this flavor is achievable if and only if $2t_0 + t_1 < n$ and $t_1 \geq t_0$.

Communication. Standard communication models assume either pairwise authenticated or pairwise secure channels among the players. In these models, unconditional BA is achievable if and only if $t < n/3$. A natural generalization of these models is to assume partial broadcast among the players to be possible, i.e., that for some number $b \geq 2$, broadcast is achievable among each set of b players. It is shown that for any b, $2 \leq b \leq n$, BA is achievable if and only if $t < \frac{b-1}{b+1}n$.

New threshold paradigm. The security of standard BA is defined with respect to one threshold t meaning that BA is achieved in the presence of up to $f \leq t$ cheaters but that no security is guaranteed at all if $f > t$. In particular, unconditionally secure protocols are completely insecure in the presence of $f \geq n/3 > t$ cheaters. However, in reality, nothing would really guarantee that $f \leq t$ and thus the usefulness of non-fully resilient protocols is questionable. Preferably, a non-fully resilient protocol should guarantee BA for some threshold t — but in case that more than t players are cheating, $f > t$, and BA cannot be achieved, it should be guaranteed that all players safely abort the protocol in unison. We show that this is possible if and only if $t = 0$. More generally, we introduce the notion of two-threshold BA, involving two different thresholds t_v and t_c: if at most t_v players cheat then the "validity condition" of BA is achieved and, if at most t_c players cheat then the "consistency condition" of BA is achieved. We show that two-threshold BA is achievable if and only if both $t_v + 2t_c < n$ and $2t_v + t_c < n$, or $t_v = 0$, or $t_c = 0$.

Zusammenfassung

Byzantine Agreement (BA) ist eine Primitive von fundamentaler Wichtigkeit für fehlertolerante verteilte Berechnungen oder kryptographische Protokolle. BA unter n Spielern erlaubt ihnen, sich auf einen Wert zu einigen, auch wenn einige der Spieler betrügerisch sind und versuchen, durch das Versenden falscher Information zu verhindern, dass sich die ehrlichen Spieler auf den selben Wert einigen.

Seit der Erstformulierung des BA-Problems hat sich eine kleine Zahl weit akzeptierter Standardmodelle etabliert, die Aspekte wie Mächtigkeit der Betrüger oder Art der Kommunikation zwischen den Spielern unterscheidet. In der Erforschung von Byzantine Agreement und dessen Anwendungsgebieten werden diese Standardmodelle stur verfolgt.

Nebst einer selektiven Übersicht solcher Standardmodelle für BA bietet diese Dissertation eine umfassendere Sicht auf das Problem, indem natürliche Verallgemeinerungen dieser Modelle und auch der Problemstellung selbst untersucht werden. Von Hauptinteresse sind dabei synchrone Netzwerke und aktive Gegner. Es stellt sich heraus, dass gewisse dieser Verallgemeinerungen, ohne dabei die Macht des Gegners einzuschränken, BA-Protokolle ermöglichen, deren Sicherheitsniveau im entsprechenden Standardmodell beweisbar unmöglich ist. Die Hauptbeiträge dieser Dissertation sind in den folgenden Abschnitten beschrieben. Dabei sei n die Anzahl der Spieler und t die Anzahl der Betrüger unter den Spielern.

Sicherheit. Standard-BA bietet entweder unbeschränkte oder berechenmässige Sicherheit. Unbeschränkt sichere BA-Protokolle sind beweisbar sicher, können aber nur verhältnismässig wenige Betrüger tolerieren, üblicherweise $t < n/3$. Berechenmässig sichere Protokolle tolerieren häufig beliebig viele Betrüger, $t < n$, aber deren Sicherheit basiert auf unbewiesenen Schwierigkeitsannahmen. Jedes bisherige solche Protokoll

aus der Literatur hat die Eigenschaft, dass es keinen einzigen Betrüger toleriert, $t = 0$, falls die zu Grunde liegende Schwierigkeitsannahme falsch ist. Im Gegensatz dazu zeigen wir, dass berechenmässige und unbeschränkte Sicherheit kombiniert werden können, indem wir Protokolle präsentieren mit berechenmässiger Sicherheit gegen eine grössere Zahl t_1 von Betrügern und zusätzlicher unbeschränkter Sicherheit gegen eine kleinere Anzahl $t_0 > 0$ von Betrügern. Es wird gezeigt, dass BA dieser Art genau dann möglich ist, wenn $2t_0 + t_1 < n$ und $t_1 \geq t_0$ gilt.

Kommunikation. Standard-Kommunikationsmodelle setzen entweder paarweise authentische Kanäle oder paarwiese sichere Kanäle unter den Spielern voraus. In diesen Modellen ist unbeschränkt sicheres BA genau dann möglich, wenn $t < n/3$ gilt. Eine natürliche Verallgemeinerung besteht daraus, partiellen Broadcast unter den Spielern vorauszusetzen, d.h., dass für eine bestimmte Zahl $b \geq 2$ Broadcast unter jeder Menge von b Spielern möglich ist. Es wird gezeigt, dass für beliebiges b, $2 \leq b \leq n$, BA genau dann möglich ist, wenn $t < \frac{b-1}{b+1}n$.

Neues Schwellen-Paradigma. Die Sicherheit von Standard-BA ist bezüglich einer Schwelle t definiert mit der Bedeutung, dass BA erreicht wird, falls bis zu $f \leq t$ Betrüger unter den Spielern sind, dass aber nicht die geringste Sicherheit garantiert ist, falls $f > t$. Insbesondere sind unbeschränkt sichere Protokolle völlig unsicher, falls $f \geq n/3 > t$ Betrüger anwesend sind. In der Realität würde jedoch nichts garantieren, dass tatsächlich nur $f \leq t$ Spieler betrügen, und deshalb ist der Nutzen von Protokollen, die nicht beliebig viele Betrüger tolerieren, fragwürdig. Vorzugsweise sollte ein solches Protokoll für eine bestimmte Schwelle t BA erreichen aber im Falle, dass mehr als t Spieler betrügen, $f > t$, und BA nicht erreicht werden kann, sollte garantiert sein, dass alle Spieler zusammen das Protokoll wohlbehalten abbrechen. Es wird gezeigt, dass dies genau dann möglich ist, wenn $t = 0$. Allgemeiner noch wird der Begriff des Zwei-Schwellen-BA eingeführt, das bezüglich zweier Schwellen t_v und t_c definiert ist: falls höchstens t_v Spieler betrügen, erreicht das Protokoll die "Validity-Bedingung" von BA, und falls höchstens t_c Spieler betrügen, errreicht das Protokoll die "Consistency-Bedingung" von BA. Es wird gezeigt, dass Zwei-Schwellen-BA genau dann möglich ist, wenn $t_v + 2t_c < n$ und $2t_v + t_c < n$ gilt, oder $t_v = 0$, oder $t_c = 0$.

Contents

Chapter 1

Introduction

The goal of information security and cryptography is to maintain the integrity of data with respect to several aspects. Two prominent aspects are authenticity and privacy of information. Authenticity of a piece of data or a statement with respect to an issuer means that the issuer indeed uttered this statement. Privacy of a piece of data means that no unauthorized being is able to gain any information about it.

Consider two parties Annegret and Beat communicating over the Internet. Using an authentication mechanism for their communication asserts them that an adversarial third party cannot modify nor introduce any piece of information on behalf of either Annegret or Beat. Using an encryption mechanism asserts them that no adversarial third party can gain information about their communication by eavesdropping (or even tampering with) their communication channel.

Traditionally, as in the example given above, such mechanisms are used in order to protect the cooperation among mutually trusted parties against adversarial threats from outside. However, mostly, cooperation takes place among parties that do not necessarily trust each other. For example, in a shop, both the customer and the salesperson want to cooperate in order to close a trade. But still, the customer would check the received change at the cash point.

Secure cooperation among mutually distrusted parties is a relatively new paradigm in the field of information security. In case that such a cooperation involves more than two parties an important aspect is consistency among the honest parties: an adversarial party should not be able to enforce different honest parties to obtain mutually inconsistent views.

Consider, for example, a global company that has three subsidiaries distributed all over the world, each managed by its own director. Each pair of directors can communicate bilaterally in an authenticated way, e.g., by means of their cell phones, but it is not possible to physically broadcast information in a way that all directors are guaranteed to get the same message. Suppose now, that the directors want to reach agreement on a common action to be performed in unison by all subsidiaries, but that an arbitrary one of the three directors is possibly corrupted by a company in competition, trying to enforce the honest directors to perform contrary actions. Two natural conditions can be required for such a task. First, all honest directors must decide in the same way. Second, if all honest directors vote for the same action then this must be the action they finally decide on. This problem is a special case of so-called *Byzantine agreement*. Note that already this simple instance of the problem has no trivial solution: It can be proven that this problem cannot be solved by having the directors exchange plain messages but that "cryptographic mechanisms" are necessary to solve this problem.

More generally than in the example above, Byzantine agreement (BA) is defined with respect to a number n of players and a threshold t indicating the maximal number of corrupted players among them; and the goal is that all honest players reach agreement on the same output value depending on the initial configuration of the honest players. BA has a wide range of applications. For instance, it serves as a building-block for redundant systems in airplanes, for the assertion of consistency among replicated databases, for fault-tolerant distributed computing, or for electronic voting.

The BA problem was defined in [PSL80] by Pease, Shostak, and Lamport. In turn, several models for BA have been proposed during the last twenty years, distinguishing aspects such as how parties communicate, what security level is required, or how powerful the corrupted parties are. Some of these models have established as widely accepted standards and are being obstinately followed in research on BA and its applications.

Together with a selective overview on some standard models for BA, some natural generalizations of these models and of the problem definition itself are introduced. The reason for considering generalizations of standards is that, although tight bounds are known for the achievability of BA in those standard models, strictly more powerful protocols can be found when considering a more general view. One possible generalization is to combine computational and unconditional security. For example, we show that some optimally resilient protocols with compu-

tational security can be augmented with additional unconditional security up to a certain level — which is strictly more than achievable in the original model only considering computational security. As another example, it is possible to extend the standard definition of BA (with respect to a threshold t) to still satisfy partial security conditions of BA for the case that more than t players are corrupted — without losing any security for the case that only up to t players are corrupted. In particular, there is a protocol that achieves unconditionally secure BA if no players are corrupted but, additionally, that all players finally agree on the same output value even if any number of players are corrupted. Another benefit from a more general view on the problem is its reduction to small building-blocks with weaker properties than BA that are much easier to implement. Such reductions can then be used in order to prove the achievability of BA with certain properties in other (not necessarily defined yet) models: in order to prove standard BA to be achievable, the mere achievability of one of its weak forms has to be demonstrated.

As a special field of interest, it is shown how these results apply to general multi-party computation (MPC), a problem that subsumes a large class of problems such as electronic voting, etc.

Outline. Chapter 2 introduces some basic definitions and concepts regarding topics such as computation, complexity, or cryptographic primitives. The definitions are thereby kept at a minimal informal level required by the following chapters.

Chapter 3 gives a selective overview on the established standard models and their related achievability and impossibility results concerning fault-tolerance and efficiency. Thereby, six particular models are of interest, characterized by the distinctions whether the network is synchronous or asynchronous, whether unconditional or computational security is required, and whether corruption is active or fail-stop.

The remaining chapters are focused on synchronous networks and on active corruption of players (which is the most severe form of corruption). Thereby two different models are considered. In the first model, besides knowing the player set and the protocol to be executed, the players do not share any consistent information such as a public-key infrastructure (PKI). In the second model, the players additionally share consistent information in form of a PKI with respect to a digital signature scheme. This distinction breaks with the tradition of primarily distinguishing between unconditional and computational security since, besides of computational security allowing for more efficient protocols, unconditional

security is achievable for Byzantine agreement basically in every model where computational security is.

Chapter 4 elaborates on some major results from the literature. Thereby, the results are mostly represented in an alternative way such as giving modular constructions that build up from small functionalities until the full functionality is reached, or representing known constructions in a way that may be more intuitive than originally. For the model where the players do not share a PKI, several standard protocols from the literature are described that are secure against $t < n/3$ corrupted players. Furthermore, a standard proof is given that, in this model, no protocol can be secure if $t \geq \lceil n/3 \rceil$. For the model where the players do share a PKI, a standard protocol is presented that is secure against any number of corrupted players, $t < n$. Moreover, for this model, a new security model is introduced where protocols do not exclusively rely on the consistency of the shared PKI nor on the security of the underlying signature scheme. Thereby the goal is to construct protocols with respect to two thresholds t_σ and $t_u \leq t_\sigma$ that are as secure as the underlying signature scheme as long as up to t_σ players get corrupted but still unconditionally secure as long as up to t_u players get corrupted. We show that BA of such "hybrid security" is achievable if and only if $t_\sigma + 2t_u < n$. This result is of special interest for MPC. It is well-known that, when given a consistent PKI with respect to a digital signature scheme, computationally secure MPC is achievable if and only if $t < n/2$ players get corrupted. Loosely speaking, this result implies that unconditional security can additionally be guaranteed "for free" for the case that only up to $t \leq n/4$ players get corrupted.

In Chapter 5, for the case where the players do not share a PKI, the standard network model is extended, where players can only communicate over pairwise channels. A natural generalization of this model is to assume partial broadcast among the players to be possible, i.e., that for some number $b \geq 2$, broadcast is achievable among each set of b players. For example, $b = 2$ refers to the standard model where each pair of players can communicate bilaterally, and $b = 3$ refers to the case where each player can send messages to any two other players such that it is guaranteed that both recipients get the same message. For any b, $2 \leq b \leq n$, it is shown that, given broadcast among each subset of b players, global BA is achievable if and only if $t < \frac{b-1}{b+1}n$. For instance, this implies that, given broadcast among each triple of players ($b = 3$), global BA is achievable if and only if $t < n/2$. This implies that the special case of $b = 3$, the minimal extension over pairwise channels ($b = 2$), allows for unconditionally

secure multi-party computation secure against $t < n/2$ corrupted players which previously required to assume fully-resilient global broadcast channels. Finally, a second natural extension over pairwise communication assumes the additional existence of an external information source that is accessible by the players.

Chapter 6 introduces a new threshold paradigm for BA, motivated by the unsatisfactory standard definition of BA: The security of standard BA is defined with respect to one threshold t meaning that all conditions of BA are required to be satisfied if up to t players get corrupted but that nothing is guaranteed if more than t players get corrupted. However, what should guarantee that the given threshold t will not be exceeded by the number f of actual corruptions? Preferably, such a protocol should guarantee BA for some threshold t — but in case that $f > t$ and BA cannot be achieved, it should be guaranteed that all players safely abort the protocol in unison. Hence, the worst any adversary could achieve in such a protocol would be a denial-of-service attack but never any inconsistencies could be enforced among the correct players. For this, we introduce the notion of detectable broadcast that involves two different thresholds t and $T > t$: if at most t players are corrupted then BA is achieved and all players accept the outcome of the protocol but still, if at most T players are corrupted then either all honest players reject the outcome, or all honest players accept implying that BA has been achieved. It is shown that detectable broadcast is achievable if and only if $t + 2T < n$ or $t = 0$. This implies that full resilience ($T < n$) is only possible for $t = 0$. More generally, the notion of two-threshold BA is introduced, involving two different thresholds t_v and t_c: if at most t_v players are corrupted then the "validity condition" of BA is achieved and, if at most t_c players are corrupted then the "consistency condition" of BA is achieved. We show that two-threshold BA is achievable if and only if both $t_v + 2t_c < n$ and $2t_v + t_c < n$, or $t_v = 0$, or $t_c = 0$. The given results for detectable broadcast are of special interest for MPC. It is well-known that broadcast channels are required in order to achieve unconditionally secure MPC for $t < n/2$. We show that a weaker form of unconditionally secure MPC for $t < n/2$ is still achievable without broadcast channels by presenting a protocol that either achieves MPC as secure as in the model with broadcast channels or wherein all honest players commonly abort the protocol even before entering a private input.

Chapter 2

Foundations

2.1 Computation and complexity

Algorithms (or programs) take a variable input x from some domain $\mathcal{D}$ ($x \in \mathcal{D}$) and, depending on this input, compute an output $y = f(x)$ of some range $\mathcal{R}$. Deterministic algorithms use no randomness and, on the same input, always compute the same output. Probabilistic (or randomized) algorithms have access to a random source and two executions with the same input may result in completely different outcomes. As a standard, the notion of the Turing machine is used for the formal description of a computation model.

Fundamental properties of algorithms are whether or not they always compute a correct output (as ideally specified), and how long they take to compute the output.

Deterministic algorithms are usually required to terminate in any case and thereby compute a correct output. In contrast, probabilistic algorithms are neither necessarily required to always terminate nor to always compute a correct output upon termination. Of course, for such "non-perfect" algorithms it is desirable that their probability of failure be small.

The computational complexity of an algorithm is defined in dependence of the input size n, typically its number of bits. The computational worst-case complexity is the maximal number of elementary computation steps required for an input of size n. As a standard, the computational complexity of algorithms is stated asymptotically.

Definition 2.1 (O and Ω). *Let g be a function from the natural numbers to the real numbers, $g : \mathbb{N} \to \mathbb{R}$.*

O-Notation: $O(g)$ denotes the set of all functions $f : \mathbb{N} \to \mathbb{R}$ that are upper-bounded by the function g:

$$O(g) := \{f : \mathbb{N} \to \mathbb{R} \mid \exists c > 0, n_0 \, \forall n \geq n_0 : f(n) \leq c \cdot g(n)\} \ .$$

Ω-Notation: $\Omega(g)$ denotes the set of all functions $f : \mathbb{N} \to \mathbb{R}$ that are lower-bounded by the function g:

$$\Omega(g) := \{f : \mathbb{N} \to \mathbb{R} \mid \exists c > 0, n_0 \, \forall n \geq n_0 : f(n) \geq c \cdot g(n)\} \ .$$

A function f is called polynomially upper-bounded *(or* polynomial*, for short) if there is a constant $k \geq 0$ such that $f \in O(n^k)$.* ◇

Note that it is common to write $f = O(g)$ or $f = \Omega(g)$ instead of $f \in O(g)$ or $f \in \Omega(g)$.

A deterministic algorithm with a polynomially upper-bounded computational worst-case complexity is called *polynomial*.

A probabilistic algorithm with a polynomially upper-bounded computational worst-case complexity that, for all inputs, always computes a correct output with probability strictly greater than $\frac{1}{2}$ is called *probabilistic polynomial* (*PP*) or an *algorithm of type "Monte Carlo"*.

A probabilistic algorithm with a polynomially upper-bounded computational average-case complexity that is not guaranteed to always terminate but, upon termination, always computes a correct output is called *probabilistic polynomial with zero error* (*ZPP*) or an *algorithm of type "Las Vegas"*.

An algorithm is called to be *efficient* if it is (probabilistic) polynomial, and called *inefficient* otherwise.

2.2 Cryptographic primitives

Two basic tasks of cryptographic primitives are to provide secrecy and authenticity of information. Their security is typically defined in function of a customizable security parameter $\kappa \in \mathbb{N}$ such that the success probability of an adversary to violate the specified security conditions of the system is negligible as a function of κ.

Definition 2.2 (Negligibility). *A quantity $\varepsilon(\kappa)$ is called* negligible *in function of κ if, for all $c > 0$, there is a constant κ_0 such that for all $\kappa > \kappa_0$ it holds that $\varepsilon(\kappa) < \frac{1}{\kappa^c}$.* ◇

The negligibility of a quantity is usually demonstrated by showing it to be exponentially small in the parameter κ, e.g., $\varepsilon(\kappa) \leq 2^{-\kappa}$.

A cryptographic primitive can be computationally or unconditionally secure. *Computationally secure* means that an adversary who is bounded to probabilistic polynomial computations cannot violate the security specifications of the primitive except for some negligibly small probability. *Unconditionally secure* means that even a computationally unbounded adversary cannot violate the security specifications of the primitive except for some negligibly small probability. As a special case, *perfectly secure* means unconditionally secure with zero error probability.

With lack of provable lower bounds, computational security must be based on unproven computational hardness assumptions. Unconditional, however, means ultimately and provably secure.

2.2.1 Encryption schemes

2.2.1.1 Symmetric encryption

A symmetric encryption scheme is a mechanism that, among a set of players that all know the same secret, allows to confidentially exchange information among each other such that nobody outside listening to the communication "gets any information" about its contents.

In particular, if two players Annegret and Beat agree on a secret key K with respect to a symmetric cryptosystem then Annegret and Beat can later exchange messages over an insecure channel in a private way. Symmetric cryptosystems can either provide unconditional [Ver26] or computational (e.g., AES [DR98]) security. See [Kob94, Sim92, Gol01a, Sti02, MOV97] for more information.

2.2.1.2 Public-key encryption

A drawback of symmetric encryption is key-management: each pair of players that wants to communicate privately is required to use a different secret key.

Asymmetric (or public-key) schemes, see for example [DH76, RSA78, ElG85, CS98], overcome this restriction by involving two different keys, a public key for encryption and a secret key for decryption. Among a set of n players, now only n key pairs have to be used instead of $\binom{n}{2}$ in order to allow each pair to communicate privately. Public-key schemes provably can only provide computational security.

2.2.2 Authentication schemes

2.2.2.1 Message-authentication codes

A message authentication code (MAC) is a mechanism that, among a set of players that all know the same secret, allows to convince each other that a certain message was issued by one of these players.

In particular, if two players Annegret and Beat agree on a secret key K with respect to an authentication code then Annegret and Beat can later exchange messages over an insecure channel in an authenticated way. An adversary can neither modify any such message nor introduce a new message in a way such that the recipient accepts it to be authentic. Message authentication codes can either provide unconditional [WC81] or computational (e.g., CBC-MAC with AES [DR98]) security. See [Sim92, Gol01a, Sti02, MOV97] for more information.

2.2.2.2 Digital signatures

A drawback of symmetric authentication schemes is that the set of players who are able to verify authenticity is exactly the set of players who can also authenticate messages.

Digital signature schemes, see for example [DH76, RSA78, GMR88, Sch91], overcome this restriction by involving two different keys, a secret key that allows to sign (authenticate) messages and a public key that allows to verify the authenticity of a signature. Not only can a signer now "prove" authenticity of information to a verifier but also the verifiers among themselves can "prove" to each other authenticity of information with respect to the signer, i.e., in contrast to message authentication codes, digital signatures are arbitrarily transferable between players without losing conclusiveness.

Definition 2.3. *A* digital signature scheme *is a triple of algorithms* $(\mathrm{G}, \mathrm{S}, \mathrm{V})$. *Algorithm* G *(the* key-generation algorithm*) generates a secret key SK and a public key PK for the signer. Algorithm* S *(the* signing algorithm*) takes a message* m*, a secret key SK, and possibly some "randomness"* r *as inputs and generates a signature* $\sigma = \mathrm{S}(m, SK, r)$*. Algorithm* V *(the* verification algorithm*) takes a message* m*, a signature* σ*, and a public key PK as inputs, and outputs*

$$\mathrm{V}(m, \sigma, PK) = \begin{cases} 1 & , \textit{if } \exists r : \sigma = \mathrm{S}(m, SK, r) \ , \\ 0 & , \textit{otherwise} \ . \end{cases}$$

Additionally, with respect to a given public key PK, it is infeasible to compute valid signatures without knowing its corresponding secret key (unforgeability).

◇

There are different standard levels of forgery a signature scheme can be vulnerable to:

Total break: The adversary can either compute the signer's secret key or can find an efficient algorithm to correctly sign arbitrary messages.

Selective forgery: The adversary can create valid signatures for some special class of messages.

Existential forgery: The adversary is able to forge the signature of at least one arbitrary message, not necessarily having control over the signature's corresponding message.

Standard (computational) signature schemes have the property that signatures are arbitrarily transferable between players. Although unconditionally secure signature schemes do not exist in this generality it is possible to construct unconditionally secure signature schemes with restricted transferability for the case that the set of potential verifiers is known in advance — so called *pseudo-signature schemes* [CR90, PW92, PW96].

Definition 2.4. *A* pseudo-signature scheme *among a player set* $P = \{p_1, \ldots, p_n\}$ *with respect to signer* $p_s \in P$ *and transferability* λ *is a triple* $(\mathrm{G}, \mathrm{S}, V)$ *where* G *is a protocol among player set* P*, and* S *and* V *are algorithms. Protocol* G *(the* initialization protocol*) generates a local secret key* SK_s *for* p_s *and local public keys* PK_i *for the players* $p_i \in P \setminus \{p_s\}$.[1] *Algorithm* S *(the* signing algorithm*) takes a message* m *and the secret key* SK_s *as inputs and generates a pseudo-signature* $\sigma = \mathrm{S}(m, SK_s)$*. Algorithm* V *(the* verification algorithm*) takes a message* m*, a signature* σ*, player* p_i*'s version of the signer's public key* PK_i*, and a transfer level* ℓ *(*$1 \leq \ell \leq \lambda$*) as inputs and computes*

$$\mathrm{V}(m, \sigma, PK_i, \ell) = \begin{cases} 1 & \text{, if } (\ell = 1 \wedge \sigma = \mathrm{S}(m, SK_s)) \ \vee \\ & \quad (\ell > 1 \wedge \exists p_j : \mathrm{V}(m, \sigma, PK_j, \ell - 1) = 1)\ , \\ 0 & \text{, otherwise}\ . \end{cases}$$

Except for some negligible probability, an adversary, without knowing the secret key SK_s*, cannot compute a message* m *and a pseudo-signature* σ *such that* $\mathrm{V}(m, \sigma, PK_i, \ell) = 1$ *for any* $p_i \in P$ *and* ℓ *(*$1 \leq \ell \leq \lambda$*).*

[1] Note that these "public keys" PK_i actually are secret verification keys. However, for notional compatibility between ordinary signatures and pseudo-signatures, we stick to the term "public key".

Except for some negligible probability, an adversarial signer p_s cannot compute a message m and a pseudo-signature σ such that, for any $p_i, p_j \in P \setminus \{p_s\}$ and any ℓ_i, ℓ_j ($1 \leq \ell_i \leq \ell_j \leq \lambda$), $\mathrm{V}(m, \sigma, PK_i, \ell_i) \neq \mathrm{V}(m, \sigma, PK_j, \ell_j)$. ◇

In contrast to standard digital signatures, a pseudo-signature scheme is secure against existential forgery by definition.

2.2.2.3 Reusability

A fundamental question is how often a given signature scheme can be used, i.e., how many signatures on different messages a signer can issue without making the particular instance of the scheme insecure, or, in other words, how much information must signer and verifier store in order to allow for s later signatures by the signer.

With respect to a computational signature scheme, the secret-key size (which is a lower bound on the information to be stored) depends on s in order $\Omega(\log s)$. With respect to an unconditional signature scheme, the overall information to be stored among the n players basically depends on s in linear order, $O(s)$, when requiring the scheme to be non-interactive. However, allowing interaction in form of a regeneration protocol [PW96] among the players, the order can be reduced to $O(\log^2 s)$ which is not considerably more than in the computational case.

2.2.3 Key management

Secrecy and authenticity of information is bound to the knowledge and ignorance of keys. Players must be sure that they hold each other's authentic public keys. This can be achieved by physically meeting, or using certificates and trust management.

Informally, a *public-key infrastructure (PKI)* is a setup among players where each player holds some private keys and authentic public keys from other players and possibly trusts some of the other players. Often, it is sufficient that players are pairwise consistent, i.e., that pairs of players hold mutually authentic public-keys from each other: it is not required that a player uses the same secret key with respect to everybody else. Especially, this holds for public-key encryption. Annegret may either use the same secret key with respect to Beat and Benito whose public keys are identical. As well, Annegret may use different secret keys with respect to Beat and Benito whose public keys differ. However, consider, for example, the signing of a mutual contract between Annegret and Beat. For the case that Beat breaks the contract, Annegret wants to be able to convince

judge Jürg that Beat had signed this contract. This not only requires that Annegret and Jürg know an authentic public key of Beat's but also that both of them hold the *same* public key and that Annegret knows that this key is "approved" for the signing of contracts.[2]

Especially, for protocols among a fixed player set $P = \{p_1, \ldots, p_n\}$ relying on digital signatures, it should be guaranteed that, with respect to every player p_i, all players hold the same public key PK_i.

Definition 2.5. *We say that a player set $P = \{p_1, \ldots, p_n\}$* is in possession of a consistent public-key infrastructure for signatures *(or* shares a PKI, *for short) if either*

A) for each player $p_s \in P$, a digital signature scheme $(\mathrm{G}, \mathrm{S}, \mathrm{V})$ is given such that SK_s is the secret key and PK_s is the public key generated by G, p_s exclusively knows secret key SK_s, and all players in P know and exclusively use public key PK_s for the verification of signatures with respect to p_s; or

B) for each player $p_s \in P$, a pseudo-signature scheme $(\mathrm{G}, \mathrm{S}, \mathrm{V})$ is set up among player set P with respect to p_s as the signer.

In case (A) we say that the players share a computational PKI. *In case (B) we say that the players* share an unconditional PKI. ◇

Given that communication among the players is possible, a consistent PKI (with respect to signatures) according to Definition 2.5 allows the players to authentically exchange public keys with respect to a public-key cryptosystem, and hence allows for computationally secure (authenticated *and* private) pairwise communication.[3]

2.3 Cryptographic protocols

Informally, a protocol is an interactive game between players. The players can be humans or machines that are connected with other players via some communication network. A player can perform local computations, can send or receive information to / from the network, and, possibly has access to a (pseudo-)random generator.

[2] More precisely, it is only required that Annegret and Jürg, except for some negligible error probability, reach the same conclusion in $\{0, 1\}$ when verifying a signature on a message m. Pseudo-signatures satisfy this property although different verifiers generally hold different public keys.

[3] Although, not necessarily with respect to the same particular intractability assumption.

More formally, a player (or "processor") is usually defined as a Turing machine (or algorithm) with a "read-write" working tape that additionally shares pairwise tapes with other players (pairwise communication channels) and / or share tapes with more than one other player (global or partial broadcast channels).

An *execution step* by a player consists of receiving a finite (possibly empty) set of messages from the other players, followed by a finite number (possibly zero) of local computation steps and finally, by sending a finite (possibly empty) set of messages to other players. We call the corresponding states *reception stage*, *computation stage*, and *sending stage*.

Throughout this text, by default, the player set is denoted by P, its cardinality by $n = |P|$, and the separate players by p_i ($i = 1, \ldots, n$), i.e., $P = \{p_1, \ldots, p_n\}$. As an exception, for a class of impossibility proofs, we assume the player set to be $P = \{p_0, \ldots, p_{n-1}\}$ for convenience.

A protocol among deterministic Turing machines is called a *deterministic protocol*. A protocol among probabilistic Turing machines is called a *probabilistic protocol* or *randomized protocol*. An instance of such a protocol is said to *terminate* if all players' local algorithms terminate.

2.3.1 Communication

2.3.1.1 Security

As a standard, a complete (i.e., fully connected) network of pairwise *authenticated channels* among the players is assumed. In terms of the tape description above this means that each pair of players p_i and p_j shares a tape that is "write-only" for p_i and "read-only" for p_j and a tape that is "write-only" for p_j and "read-only" for p_i. Thereby the player on the "write-only" side (the sender) has exclusive write-access to the tape in the sense that nobody else in the universe can change the contents of the tape. In particular, this implies that the communication between the players cannot be prevented by an adversary.

Additionally, one can assume that the authenticated channels provide *privacy*, i.e., that the player on the "read-only" side (the recipient) has exclusive read-access to the tape in the sense that nobody else in the universe can read the communication on the channel. A channel that is authenticated and private is called a *secure channel*.

Sometimes, also authenticated broadcast channels among the players are assumed, called *broadcast channels* for short. A broadcast channel among a set of players can be modeled as a tape shared among these players whereby one particular player (the sender) has "read-write" access

and which is of type "read-only" for all remaining players. Additionally, a broadcast channel guarantees that, independently of the sender's behavior, all players receive the same value.

Among a set of k players, there are k possible broadcast channels — one with respect to each player being a sender.

2.3.1.2 Synchronicity

In a (*fully*) *synchronous network* all players synchronously operate in clock cycles whereas each cycle is assigned a unique, globally known time defined by a global clock. Messages being sent during a clock cycle are guaranteed to have arrived at the beginning of the next cycle. Accordingly, each player performs exactly one execution step per cycle: receive, compute, and send.

In a *synchronous network without a global clock* all players operate in synchronous clock cycles but they do not have access to a global clock, i.e., the same clock cycle is not necessarily assigned the same time by all players. Messages being sent during a clock cycle are guaranteed to have arrived at the beginning of the next cycle.

In an *asynchronous network* the players operate in local clock cycles that are not synchronized among the players. Messages being sent by a player are only guaranteed to be delivered eventually.

2.3.2 Adversary

Usually, a central adversary is assumed with respect to some threshold t. This adversary may corrupt up to t of the n players (meaning to take control over them) and centrally coordinate the behavior of these players. Such a player is referred to as being *corrupted* or *faulty*. Non-corrupted players are referred to as being *correct* or *honest*. The adversary has access to all information that is sent over non-private communication channels. A protocol that withstands an adversary with respect to a given threshold t is called to be of *resilience* t or to be *t-resilient*.

2.3.2.1 Corruption flavors

Player corruption: A *passive* adversary can read all internal information of the corrupted players, i.e., she is allowed to read the corrupted players' private inputs and outputs, internal states and random tapes, and all their communication with other players. The adversary cannot influence

the behavior of the players, i.e., she cannot make them deviate from a protocol in any way. This corruption flavor is called *passive corruption*.

A *fail-stop* adversary can, for every single player she corrupts, select an arbitrary point of time in the protocol to make him "die". At this point of time, the player stops the sending and receiving of messages. The adversary can neither read any internal information held by the corrupted players nor can she make them misbehave in any other way than "die". This corruption flavor is called *fail corruption* (or *crash corruption*).

An *active* (or *Byzantine*) adversary may corrupt players by making them misbehave in an arbitrarily malicious way. She can read all their information and make them deviate from the protocol completely.

Network scheduling: It is assumed that the adversary is in power to maliciously schedule the whole network, meaning that, for each message being sent during the protocol, the adversary can arbitrarily delay its delivery within the limits that are guaranteed by the network.

In a model with synchronous networks, during a protocol round, typically, all players perform some local computation and then send some information over the communication channels. In presence of an active adversary, malicious scheduling implies that the behavior of the corrupted players during any round k can be based on all information sent to any corrupted player and all information sent over non-private communication channels already during round k. It is simply assumed that the adversary can undetectedly delay the corrupted players' round actions until the "round-k information" of all correct players has already been spread.

In a model with asynchronous networks, malicious scheduling means that the adversary can arbitrarily delay the delivery of sent messages — but with the restriction that every message must eventually be delivered.

2.3.2.2 Computational power

Typically, the adversary is assumed to be either computationally bounded or the adversarial power is assumed to be unlimited. *Computational security* denotes security against an adversary who is bounded to probabilistic polynomial time computations. *Unconditional security* or *information-theoretic security* denotes security against an unlimited adversary. Analogously to cryptographic primitives, security is typically defined with respect to a security parameter κ, allowing an error probability ε that is negligible in function of κ, i.e., it is tolerated that the respective

task fails with probability at most ε. Unconditional security with zero error probability is called *perfect security*.

2.3.2.3 Adaptiveness

A *static* adversary selects the players to be corrupted at the beginning of the protocol. An *adaptive* adversary dynamically chooses new players to be corrupted at any points of time during the protocol depending on the development of the protocol so far — but corrupting at most t players overall.

Thereby, it is always assumed that, as soon as a player is corrupted, he remains corrupted until the very end of the protocol (a so called *non-mobile* adversary).

2.3.3 Complexities

The complexity of a protocol can be stated with respect to the local computation complexity of the players and with respect to the amount of communication that is required among the players.

The *computational complexity* of a protocol is defined as the maximum over the local computational worst-case complexities of all correct players.

The *communication complexity* of a protocol is defined with respect to two aspects. The *bit complexity* $\mathcal{B}$ of a protocol is the total number of bits to be sent by all correct players during the protocol in the worst case, overall. The *round complexity* $\mathcal{R}$ of a protocol is the maximal number of subsequent execution steps that are required by any correct player in the worst case. More precisely, if the network is synchronous then the round complexity is the maximal number of cycles required by any correct player in the worst case. In particular, also cycles are counted wherein a player does neither communicate nor perform any local computations. A protocol for asynchronous networks does not operate in coordinated rounds and thus there seems to be no reasonable definition for its round complexity. However the round complexity of an asynchronous protocol is often defined nevertheless. We will use the following definition: the *round complexity* $\mathcal{R}$ of an asynchronous protocol is its round complexity when run in a synchronous network.

A protocol is *efficient* if its computational, round and bit complexities are all polynomially bounded in the number n of players and a possible security parameter κ.

Analogously to the definitions for probabilistic algorithms, an efficient probabilistic protocol that always terminates in a polynomial number of rounds but that does not guarantee that all players always compute correct outputs is called a protocol of type "Monte Carlo". A probabilistic protocol that terminates in an average polynomial number of rounds in which, upon termination, the players always compute correct outputs but that does not guarantee termination is called a protocol of type "Las Vegas".

Note that communication between players is typically much more expensive than local computations and hence, in the sequel, the analyses of the protocols' complexities is focused on communication. Their computational complexities are evident and can be easily verified. Whereas the round complexities are exactly stated, the bit complexities are given as asymptotical upper bounds in O-notation.

2.3.4 Knowledge and initialization

It is always assumed that all players know the player set, the protocol, and the whole network topology, i.e., they know which players participate in the protocol and how they are connected with communication channels. In the synchronous case, it is usually also assumed that the players agree on a common point in time when the protocol is to be started.

It can be assumed that, additionally, some (partially secret) data is consistently set up among the players. This could for example be attained by a precomputation phase involving a mutually trusted party to distribute some related information to the players. The shared data would typically be a PKI. In this case we say that the players hold *consistently shared data.* In the case where no such shared data is assumed (except for the common knowledge of the player set, the protocol, and the network topology) we say that the players hold *no consistently shared data.*

2.4 Definitions and notations

2.4.1 Standard models

We will always assume the network to be *synchronous* and the adversary to be *active* — unless explicitly stated otherwise. The security of protocols

is generally stated with respect to an *adaptive* adversary whereas impossibility results are stated with respect to a *static* adversary.

Definition 2.6 ($\mathcal{M}_{\mathrm{aut}}$).

Communication: Complete synchronous network of pairwise authenticated channels.

Adversary: Active threshold adversary.

Initialization: No consistently shared data.

◇

Definition 2.7 ($\mathcal{M}_{\mathrm{sec}}$).

Communication: Complete synchronous network of pairwise secure channels.

Adversary: Active threshold adversary.

Initialization: No consistently shared data.

◇

Definition 2.8 ($\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}}$). *Model $\mathcal{M}_{\mathrm{aut}}$. Additionally, for each player $p_i \in P$ there is an authenticated broadcast channel from p_i to P. The broadcast channels are synchronized with the pairwise communication channels.* ◇

Definition 2.9 ($\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}$). *Model $\mathcal{M}_{\mathrm{sec}}$. Additionally, for each player $p_i \in P$ there is an authenticated broadcast channel from p_i to P. The broadcast channels are synchronized with the pairwise communication channels.* ◇

Definition 2.10 ($\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}$).

Communication: Complete synchronous network of pairwise authenticated channels.

Adversary: Active threshold adversary.

Initialization: Consistently shared data in form of a PKI.

◇

Definition 2.11 ($\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}$).

Communication: Complete synchronous network of pairwise secure channels.

Adversary: Active threshold adversary.

Initialization: Consistently shared data in form of a PKI.

◇

Definition 2.12.
Let $\mathcal{M} \in \{\mathcal{M}_{\mathrm{aut}}, \mathcal{M}_{\mathrm{sec}}, \mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}}, \mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}, \mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}, \mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}\}$. *Then* $\mathcal{M}[\mathrm{u}]$ *refers to standard model* $\mathcal{M}$ *with an unbounded adversary (unconditional security) and* $\mathcal{M}[\mathrm{c}]$ *refers to standard model* $\mathcal{M}$ *with an adversary who is computationally bounded (computational security). An asterisk stands for any of the defined sub-models, e.g.,* $\mathcal{M}_{*}^{\mathrm{pki}}$ *means either* $\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}$ *or* $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}$. ◇

2.4.2 Protocol notation

Protocols are specified with respect to a player set S (which, in case of a sub-protocol, may be only a subset of all players, $S \subset P$) and stated with respect to the local view of player p_i, meaning that all players $p_i \in S$ execute this code in parallel with respect to their own identity i. Player p_i's input is called x_i. For instance,

$$\text{Protocol } \texttt{WeakConsensus}\,(S, x_i)$$

refers to a protocol that solves the "weak consensus" problem among the player set S where each player $p_i \in S$ holds input variable x_i that is loaded with the players input. Often, there is only one player holding an input, for example the sender in a broadcast protocol. Then

$$\text{Protocol } \texttt{Broadcast}\,(S, p_1, x_1)$$

refers to a protocol that solves the "broadcast" problem among the player set S where player p_1 holds input x_1.

If an action is only to be performed by one particular player then this is specified in form of an "if-statement", for example,

$$\texttt{if } i = 1 \texttt{ then SendToAll}(v_1) \texttt{ fi}; \quad \texttt{Receive}(w_i)$$

means that player p_1 sends the value stored in variable v_1 to all players in S and that each player p_i (including p_1) writes the received value to variable w_i.

At the end of a protocol, each player computes an output (usually called y_i), written

$$\text{return } y_i\ .$$

For a protocol, an input domain $\mathcal{D}$ and an output range $\mathcal{R}$ is specified, i.e., each input x_i must be selected from $\mathcal{D}$ and each output y_i is computed in $\mathcal{R}$. From the protocol context also the domain of intermediary local variables and values to be sent between players become clear.

For simplicity, it is not explicitly stated how to handle values received from corrupted players that are outside the (possibly implicitly) specified domain. Such a value is always implicitly assumed to be replaced by a default value or by any arbitrary value inside the specified domain.

2.5 Chernoff and Hoeffding bounds

For a detailed analysis of the protocols in Chapter 5.4, Chernoff and Hoeffding bounds [Hoe63, Chv79] will be applied in order to estimate upper bounds on their error probabilities.

The Chernoff bound gives an upper bound on the probability that of n independent Bernoulli trials the outcome deviates form the expected value by a given fraction.

Let X_i ($1 \leq i \leq n$) be a sequence of independent 0-1 distributed random variables with expected value μ. By $\mathcal{C}(\mu, n, \lambda)$ we denote the Chernoff bound as follows

$$\begin{array}{ll} \lambda < 1: & \mathcal{C}_{\downarrow}(\mu, n, \lambda) = \text{Prob}(\sum_{i=1}^{n} X_i \leq \lambda\, \mu n) \leq e^{-\frac{\mu n}{2}(1-\lambda)^2} \\ \lambda > 1: & \mathcal{C}_{\uparrow}(\mu, n, \lambda) = \text{Prob}(\sum_{i=1}^{n} X_i \geq \lambda\, \mu n) \leq e^{-\frac{\mu n}{3}(\lambda-1)^2} \end{array} \tag{2.1}$$

Furthermore, a bound by Hoeffding can be used to estimate tail probabilities of hyper-geometric distributions. By the term $\mathcal{H}(N, K, n, k)$ we refer to a setting where N items are given of which K are "good". The experiment consists of selecting n out of the N items at random, and $\mathcal{H}(N, K, n, k)$ denotes the probability that at least k of the n selections are "good". Let $t = \frac{k}{n} - \frac{K}{N}$. The following inequation holds for any t such that $0 \leq t \leq 1 - \frac{K}{N}$ [Chv79],

$$\mathcal{H}(N, K, n, k) = \sum_{i=k}^{n} \binom{K}{i} \binom{N-K}{n-i} \binom{N}{n}^{-1} \leq e^{-2t^2 n}\,. \tag{2.2}$$

Chapter 3

Byzantine Agreement and Multi-Party Computation

3.1 Byzantine agreement

There are two major variations of Byzantine agreement, *broadcast* and *consensus*. The goal of broadcast (or the Byzantine generals problem) is to have some designated player, called the sender, consistently send an input value (or message) to all other players. In consensus, every player starts with an input value with the goal to make all players agree on the same output value. If all correct players hold the same input value then the output value is required to be the same as this input value. Both problems are trivial if the input domain $\mathcal{D}$ only contains one element. Thus we will always implicitly focus on the case where $|\mathcal{D}| > 1$ — also for further variations of these definitions.

Definition 3.1 (Broadcast). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P, where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$, achieves* broadcast *(or is a* broadcast protocol*) with respect to P, p_s, and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$.

Consistency (or Agreement): All correct players decide on the same output value, i.e., if $p_i \in P$ and $p_j \in P$ are correct then $y_i = y_j$. ◇

Usually, also "termination" is demanded by the standard definition of broadcast, i.e., that it must be guaranteed that all correct players eventually terminate the protocol except, possibly, for some error probability negligibly small in a security parameter κ. Note that, by the given round-complexity analyses of all protocols described in the sequel, termination is always immediately implied.

The conditions "validity" and "consistency" are of orthogonal flavor. Validity can be easily achieved by having the sender distribute his input value to every recipient. However, this method allows a corrupted sender to violate the consistency condition by simply distributing distinct values. On the other hand, consistency can be easily achieved by having all recipients always output the same value $v \in \mathcal{D}$. However, this method does not satisfy the validity condition since the sender cannot influence the outcome of the protocol. It can already be seen that the solution to the problem must lie somewhere between these two extremal variants, involving several rounds of communication and cross-checking of information between the players.

Definition 3.2 (Consensus). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where every player $p_i \in P$ holds an input value $x_i \in \mathcal{D}$ and finally decides on an output value $y_i \in \mathcal{D}$ achieves* consensus *(or is a* consensus protocol*) with respect to P and $\mathcal{D}$ if it satisfies the following conditions:*

Validity (or Persistency): If all correct players p_i hold the same input value $x_i = v$ then all correct players p_i decide on it, $y_i = v$.

Consistency (or Agreement): All correct players decide on the same output value, i.e., if $p_i \in P$ and $p_j \in P$ are correct then $y_i = y_j$. ◇

When clear from the context, we simply say that a given protocol achieves broadcast (or consensus), neglecting the parameters P, p_s, and $\mathcal{D}$.

It is evident that broadcast and consensus can be trivially achieved for the case $t = 0$, i.e., when no player can get corrupted. On the other hand, there might also seem to exist easy solutions for the cases $t = n$ and $t = n - 1$ since the respective conditions become trivial when there is at most one correct player. However, recall that the threshold t refers to the *maximal* number of players that can get corrupted during the protocol (c.f. Section 2.3.2). Thus, for any given t, a protocol secure against t corrupted players is defined to be a protocol secure against any number f of corrupted players with $0 \leq f \leq t$,[4] and hence these special cases are

[4]It is natural to assume that the adversary can also show a corruption profile that is

non-trivial. For arbitrary resilience, instead of $t \leq n$, we adopt the popular notation $t < n$, which specifies the maximal case involving at least one correct player.

In any model with an active adversary, the definition of consensus can at most allow for a strict minority of corrupted players ($t < n/2$) — otherwise the corrupted players, by majority, would always be able to dictate the outcome independently of the inputs by the correct players.

Proposition 3.1. *Consensus among $n \geq 2$ players, secure against $t \geq n/2$ actively corrupted players is impossible.*

Proof. Let $P_0 \dot{\cup} P_1 = P$ be a partition of the player set into two sets of cardinalities $|P_0| = \max(n-t, 1)$, $|P_1| = \min(t, n-1)$; and let all players $p_i \in P_0$ hold the same input value $y_i = 0$ and let all players $p_j \in P_1$ hold input value $y_j = 1$. The adversary can now choose either $\emptyset$, P_0, or P_1 uniformly at random and corrupt the respective players but having them honestly follow the protocol and thus, a correct player cannot distinguish between correct and corrupted players.

If, at the end, all players compute the same output value v then validity is violated with probability at least $\frac{1}{3}$ since the adversary corrupts P_v with probability $\frac{1}{3}$. If, at the end, the players compute different output values then consistency is violated with probability at least $\frac{1}{3}$ since the adversary does not corrupt any player with probability $\frac{1}{3}$. □

In contrast, the definition of broadcast basically allows for any number of corrupted players since the validity condition only depends on the correctness of the single sender of the protocol.

In the sequel, we mainly focus on binary Byzantine agreement (domain $\mathcal{D} = \{0, 1\}$) since Byzantine agreement for any finite domain $\mathcal{D}$ can be efficiently solved with any binary protocol (cf. Section 3.1.4).

Historically, resilience of protocols is distinguished between unconditional security and computational security. In the former case the players would not be allowed to consistently share any data, e.g., standard model $\mathcal{M}_{\mathrm{sec}}[\mathrm{u}]$. In the latter case it would be implicitly assumed that the players share consistent data, typically in form of a PKI with respect to a given signature scheme, e.g., standard model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}[\mathrm{c}]$ — with the goal that the protocol would be as difficult to break as forging corresponding signatures. In other words, the historical distinction between unconditional and computational security implicitly makes the additional distinction of whether or not the players are in possession of consistently shared data.

strictly weaker than in the worst case.

However, when not making this implicit distinction then, except for the complexity of protocols, there is no real difference between the achievability of computational security and the achievability of unconditional security (cf. Section 3.1.5). Thus, it seems to make more sense to primarily distinguish between the case where no consistently shared data is assumed among the players (nor any reliable precomputation), i.e., plain models $\mathcal{M} \in \{\mathcal{M}_{\mathrm{aut}}, \mathcal{M}_{\mathrm{sec}}, \dots\}$, and the case where consistently shared data among the players (or a reliable precomputation) is assumed, i.e., models $\mathcal{M} \in \{\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}, \mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}, \dots\}$.

Since the goal of this section is to give a short historical overview of the most important results in this field, here, we will follow the tradition of distinguishing between unconditional security with no consistently shared data and computational security with consistently shared data. In the subsequent sections, however, we will only distinguish between whether or not consistently shared data is assumed.

The following sections give a short overview on previous results related to the six possible models resulting from the distinctions whether the network is synchronous or asynchronous, whether unconditional or computational security is required, and whether corruption is active or fail-stop. Thereby, the main points of interest are

- tight conditions on t such that Byzantine agreement is achievable,
- the existence of efficient protocols, and
- the message and round complexities of protocols.

3.1.1 Synchronous networks

The initial problem statement by Lamport, Shostak, and Pease [PSL80, LSP82] refers to synchronous networks, active adversaries, and both unconditional or computational security. Fail-stop adversaries were later considered by Dolev and Strong [DS82], and Lamport and Fischer [LF82]. A summary of the most important bounds with respect to active adversaries and a domain $\mathcal{D}$ of constant size are given in Tables 3.1 and 3.2.

3.1.1.1 Unconditional security

Active corruption. If $n \leq 3t$ then Byzantine agreement perfectly secure against active adversaries is impossible as was already shown in [PSL80, LSP82]. Karlin and Yao [KY84] later proved that this bound more generally holds with respect to (non-perfect) unconditional security. An important, easily generizeable paradigm for impossibility proofs for Byzantine

Security	Determinism	Protocol	Resilience
uncond	DET/RND	BC/C	$n > 3t$
comp	DET/RND	BC	$n > t$
		C	$n > 2t$

Security	Determinism	Protocol	t-constraint	Round-complexity
uncond/comp	DET	BC/C	$n > 3t \Rightarrow$	$t+1$
uncond	RND	BC/C	$n > 3t \Rightarrow$	$\Omega(1)$
comp	RND	BC	$n \leq 2t \Rightarrow$	$t+1$
		BC/C	$n > 2t \Rightarrow$	$\Omega(1)$

Security	Determinism	Protocol	t-constraint	Bit-complexity
uncond	DET/RND	BC/C	$n > 3t \Rightarrow$	$\Omega(nt)$
comp	DET/RND	BC/C	$n \leq 3t \Rightarrow$	$\Omega(nt\|\sigma\|)$

Table 3.1: *Synchronous networks: Lower bounds for broadcast (BC) and consensus (C) with a domain $\mathcal{D}$ of constant size with respect to resilience, round-, and bit-complexity. The bounds for the round and bit-complexities depend on the resilience t to be achieved by the protocol (column "t-constraint"). $|\sigma|$ denotes the maximal size of a signature.*

agreement problems was given by Fischer, Lynch, and Merritt [FLM86]. That this bound is tight was shown in [LSP82] by giving a perfectly secure but inefficient protocol for the case $n > 3t$. The first efficient protocol for $n > 3t$ was given in [DS82], followed by a series of other solutions [DFF+82, TPS87, BDDS92, FM97, BGP89, CW92, GM98].

Fischer and Lynch [FL82] proved that, independently of n, every deterministic protocol for Byzantine agreement requires at least $t+1$ rounds of communication. Although the protocol in [LSP82] is inefficient it requires exactly $t + 1$ rounds and hence is round-optimal. The first *efficient* protocol that is both optimally resilient ($n > 3t$) and round-optimal, was given in [GM98] by Garay and Moses, following solutions with suboptimal resilience (up to $n > 4t$) given in [DRS82, MW94, BGP89, BG93].

Every protocol for Byzantine agreement requires a bit-complexity of at least $\Omega(nt)$ as was proven by Dolev and Reischuk [DR85]. The first protocols to achieve this lower bound were given by Berman, Garay, and Perry [BG89b, BGP92a], and independently by Coan and Welch [CW92] — for a domain $\mathcal{D}$ of constant size. However, for domains $\mathcal{D}$ of general

Security	Determinism	Type	Protocol	Resilience	Round-/	Bit-Complexity
uncond	DET	BC / C	[GM98]	$\mathbf{n > 3t}$	$\mathbf{t+1}$	$O(n^7t^3)$
			[BGP92a]	$\mathbf{n > 3t}$	$O(t)$	$\mathbf{O(nt)}$
			[CW92]	$\mathbf{n > 3t}$	$O(t)$	$\mathbf{O(nt)}$
	RND	BC / C	[FM97]	$\mathbf{n > 3t}$	$O(\kappa)$	$O(nt + \kappa t^7)$
comp	DET	BC	[DS83]	$\mathbf{n > t}$	$\mathbf{t+1}$	$O(n^3\|\sigma\|)$
		C	[DS83]	$\mathbf{n > 2t}$	$\mathbf{t+1}$	$O(n^4\|\sigma\|)$
	RND	BC / C	[Tou84] (*)	$n > 2t$	$O(\kappa)$	$O(\kappa n^3\|\sigma\|)$

(*) requires some additional precomputation besides PKI.

Table 3.2: *Synchronous networks: Protocols for broadcast (BC) and consensus (C) with a domain $\mathcal{D}$ of constant size. The bounds for probabilistic protocols are given with respect to their "Monte Carlo" variants and a maximal error probability of $\varepsilon \leq 2^{-\kappa}$. For the case of computational security, $|\sigma|$ denotes the maximal size of a signature. The bounds represented in boldface are tight — see Table 3.1.*

cardinality, their protocols have a bit complexity of $\Omega(nt \log |\mathcal{D}|)$, which is not known to be optimal. Both protocols are optimally resilient ($n > 3t$) and require $t + o(t)$ rounds of communication. Whereas this round complexity is of optimal order, i.e. $O(t)$, it does not match the exact lower bound of $t + 1$. So far, no protocol is known that is optimal with respect to resilience ($n > 3t$), order of bit-complexity ($O(nt)$), and exact number of rounds ($t + 1$).

The lower bound of $t + 1$ on the number of communication rounds does not apply for probabilistic protocols as was first discovered by Ben-Or [Ben83] and independently by Rabin [Rab83a]. Ben-Or [Ben83] proposed a protocol with resilience $t = O(\sqrt{n})$ that terminates in a constant expected number of rounds. In [Bra87b] Bracha proved the existence of almost optimally resilient protocols ($n \geq (3 + \varepsilon)n$, for any $\varepsilon > 0$) that terminate in an expected number of $O(\log n)$ rounds. The first optimally resilient protocol requiring a constant expected number of rounds was given by Feldman and Micali [FM97] requiring an overall bit-complexity of $O((nt + t^7) \log |\mathcal{D}|)$.

The protocols in [Ben83, Bra87b, FM97] provide perfect security, error probability 0, but can have (rare) non-terminating runs, i.e., they are of type "Las Vegas". Furthermore, these protocols do not guarantee that all correct players simultaneously terminate during the same round. In fact,

$t+1$ communication rounds are necessary in order to guarantee simultaneous termination and error probability 0 [DRS82, DRS90, DM90] — see Section 3.1.4 ("Simultaneous versus eventual Byzantine agreement") for a more precise treatment. However, the protocols in [Ben83, Bra87b, FM97] can all be turned into protocols wherein all correct players simultaneously terminate after a fixed number of rounds linear in a customizable security parameter κ, guaranteeing correctness of the outcome with an error probability exponentially small in κ, i.e., they can be turned into "quasi constant-round" protocols of type "Monte Carlo" with exponentially small error probabilities.

Whereas all cited deterministic protocols and the probabilistic protocols in [Ben83, Bra87b] require only pairwise authenticated channels (since no secret information is involved), the protocol in [FM97] requires secure pairwise channels for the correctness of their common coin.

Fail-stop corruption (or crash failures). Fail-stop corruption was first considered in [DS82, LF82] where efficient, unconditionally secure protocols are given that tolerate any number of corrupted players ($t < n$). As for active corruption, independently of n, every deterministic protocol requires at least $t+1$ communication rounds [DS82, LF82]. More generally, $t+1$ communication rounds are necessary in order to guarantee simultaneous termination and error probability 0 as proven in [DM90]. Even if no single player is corrupted, any agreement protocol requires every player to either send or receive at least one message, and hence $\Omega(n)$ is a lower bound on the bit-complexity also for the case of fail-stop corruption. Galil, Mayer, and Yung [GMY95] showed that this bound can be achieved with respect to unconditional security by giving an optimally resilient ($t < n$) protocol that requires a polynomial number of rounds in t.

3.1.1.2 Computational security

Regarding Section 3.1.1.1, with respect to fail-stop corruption only minor improvements can be expected from weakening security from unconditional to computational, e.g., a resilience of $t < n$ is already achievable unconditionally, and the lower bounds on round and message complexity are still the same with respect to computational security as with respect to unconditional security. Hence, in this section, we focus on active adversaries.

Computational security is typically achieved based on a digital signature scheme. Thereby it is assumed that the players already share a

computational PKI according to Definition 2.5. Note that assuming a PKI with respect to a signature scheme immediately allows for pairwise secure communication (given that one-way functions indeed exist), and hence we only require to assume pairwise authenticated channels for this model (cf. Section 2.2.3).

For this paragraph, we assume that the length of a signature is $|\sigma|$ bits including possible random padding and other additional information, i.e., $|\sigma|$ can basically be seen as being linear in the security parameter. A computationally secure broadcast protocol for $t < n$ was already given in [LSP82], although an inefficient one. Note that the achievability of broadcast for $t < n$ immediately implies the achievability of consensus for $t < n/2$ without any further assumptions (c.f. Proposition 4.11). In [DS83] the first efficient (deterministic) protocol was given requiring $t + 1$ rounds of communication together with a proof that $t + 1$ is a lower bound on the number of rounds for deterministic protocols. This lower bound was independently proven by DeMillo, Lynch, and Merritt [DLM82]. The protocol in [DS83] requires a total number of $O(nt)$ messages with an overall bit-complexity of $O(nt(\log|\mathcal{D}| + t|\sigma|))$. Dolev and Reischuk [DR85] proved that at least $\Omega(nt)$ signatures are required by any protocol tolerating $t < n$, which implies a lower bound on the bit-complexity of $\Omega(t \log|\mathcal{D}| + nt|\sigma|)$.[5]

Rabin presented the first efficient probabilistic protocol in [Rab83a]. It requires an expected constant number of rounds and tolerates $t < n/4$ player corruptions. Feldman and Micali [FM85] constructed an efficient protocol to tolerate $t < n/3$ and running in constant expected time. However, their solution requires a one-time interactive precomputation phase with $\Omega(t)$ rounds of communication. In [Bra87b], Bracha proved that, for any $\varepsilon > 0$, there is a protocol that tolerates $t \leq n/(2 + \varepsilon)$ and runs in an expected number of $O(\log n)$ rounds. For the exact bound $t < n/2$, Toueg [Tou84] finally gave the construction of a "Monte Carlo" protocol that terminates in a fixed number of rounds linear in a customizable security parameter κ and guarantees correctness of the outcome with an error probability exponentially small in κ. The protocol can be transformed into a protocol of type "Las Vegas" requiring a constant expected number of communication rounds. Whereas the protocols in [FM85, Bra87b] only assume that a PKI be shared among the players, those in [Rab83a, Tou84] require some additional data to be set up among the players (once for a

[5] Note that computational security could also be achieved without using an explicit signature scheme. What is actually meant in [DR85] is that $\Omega(nt)$ messages have to be sent from a player p to a player q in a way such that the recipient q can prove to any third player r that he received this message from player p.

life time).

3.1.2 Asynchronous networks

As an important difference to the synchronous model, broadcast as defined in Definition 3.1 at the beginning of this chapter is not possible with an asynchronous network: A correct recipient simply cannot distinguish between the cases where the sender is correct but being arbitrarily delayed and where the sender is corrupted and will not deliver a message at all. Bracha [Bra87a] defined a somewhat weaker version of broadcast where the adversary may cause non-termination:

Definition 3.3 (Asynchronous Broadcast). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ achieves* asynchronous broadcast *(or is an* asynchronous broadcast protocol*) with respect to P, p_s, and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If the sender p_s is correct then all correct players p_i eventually terminate and decide on the sender's input value, $y_i = x_s \in \mathcal{D}$.

Consistency: If any correct player terminates then all correct players terminate deciding on the same output value $v \in \mathcal{D}$. ◇

In contrast to broadcast, Definition 3.2 for consensus still makes sense since each correct player is guaranteed to eventually receive the inputs of at least $n - t$ different players.

Since broadcast (in its original version) cannot be achieved in asynchronous networks, we focus on consensus in this subsection.

Fischer, Lynch, and Paterson [FLP85] proved that in an asynchronous network, already with respect to fail-stop corruption, there is no protocol that achieves consensus with probability 1 and is guaranteed to always terminate — even if only one single player is corrupted ($t = 1$). Thus, all protocols mentioned in this section are inherently probabilistic.

3.1.2.1 Unconditional security

Active corruption. In [Ben83], Ben-Or proposed the first asynchronous protocol with unconditional security, tolerating $t < n/5$ but generally being inefficient. However, for $t = O(\sqrt{n})$, his protocol is efficient and requires a constant expected number of rounds. Feldman [Fel89] gave the first efficient protocol with linear resilience, $n > 4t$. Bracha [Bra87a]

gave the first optimally resilient but inefficient protocol, tolerating $n > 3t$, i.e., the same number of corrupted players as can be tolerated with respect to synchronous networks. Canetti and Rabin [CR93] gave the first efficient protocol with optimal resilience, $n > 3t$, requiring a constant expected number of rounds. In contrast to most probabilistic protocols for broadcast or consensus their protocol cannot be directly transformed into a "Monte Carlo" protocol with a deterministic constant number of rounds and a negligible error probability. Also the bit-complexity of their protocol is very high (more than $\Omega(n^{10})$). For a weaker model where the adversary does not have full control over the network delays, a simpler solution was given by Bracha and Toueg in [BT85].

Fail-stop corruption. No protocol can tolerate more than $t < n/2$ fail-corruptions as was proven in [BT85]. The first protocol to achieve this bound was given in [Ben83]. However this protocol is inefficient. For a weaker model where the adversary does not have full control over the network delays, an efficient solution was given in [BT85].

3.1.2.2 Computational security

Active corruption. A lower bound of $n > 3t$ was proven in [BT85, Tou84]. The first protocol with respect to this model was given by Rabin in [Rab83a], tolerating $t < n/10$. The first efficient protocol with optimal resilience was presented in [Tou84] by Toueg. The protocol terminates in a fixed number of rounds linear in a customizable security parameter κ and guarantees correctness of the outcome with an error probability exponentially small in κ. It can be turned into a protocol of type "Las Vegas" terminating in a constant expected number of rounds. Cachin, Kursawe, and Shoup [CKS00] improved over the protocol in [Tou84], reducing the (expected) message complexity to $O(n^2|\sigma|)$. Besides a shared PKI, all mentioned protocols require some additional precomputation. In [Rab83a, Tou84] this additional precomputation grows linearly in the number of required later broadcasts. In [CKS00] the precomputation grows at most polylogarithmically in the number of required later broadcasts; but the protocol's security is proven only in the so-called "random-oracle" model [BR93]. In [Nie02], Nielsen showed how to overcome this requirement by presenting a cryptographic coin protocol based on standard computational assumptions.

Fail-stop corruption. The lower bound of $n > 2t$ proven in [BT85] also holds for computational security. The protocols in [CKS00] can finally be

adapted to the fail-stop case achieving consensus for $n > 2t$ while being efficient and requiring a constant expected number of rounds.

3.1.3 Variations of Byzantine agreement

Interactive consistency. A problem somewhere between broadcast and consensus called *interactive consistency* was introduced in [PSL80]. There, the goal is to have all players broadcast a value to everybody and all correct players decide on the same vector of broadcasted values. Obviously, interactive consistency is achievable whenever broadcast is achievable.

Firing squad. Burns and Lynch [BL87] introduced the *firing squad problem* with respect to synchronous networks without a common clock with the goal to initialize a global clock among the players, and thus to achieve full synchronicity. More generally, the goal is to make all correct players synchronously perform a common action during the same round, although the players initially do not agree on a point of time when this action is to be performed.

For example, this problem is of special interest if some player wants to "unexpectedly" initiate the execution of a new protocol (even in a fully synchronous network). A correct initiator should be able to make all correct players start the protocol in the same communication round whereas a corrupted player should not be able to initiate a protocol in a way such that not all correct players start simultaneously. In [BL87] an efficient construction is given that transforms any secure protocol for broadcast with $n > 3t$ into a secure protocol for the firing squad problem. More results for this problem were given in [CDDS89] by Coan, Dolev, Dwork, and Stockmeyer, e.g., that no firing squad protocol exists for $n \leq 3t$ even with respect to computational security. Furthermore, in [CDDS89], a protocol for "Byzantine agreement with non-unison start" is given. Given the setup of a consistent PKI, their protocol achieves broadcast for $t < n$ even when not all correct players start the protocol during the same round, i.e., the broadcast can be initiated by any player on the fly.

Strong validity. Consensus protocols usually make the players decide on a default value when not all correct players start with the same input value. Neiger [Nei94] defined the *strong consensus problem* where the output value finally agreed on is required to be the input value of at least one correct player. With respect to synchronous networks, an active adversary, unconditional security, and an input domain of size m ($|\mathcal{D}| = m$)

he proved $n > \max(3, m)t$ to be a tight bound for the achievability of strong consensus. The first efficient protocol with optimal resilience to achieve strong consensus was given in [FG02]. Further models are considered there.

Weak broadcast. A weak form of broadcast was introduced in [Dol82], called *crusader agreement*. As for example done in [GP92, FM00a, GL02], the definition of crusader agreement can be slightly modified, yielding a version which we will call weak broadcast (called "MakeUnique" in [GP92] and "broadcast with abort and strengthened validity" in [GL02]). The idea is to introduce an "invalidity" symbol $\perp$ with the goal that no correct players ever decide on different values that are valid. Note that the term *weak broadcast* [FM00a] is not to be confused with the *weak Byzantine generals* problem introduced by Lamport in [Lam83]. In light of [Lam83], the reuse of the predicate "weak" may seem awkward. On the other hand, its reuse is justified for two reasons. First, Lamport used this term for a problem that he proved to be futile anyway and hence wasting this term for nothing.[6] Second, a problem called *detectable broadcast* (cf. Definition 6.5 in Section 6.3) is strictly stronger and achievable for all cases where the weak Byzantine generals problem is achievable.

Definition 3.4 (Weak Broadcast). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D} \cup \{\perp\}$ (with $\perp \notin \mathcal{D}$) achieves* weak broadcast *with respect to P, p_s, and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$.

Consistency: If any correct player p_i decides on a value $y_i \in \mathcal{D}$ then all correct players p_j decide on a value $y_j \in \{y_i, \perp\}$. ◇

Definition 3.5 (Weak Consensus). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where every player $p_i \in P$ holds an input value $x_i \in \mathcal{D}$ and finally decides on an output value $y_i \in \mathcal{D} \cup \{\perp\}$ (with $\perp \notin \mathcal{D}$) achieves* weak consensus *with respect to P and $\mathcal{D}$ if it satisfies the following conditions:*

Validity: If all correct players p_i hold the same input value $x_i = v$ then all correct players p_i decide on it, $y_i = v$.

[6] ;-)

Consistency: If any correct player p_i decides on a value $y_i \in \mathcal{D}$ then all correct players p_j decide on a value $y_j \in \{y_i, \perp\}$. ◇

Graded broadcast. Graded broadcast [FM97] is a variation of broadcast where, additionally to the output value, every player gets a grade $g \in \{0, 1, 2\}$ on the outcome of the protocol. We present a slightly modified version with binary grades. If any correct player gets grade 1 then all correct players decide on the same output value, i.e., getting grade 1 implies detecting agreement. If the sender is correct then the protocol achieves broadcast and all correct players get grade 1.

Definition 3.6 (Graded Broadcast). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ and a grade $g_i \in \{0, 1\}$ achieves* graded broadcast *(or* gradecast*) with respect to P, p_s, and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$, and get grade $g_i = 1$.

Consistency: If any correct player p_i gets grade $g_i = 1$ then all correct players p_j decide on the same output value, $y_j = y_i$. ◇

Definition 3.7 (Graded Consensus). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where every player $p_i \in P$ holds an input value $x_i \in \mathcal{D}$ and finally decides on an output value $y_i \in \mathcal{D}$ and a grade $g_i \in \{0, 1\}$ achieves* graded consensus *with respect to P and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If all correct players p_i hold the same input value $x_i = v$ then all correct players p_i decide on it, $y_i = v$, and get grade $g_i = 1$.

Consistency: If any correct player p_i gets grade $g_i = 1$ then all correct players p_j decide on the same output value, $y_j = y_i$. ◇

King consensus. King consensus is a form of consensus with a weakened consistency property. It was implicitly introduced in [BGP89].

Definition 3.8 (King Consensus). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players, let $\mathcal{D}$ be a finite domain, and let $p_k \in P$ be a designated player called the* king *(whereas every player knows the identity p_k of the king). A protocol Ψ among P where every player $p_i \in P$ holds an input value $x_i \in \mathcal{D}$ and finally decides on*

an output value $y_i \in \mathcal{D}$ achieves king consensus *with respect to P, p_k, and $\mathcal{D}$, if it satisfies the following conditions:*

Validity: If all correct players p_i hold the same input value $x_i = v$ then all correct players p_i decide on it, $y_i = v$.

Consistency: If king p_k is correct then all correct players p_j decide on the same output value, $y_j = y_k$. ◇

3.1.4 Further aspects

We briefly summarize additional important work in the area where alternative models or problems are considered. Thereby, unless stated otherwise, we focus on synchronous networks, unconditional security, and active adversaries.

Reducing multi-valued to binary Byzantine agreement. Byzantine agreement for any finite domain $\mathcal{D}$ can be efficiently solved with any binary protocol. A simple way to do so is to encode elements from $\mathcal{D}$ in binary and to run $\lceil \log_2 |\mathcal{D}| \rceil$ binary protocols in parallel, one for each bit. A more efficient method was shown by Turpin and Coan in [TC84] for the case of unconditional security against $t < n/3$ corrupted players. First, multi-valued graded broadcast or graded consensus is invoked on the input(s) in $\mathcal{D}$. Then binary consensus on the grade value is invoked. Now, if consensus on the grade value results in output 1 then the players decide on their output (in $\mathcal{D}$) from the multi-valued protocol and otherwise, they decide on some default value $v \in \mathcal{D}$. This multi-valued protocol requires at most 2 more communication rounds than the binary protocol and an overhead in the overall message complexity of $n^2 \log_2 |\mathcal{D}|$ bits over the binary protocol. Hence, the message complexity of multi-valued Byzantine agreement depends on the domain size by a factor of at most n^2.

Multiple invocations of Byzantine agreement. Protocols for Byzantine agreement with signed messages require that messages exchanged during different invocations of the protocol must be uniquely associated with their corresponding invocation as was observed for example by Gong, Lincoln, and Rushby [GLR95].

If only the plain messages were signed then the adversary would be able to "forge" signatures of correct players by a simple replay attack, i.e., by using signatures previously seen during another invocation of the

protocol. This problem can be solved by associating each invocation with a unique protocol ID and including this ID in every message.

Early stopping. Deterministic protocols usually require a fixed number of rounds even when no single player misbehaves. The idea of early stopping is to tailor the protocol such that its number of rounds decreases with the number $f \leq t$ of players that get indeed corrupted until the end of the protocol. This is typically achieved by a mechanism that allows to locally detect agreement among the players and an internal persistency property of the protocol that guarantees that, once achieved, agreement can not be violated anymore by the adversary. This then allows a player to safely stop as soon as he detects agreement. Note that all probabilistic protocols of type "Las Vegas" include such an early-stopping mechanism.

The first such protocol was given by Dolev, Reischuk, and Strong in [DRS82, DRS90] for the case that $n = \Omega(t^2)$. Furthermore, they proved that $\min(t+1, f+2)$ (where f is the number of corrupted players at the end of the protocol) is a lower bound on the number of rounds that an early stopping protocol can achieve. In [MW88], Moses and Waarts gave an efficient protocol with resilience $n > 6t$ that achieves this lower bound of $\min(t+1, f+2)$. A protocol with optimal resilience, i.e. $n > 3t$, that achieves this lower bound was given by Berman, Garay, and Perry [BGP92b]. However, their protocol requires an exponential amount of data to be exchanged between the players.

Goldreich and Petrank [GP90] used the idea of early stopping in order to combine probabilistic constant-expected-round protocols with deterministic (fixed-round) protocols such that the resulting protocol still requires a constant expected number of rounds but requiring at most $t + O(\log t)$ rounds in the worst case. More generally, they observed that almost all existing protocols for Byzantine agreement have a common structural property that can be exploited in order to combine a probabilistic and a deterministic protocol such that the combined protocol terminates both, in the same order of expected number of rounds as the probabilistic protocol, and in almost the same order of number of rounds as the deterministic protocol. Zamsky [Zam96] described a protocol with resilience $n > 8t$ that requires a constant expected number of rounds but is still round optimal ($\min(t+1, f+2)$ rounds) in the worst case.

Simultaneous versus eventual Byzantine agreement. The drawback of probabilistic protocols of type "Las Vegas" or protocols with early stopping, in general, is that they cannot guarantee that all players simultane-

ously terminate the protocol during the same round, i.e., the adversary can always achieve that the players get "desynchronized" by one or more communication rounds. Protocols of this flavor are said to achieve *eventual Byzantine agreement (EBA)*: agreement is eventually achieved but the players do not necessarily terminate at the same time; whereas *simultaneous Byzantine agreement (SBA)* additionally requires that all players terminate at the same time. As proven in [DRS82, DRS90, HM90, DM90], deterministic protocols and probabilistic protocols of type "Las Vegas" require at least $t + 1$ communication rounds in order to guarantee SBA.

For most applications SBA is required, implying that early-stopping protocols (and "Las Vegas" protocols, in particular) cannot be applied.

Extended adversary models. Meyer and Pradhan [MP91], and Garay and Perry [GP92] considered a model where active and fail-stop corruption can occur simultaneously, i.e., that the adversary may corrupt t_a of the players actively and some (distinct) t_f players in a fail-stop manner. They proved that, in this model, Byzantine agreement is possible if and only if $3t_a + t_f < n$ and gave efficient protocols for the achievable cases. Hirt and Maurer [HM00] introduced the notion of a general adversary with respect to secure multi-party computation (see Section 3.2.2, "Extended adversary models"). They implicitly proved that broadcast unconditionally secure against an active adversary is possible if and only if no three elements of the adversary structure cover the full player set. Their very elegant recursive construction for respective multi-party-computation protocols allows for optimally resilient broadcast with computation and bit complexities polynomial in the size of the adversary structure — but generally exponential in n. The first protocols with a bit complexity polynomial in n were given in [FM98, AFM99].

Link- and omission failures. So far, we always assumed *players*, i.e., processors, to be corrupted (plus possibly the scheduling of messages in the network) whereas the channels between the players were assumed to be reliable. In contrast, in a model with link failures the processors are assumed to be reliable but the communication channels may fail. Amongst others, this model is treated by Berman, Diks, and Pelc in [Pel92, BDP97]. A failure type somewhere between fail-stop corruption of processors and link failures was introduced by Hadzilacos [Had83], called omission failures, where processors may fail to send one or more messages. This model is further treated in [CMS89, HH93b].

Overview texts. For further information, the reading of [Lyn96, AW98, HM90, FHMV95, CD89, Can95] is recommended.

3.1.5 Folklore and fairy tales

Also with respect to Byzantine agreement, generally, "more things are known than are true". This section gives a variety of further aspects of the problem that are often wrongfully believed, ignored, or misunderstood. Again, we focus on synchronous networks and active adversaries.

Unconditional versus computational resilience. As already stated at the beginning of this chapter it is often believed that, with respect to general "achievability", there is a fundamental difference between unconditionally secure and computationally secure Byzantine agreement.

"Claim" 3.1. *Given a synchronous network of pairwise authenticated (or secure) channels among the players, unconditionally secure Byzantine agreement is achievable if and only if $t < n/3$ and computationally secure Byzantine agreement is achievable if $t < n$.*

Although not essentially wrong, such claims implicitly rely on a further difference in the models. Whereas, for unconditional security, typically no consistently shared data (see Section 2.3.4) is assumed among the players, this is mostly assumed for the case of computational security. For the case that the players do not hold consistently shared data, Borcherding [Bor96] observed that $n > 3t$ is necessary in order to achieve Byzantine agreement even with respect to computational security. This fact is also implied by an impossibility proof in [FLM86]. On the other hand, Pfitzmann and Waidner [PW96] proved that consistently shared data allows to tolerate $t < n$ even in the unconditional case by giving a construction for unconditional pseudo-signatures (cf. Section 2.2.2). However, for $n \leq 3t$, computationally secure protocols are much more efficient.

Composition of Byzantine agreement protocols. As a reaction to work by Lindell, Lysyanskaya, and Rabin [LLR02] some people started to believe the following

"Claim" 3.2. *Even when given a consistent PKI among the players (model $\mathcal{M}_{\text{sec}}^{\text{pki}}$), parallel composition of Byzantine agreement is impossible if $n \leq 3t$ and sequential composition is not known to be achievable efficiently for general n when $n \leq 2t$.*

Of course, this claim only holds for models where the correct players are rigorously restricted — so called stateless players. A stateless player is only allowed to store data during a precomputation phase whereas data that is stored during an execution of the protocol is irrecoverable after the execution. In a model where players are allowed to coordinate different executions of the protocol (e.g., by storing an ID number between two sequential executions) this claim is obviously false (see also Section 3.1.4 "Multiple invocations of Byzantine agreement").

On the other hand, in [LLR02], a solution is given for the sequential composition of Byzantine agreement among stateless players for $t < n/2$. However, their solution requires an external trusted party to synchronously trigger all the players to execute the protocol, since the protocol does no "firing" à la [BL87] and the players are not allowed to remember how many cycles have passed since the precomputation.

Furthermore, the "stateless players" model implies that, in general, the parallel composition of Byzantine agreement (or the parallel composition of authenticated communication) is impossible even for $t = 0$ since a player cannot assign messages received in parallel to their corresponding protocol instances. Additionally, following the same reasoning as in [LLR02], if the authenticated channels have to be simulated with help of insecure channels and authentication mechanisms then the "stateless players" model does not even allow for the sequential composition of Byzantine agreement for any $t > 0$. Hence, their positive result with respect to sequential composition only holds for a model where the authenticated channels are physically given.

Multiple invocations of constant-expected-round protocols. Depending on the application, typically a lot of parallel and sequential invocations of broadcast are required. If the given broadcast protocol is deterministic, the overall round complexity of all invocations together is fixed by the number of sequential executions times the number of rounds of the single protocol. Analogously, given a probabilistic protocol of type "Las Vegas" that requires a constant expected number of rounds (e.g., the one in [FM97]), we get the following

"Claim" 3.3. *Given a protocol Ψ that requires a constant expected number of rounds, the expected round complexity of k parallel invocations of Ψ is constant and the expected round complexity of k sequential invocations of Ψ is $O(k)$.*

In [BE03], Ben-Or and El-Yaniv observed that k parallel invocations of protocols requiring some $O(f(n))$ expected number of rounds are not

necessarily bounded by the same order $O(f(n))$. However, for "typical" protocols as the one in [FM97] where the probability of non-termination decreases exponentially in the number of rounds, the expected number of rounds of k parallel executions is still logarithmically bounded, $O(\log k)$. See also Section 4.1.2.3.

Weak Byzantine generals. The weak Byzantine generals problem defined by Lamport in [Lam83] is the same as the broadcast problem but with a weakened validity condition: all correct players must agree on the same output value which is required to be the sender's input only if no player is corrupted. Evidently, unconditionally secure "weak Byzantine generals" is achievable if $t < n/3$ since its conditions are strictly weaker than those of broadcast.

"Claim" 3.4. *In model $\mathcal{M}_{\mathrm{aut}}$ or $\mathcal{M}_{\mathrm{sec}}$, unconditionally secure "weak Byzantine generals" is not achievable if $n \leq 3t$.*

Indeed, in [Lam83], it is proven that the weak Byzantine generals problem cannot be achieved if $n \leq 3t$, i.e., that this weakened variant does not allow for more players to be corrupted than broadcast. However, this proof is restricted to a model that does not allow for private (local) randomness by the players. Assuming the players to have access to private randomness in fact allows to tolerate $t < n$ as is implied by Theorem 6.9.

3.2 Multi-party computation

Byzantine agreement is a special case of the more general problem of *multi-party computation* (*MPC*), initially defined by Yao [Yao82], where the players want to distributedly evaluate some agreed function(s) on their inputs in a way preserving privacy of their inputs and correctness of the computed result.[7]

More precisely, in a multi-party computation among a player set P with respect to a collection of functions $(f_1, \ldots, f_n)$, every player $p_i \in P$ holds a secret input (vector) x_i and secretly receives an output (vector) $y_i = f_i(x_1, \ldots, x_n)$.

[7] Note that this problem is sometimes called *secure function evaluation* whereas the term multi-party computation would then refer to the more general problem of ongoing computations where several function evaluations might be "intertwined".

From a qualitative point of view, the security of multi-party computation is often broken down to the conditions "privacy", "correctness", "robustness", and "fairness", and ideally, a protocol should satisfy all these properties. However, depending on the adversarial power, a protocol cannot always guarantee all properties. In particular, there are settings where it is unavoidable that the adversary can force correct players to abort the computation.

Privacy. A protocol achieves *privacy* if the adversary cannot learn more about the correct players' inputs than given by the inputs and outputs of the corrupted players.

Correctness. A protocol achieves *correctness* if the correct players' outputs are indeed computed as defined by the functions f_i.

Robustness. A protocol achieves *robustness* if no correct player aborts the protocol.

Fairness. A protocol achieves *fairness* if the adversary gets no information about the correct players' inputs in case that any correct player aborts.

More formally, multi-party computation is modeled by an *ideal process* involving a mutually trusted party τ where the players secretly hand their inputs to τ, followed by τ computing the players' outputs and secretly handing them back to the corresponding players. This model is referred to as the *ideal model* [Bea91, Can00, Gol01b].

The goal of multi-party computation is now to achieve the same functionality in the so-called *real model* where there is no such trusted party such that an adversary gets no advantage compared to an execution of the ideal protocol. A multi-party computation protocol is defined to be secure if, for every adversary $\mathcal{A}$ in the protocol, there is an adversary $\mathcal{S}$ in the ideal model that, with similar costs, achieves (essentially) the same output distribution as the adversary in the protocol. For the case that not all conditions can be satisfied by a real-model execution, it is natural to restrict the trusted party's behavior in its corresponding ideal model to what is in fact achievable in the real model, as is done in [Gol01b] (see, for example, Goldreich's "first malicious model").

In this text, two models are of particular interest. The first model (standard computation) captures all qualitative properties listed above. The second model (non-robust computation) achieves all properties except for robustness. In the sequel, when not explicitly stating the model we refer to standard computation.

Standard computation. In the corresponding ideal model, the players secretly hand their inputs to the trusted party τ who reliably evaluates the functions f_i on the given inputs x_j, and secretly hands each output y_i back to player p_i. A real-model protocol is secure if any real-model adversary $\mathcal{A}$ can be transformed into an ideal-model adversary $\mathcal{S}$ that, by interacting with τ, enforces an output distribution that is indistinguishable from the one $\mathcal{A}$ enforces in the real model whereas the output distribution is the distribution of the adversary's view and the correct players' outputs.

Non-robust computation. The corresponding ideal model is defined as for standard computation — with the only difference that an actively corrupted player can hand the special symbol $\perp$ to τ, in which case τ delivers output $\perp$ to every player meaning that the computation was aborted.

The standard models for multi-party computation primarily distinguish between unconditional and computational security, and between passive and active corruption.

With respect to unconditional security, model $\mathcal{M}_{\mathrm{sec}}[\mathrm{u}]$ (secure bilateral channels and no broadcast channels among the players) is typically assumed independently of whether the adversary is passive or active. We will refer to this model as the *unconditional model*.

With respect to computational security against a passive adversary, also model $\mathcal{M}_{\mathrm{sec}}[\mathrm{c}]$ (secure bilateral channels and no broadcast channels among the players) is assumed. With respect to computational security against an active adversary, model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}[\mathrm{c}]$ (secure bilateral channels and broadcast channels among the players) is assumed. Alternatively to model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}[\mathrm{c}]$, also model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}[\mathrm{c}]$ can be assumed since a PKI allows for private bilateral communication and broadcast. We will refer to these models as the *computational models*.

In the unconditional model, it is demanded that the output distribution produced by any real-model execution be statistically indistinguishable by the one produced by an ideal-model execution. In the computational model, only computational indistinguishability is demanded and $\mathcal{A}$, $\mathcal{S}$, and the transformation from $\mathcal{A}$ to $\mathcal{S}$ are required to be polynomial-time computable.

The following sections give a short selective overview on previous results in this topic whereby we focus on models with synchronous networks. For a more precise treatment of the topic, the reader is referred to [Fra93, Can95, Can00, Gol01b, Cra99, Hir01].

3.2.1 Protocols and impossibility results

Goldreich, Micali, and Wigderson [GMW87] gave the first complete solution to the problem. Their protocols work in the computational model. For the model with a passive adversary they gave an efficient protocol that tolerates any number of corrupted players, $t < n$. For the model with an active adversary they gave an efficient protocol that tolerates any faulty minority, $n > 2t$, which is optimal in the sense that no protocol exists for $n \leq 2t$. Other protocols for the computational model were for instance given by Franklin and Haber [FH96], Gennaro, Rabin, and Rabin [GRR98], and Cramer, Damgård, and Nielsen [CDN01]. In [GMW87, BG89a, GL90, Gol01b] protocols were given that are computationally secure against $t < n$ players but wherein the adversary can force the protocol to prematurely abort, i.e., the protocol achieves privacy and correctness but neither robustness nor fairness.

For the unconditional model, Ben-Or, Goldwasser, and Wigderson, and Chaum, Crépeau, and Damgård [BGW88, CCD88] gave efficient protocols for the passive case that tolerate $t < n/2$ and protocols for the active case that tolerate $t < n/3$. Both bounds are tight. Other protocols with unconditional security were for instance given by Gennaro, Rabin, and Rabin [GRR98] (fast multiplication), Ishai and Kushilevitz [IK00] (constant-round protocols), and Hirt and Maurer [HM01] (player elimination).

For the unconditional model where, additionally, reliable broadcast channels are assumed among all the players (i.e., for "non-standard" model $\mathcal{M}_{\text{sec}}^{\text{bc}}[\text{u}]$), Beaver [Bea89], and Rabin and Ben-Or [RB89], proposed efficient protocols that tolerate $t < n/2$. This bound is tight. A more efficient protocol for this model was given by Cramer et al. [CDD+99].

3.2.2 Further aspects

The role of broadcast. Broadcast is an important building block for MPC protocols with an active or a fail-stop adversary. At some points in a computation it may be necessary that the players vote on how to proceed with the protocol. Thereby it is important that all players receive exactly the same vote from every player since, otherwise, two correct players might proceed with completely different steps of the protocol. For instance, in order to guarantee that the data distributed among correct players remains consistent, MPC protocols often require that subsets of players cross-check their information. In case that such a cross-check reveals any inconsistency, the involved players announce this inconsis-

tency with a public "complaint" and, as a reaction to the complaint, some sub-protocol is invoked among all players in order to correct this error. In case that no inconsistency is revealed the protocol can continue directly without any further measures. Such "complaints" must be guaranteed to be consistently received by all players and hence are usually sent with help of a broadcast protocol (or channel).

Secret sharing. In MPC, players are often required to distribute a secret piece of information among the players without revealing it. This can be achieved with *secret sharing*, introduced by Blakley [Bla79] and Shamir [Sha79]. Secret sharing is a two-phase protocol among a player set P with a designated player, called the *dealer*, and two designated subsets of players $S, R \subseteq P$, the *shareholders* S and the *recipients* R. First, in phase "share", the dealer inputs a secret σ and each player $s_i \in S$ computes a local *share* σ_i. Thereby it is guaranteed that the adversary gets no more information about the secret σ than she has already known before this phase. Second, in phase "reconstruct" (at any later point of time), all shareholders $s_i \in S$ input their shares σ_i and the recipients $r_i \in R$ all reconstruct secret σ. Typically, it holds that $P = S = R$.

Whereas ordinary secret sharing is enough when dealing with a passive adversary, further conditions must be specified for the case that the dealer is corrupted when dealing with an active adversary: at the end of the sharing phase, a unique secret σ must be defined that will be reconstructed by all correct recipients during the reconstruction phase.[8] In other words, it is guaranteed that the dealer is committed to a unique value that cannot be changed after the sharing phase. This is the notion of *verifiable secret sharing*, introduced by Chor, Goldwasser, Micali, and Awerbuch [CGMA85].

Extended adversary models. In contrast to unconditionally secure protocols, computationally secure ones have the disadvantage that they rely on unproven computational assumptions and hence could be completely insecure. Chaum, Damgård, van de Graaf [CDG87] gave the first computational MPC protocol to partly overcome this problem. Their protocol, assuming authenticated broadcast channels among the players, is computationally secure against $t < n/2$ corrupted players but, with respect to one distinct player, unconditional privacy is guaranteed. In [Cha89],

[8] This includes the possibility that the dealer is publicly detected to be cheating where σ would be set to a default value.

for model $\mathcal{M}_{\mathrm{sec}}$, Chaum gave a protocol with respect to a passive adversary that is computationally secure against $t < n$ corrupted players and, at the same time, provides unconditional security for $t < n/2$. In other words, in order to make the protocol fail, the adversary must be able to "break" the computational assumption and, additionally, corrupt half or more of the players. For model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}$ and with respect to an active adversary, he gave a protocol that is computationally secure against $t < n/2$ corrupted players and, at the same time, provides unconditional privacy (but not necessarily correctness) for $t < n/3$. Note that the later protocols in [Bea89, RB89, CDD+99] strictly imply this result.

Another flavor of hybrid security can be defined with respect to a *mixed adversary* that may corrupt some t_a of the players actively, and additionally some t_p of the players passively. Such a model was first introduced by Galil, Haber, and Yung [GHY87].[9] For model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}$, Chaum gave an MPC protocol that is unconditionally secure for $2t_a + t_c < n$ and $2t_c < n$ where t_c denotes the maximal number of corrupted players that collude, possibly including actively corrupted ones [Cha89]. This result is also strictly implied by the protocols in [Bea89, RB89, CDD+99]. In [FHM98], besides active and passive corruption, the additional fail corruption of t_f players is considered. For the model with secure channels it is shown that MPC is achievable if and only if $3t_a + 2t_p + t_f < n$. For the model with secure channels and authenticated broadcast channels it is shown that MPC is achievable if and only if $2t_a + 2t_p + t_f < n$.

Ito, Saito, and Nishizeki [ISN87] introduced the notion of general access structures for secret sharing. A general access structure is a subset of the power set of the player set. Thereby the idea is that exactly the player sets contained in the access structure shall be able to reconstruct a shared secret. Dually, a general adversary structure is also a subset of the power set of the player set — but meaning that the adversary may select exactly one of the elements of the structure and corrupt the corresponding players. The threshold case of a general adversary structure then simply forms the special case where the adversary structure contains exactly all player subsets of cardinality up to t. An adversary that is restricted to a general such access structure is called *general adversary*. Hirt and Maurer [HM00] showed that, in the passive model with pairwise secure channels, MPC unconditionally secure against a general adversary is achievable if and only if the union of no two elements of the adversary

[9] Note that, unlike the work referred to below, they distinguish between compromisers (passively corrupted players, i.e., read-only corruption) and violators (actively corrupted players but with no read access by the adversary, i.e., write-only corruption).

structure is equal to the player set. For the active model with pairwise secure channels, they showed that MPC unconditionally secure against a general adversary is achievable if and only if the union of no three elements of the adversary structure is equal to the player set. A natural, modular construction for such general MPC that can be based on any linear scheme for secret sharing is given in [CDM00] by Cramer, Damgård, and Maurer. In [FHM99], a hybrid model with a general mixed adversary is considered.

Chapter 4

Protocols and Impossibility Proofs

This chapter elaborates on some fundamental protocols and impossibility results for Byzantine agreement. Throughout, the network is assumed to be synchronous, and the adversary is assumed to be active. Section 4.1 treats the case where no data is consistently shared among the players (except for the common knowledge of the player set, the protocol, and the network topology — see Section 2.3.4), i.e., plain models $\mathcal{M}_{\mathrm{aut}}$ and $\mathcal{M}_{\mathrm{sec}}$. Section 4.2 treats the more powerful case where some data is consistently shared among the players in the form of a consistent PKI (models $\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}$ and $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}$). In Section 4.2.2, a hybrid model is defined where the players share a consistent PKI but Byzantine agreement is also required to be achieved in some cases where the adversary is able to forge signatures: If the adversary cannot forge signatures then Byzantine agreement is required to be achieved with respect to some large threshold but even if the adversary can forge signatures then Byzantine agreement is still required to be achieved with respect to some small threshold. In other words, a model is treated where protocols do not exclusively rely on the consistency of the given PKI and the security of the given signature scheme. Section 4.3 describes how the achievability of temporary broadcast among a set of players can be exploited in order to set up consistently shared data in the form of a consistent PKI — the converse problem of the one in Section 4.2 where consistently shared data is exploited in order to achieve broadcast.

4.1 Broadcast without consistently shared data

This section presents a selection of deterministic and probabilistic protocols for broadcast secure against $t < n/3$ actively corrupted players for models wherein no consistently shared data exists among the players. The deterministic protocols all provide perfect security; the probabilistic ones provide unconditional security with an error probability that is exponentially small in the security parameter κ. Finally, it is proven that Byzantine agreement is not possible if $n \geq 3t$ when no data is consistently shared among the players. These results are subsumed by the following theorem.

Theorem 4.1 (Tight Bounds for Broadcast [LSP82, DS82]). *In standard models $\mathcal{M}_{\mathrm{aut}}[\mathrm{u}]$ and $\mathcal{M}_{\mathrm{sec}}[\mathrm{c}]$ when no data is consistently shared among the players, (efficient) broadcast or consensus are possible if and only if $n > 3t$.*

Proof. The theorem follows from Theorem 4.8 (achievability of broadcast) and Theorem 4.16 (impossibility of broadcast), and Proposition 4.11 (equivalence of broadcast and consensus). □

Note that the achievability of Byzantine agreement in model $\mathcal{M}_{\mathrm{aut}}[\mathrm{u}]$ implies its achievability also in the models $\mathcal{M}_{\mathrm{aut}}[\mathrm{c}]$ (weaker security), $\mathcal{M}_{\mathrm{sec}}[\mathrm{u}]$ (stronger communication model), and $\mathcal{M}_{\mathrm{sec}}[\mathrm{c}]$. On the other hand, the impossibility of Byzantine agreement in model $\mathcal{M}_{\mathrm{sec}}[\mathrm{c}]$ implies its impossibility also in the models $\mathcal{M}_{\mathrm{aut}}[\mathrm{c}]$ (weaker communication model), $\mathcal{M}_{\mathrm{sec}}[\mathrm{u}]$ (stronger security), and $\mathcal{M}_{\mathrm{aut}}[\mathrm{u}]$.

4.1.1 Deterministic broadcast protocols

Two basic deterministic broadcast protocols are presented, Exponential Information-Gathering (EIG) [LSP82, BDDS92] and Phase-King [BG89b, BGP89]. The model is $\mathcal{M}_{\mathrm{aut}}$ (pairwise authenticated channels). Both protocols provide perfect security against an active adversary who corrupts up to $t < n/3$ players.

4.1.1.1 EIG protocol

Information gathering (IG) is the archetypal mechanism for broadcast (and Byzantine agreement in general). Its principle is based on a simple recursion of message distribution. However, when applied in its original way, this leads to inefficient protocols, namely exponential information gathering (EIG) [LSP82]. In [BDDS92] it is shown how EIG can be made

efficient by limiting the number of recursions to a constant but repeating the same procedure several times.

EIG among n players is implicitly based on sub-protocols that achieve broadcast among less than n players but satisfying validity and consistency with respect to different thresholds.

Definition 4.1 (Two-Threshold Broadcast). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ achieves* two-threshold broadcast *(*TTBC*, for short) with respect to P, p_s, $\mathcal{D}$, and thresholds t_v and t_c if it satisfies the following conditions:*

Validity: If at most $f \leq t_v$ players are corrupted and the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$.

Consistency: If at most $f \leq t_c$ players are corrupted then all correct players decide on the same output value. ◇

TTBC among a player set $S \subseteq P$ ($n = |S|$) with sender p_s and thresholds t_v and t_c such that $t_c \leq t_v$ recursively works as follows. First, the sender p_s distributes his input value x_s to all players in S. Second, each player $p_i \in S \setminus \{p_s\}$ redistributes the value he received from p_s with an instance of TTBC among the $n' = n - 1$ players in $S \setminus \{p_s\}$ with thresholds $t'_v = t_v$ and $t'_c = t_c - 1$. The EIG protocol for broadcast then simply consists of an invocation of TTBC among P with thresholds $t_v = t_c = t$.

Protocol 4.1 $\texttt{TTBC}\,(S, p_s, x_s, t_v, t_c)$

1. `if` $i = s$ `then` $\texttt{SendToAll}(x_s)$; $y_i := x_s$; return y_i `fi`; $\texttt{Receive}(x_i^s)$;
2. `if` $t_c = 0$ `then` $y_i := x_i^s$ `else`
3. $\quad \forall p_j \in S \setminus \{p_s\} : x_i^j := \texttt{TTBC}\,(S \setminus \{p_s\}, p_j, x_j^s, t_v, t_c - 1)$;
4. $\quad T_i^0 := \{j \in \{1, \ldots, n\} \setminus \{s\} \mid x_i^j = 0\}$;
   $\quad T_i^1 := \{j \in \{1, \ldots, n\} \setminus \{s\} \mid x_i^j = 1\}$;
5. $\quad$ `if` $|T_i^0| \geq n - t_v - 1$ `then` $y_i := 0$ `else` $y_i := 1$ `fi`;
6. `fi`; return y_i

Lemma 4.2. *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.1 achieves perfectly secure TTBC if $2t_v + t_c < n$ and $t_c \leq t_v$.*

Proof. The proof proceeds by backward induction on t_c. The induction base is given by $t_c = 0$, in which case the Protocol 4.1 trivially achieves TTBC.

Let now n, t_v and $t_c > 0$ be given such that $2t_v + t_c < n$ and ($t_c \leq t_v$), and suppose that Protocol 4.1 achieves TTBC for $n' = n - 1$, $t'_v = t_v$, and $t'_c = t_c - 1$.

We first show that validity is satisfied with respect to threshold t_v. Assume that $f \leq t_v$ players are corrupted and that the sender p_s is correct with input x_s. Then all correct players p_i receive $x_i^s = x_s$ and redistribute this value with TTBC among $n' = n - 1$ players. Since, by assumption, all executions of TTBC in step 3 satisfy the validity property for $t'_v = t_v$, every correct player p_i hence receives value $x_i^j = x_s$ from at least $n - t_v - 1$ different players $p_j \in S \backslash \{p_s\}$ and $x_i^j = 1 - x_s$ by at most $t_v \geq t_c$ different players. Since $2t_v + t_c < n$ and $t_c > 0$ it holds that $t_v < n - t_v - 1$, and hence that every correct player decides on $y_i = x_i^s = x_s$ according to step 5.

We now show that consistency is satisfied with respect to threshold t_c. Assume that $f \leq t_c$ players are corrupted. If the sender p_s is correct then consistency follows from validity for t_v since $t_c \leq t_v$. Thus, assume that the sender is corrupted (and hence $t_c \geq f > 0$). Then, among the remaining $n' = n - 1$ players involved in the executions of TTBC in step 3, there are at most $t' = f - 1$ corrupted players left. By assumption, these executions among $n' = n - 1$ players achieve consistency for $t'_c = t_c - 1$ and hence any two correct players p_i and p_j compute exactly the same values $x_i^k = x_j^k$ for all $k \neq s$, and the correct players all compute y_i in the same way. □

Protocol 4.2 `Broadcast` (P, p_s, x_s)

1. $y_i := TTBC\,(P, p_s, x_s, t, t)$;
2. return y_i

Theorem 4.3. *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.2 achieves broadcast perfectly secure against $t < n/3$ corrupted players. Its round complexity is $\mathcal{R} = t + 1$ and its bit complexity is exponential in n.*

Proof. That Protocol 4.2 achieves broadcast follows from Definition 4.1, Lemma 4.2, and the fact that $t_c = t_v = t < n/3$. Round and bit complexities can be easily verified by code inspection. □

4.1.1.2 Phase-king protocol

The phase-king paradigm was proposed in [BG89b, BGP89]. We reformulate it in a modular way as a stepwise reduction from broadcast to weak consensus. The reduction proceeds in three steps, each reducing a more

involved problem to a simpler one: graded consensus is reduced to weak consensus, king consensus to graded consensus, and, finally, broadcast to king consensus.

Weak consensus can be achieved by simply having the players tell each other their initial value. Each player will then tally the votes for 0 and 1. Every player p_i then computes bit $y_i \in \{0,1\}$ if he received y_i at least $n-t$ times — otherwise he computes $y_i = \perp$.

Protocol 4.3 `WeakConsensus` (P, x_i)

1. `SendToAll`(x_i); `Receive`$(x_i^1, \ldots, x_i^n)$;
2. $S_i^0 := \{j \in \{1,\ldots,n\} \mid x_i^j = 0\}$; $S_i^1 := \{j \in \{1,\ldots,n\} \mid x_i^j = 1\}$;
3. `if` $|S_i^0| > |S_i^1|$ `then` $y_i := 0$ `else` $y_i := 1$ `fi`;
4. `if` $|S_i^{y_i}| < n - t$ `then` $y_i := \perp$ `fi`;
5. return y_i

Lemma 4.4 (Weak Consensus). *In standard model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.3 achieves weak consensus among n players secure against $t < n/3$ corrupted players.*

Proof.
Validity: If all correct players hold the same value v at the beginning of the protocol then there are at least $n-t$ players (all correct ones) that consistently distribute v during step 1 of the protocol, and hence for every correct player p_i, it holds that $|S_i^v| \geq n-t$ and $|S_i^{1-v}| \leq t < n-t$, and hence $y_i = v$ at the end of the protocol.

Consistency: Let p_i and p_j be two correct players and suppose that p_i decides on $y_i \in \{0,1\}$, i.e., $|S_i^{y_i}| \geq n-t$. Since at most t players (all corrupted ones) distribute inconsistent information this implies that $|S_j^{y_i}| \geq (n-t)-t = n-2t > t$ and thus $|S_j^{1-y_i}| < n-t$, and hence, $y_j \in \{y_i, \perp\}$. □

Protocol 4.3 can now be transformed into a protocol for graded consensus, as described by Protocol 4.4. The players first perform an invocation of weak consensus on their inputs followed by an "echo round" where the players tell each other their particular outcome of weak consensus. Finally, every player p_i tallies the votes for 0 and 1 in order to determine his output value y_i and grade g_i.

Protocol 4.4 `GradedConsensus` (P, x_i)

1. $z_i :=$ `WeakConsensus` (P, x_i);
2. `SendToAll`(z_i); `Receive`$(z_i^1, \ldots, z_i^n)$;
3. $S_i^0 := \{j \in \{1,\ldots,n\} \mid z_i^j = 0\}$; $S_i^1 := \{j \in \{1,\ldots,n\} \mid z_i^j = 1\}$;
4. `if` $|S_i^0| > |S_i^1|$ `then` $y_i := 0$ `else` $y_i := 1$ `fi`;
5. `if` $|S_i^{y_i}| \geq n - t$ `then` $g_i := 1$ `else` $g_i := 0$ `fi`;
6. return (y_i, g_i)

Lemma 4.5 (Graded Consensus). *In standard model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.4 achieves graded consensus among n players secure against $t < n/3$ corrupted players.*

Proof.
Validity: If all correct players hold the same value v at the beginning of the protocol then, by the validity property of weak consensus, all correct players p_i get $z_i = v$ which they send during step 2. Hence, during step 2, all correct players receive v at least $n - t$ times, and hence $|S_i^v| \geq n - t$ and $|S_i^{1-v}| \leq t$, and hence $y_i = v$ and $g_i = 1$.

Consistency: Let p_i and p_j be two correct players and suppose that p_i decides on $y_i \in \{0, 1\}$ and $g_i = 1$, i.e., $|S_i^{y_i}| \geq n - t$. Hence, at least $(n - t) - t > t$ correct players sent value y_i during step 2. Together with the consistency property of weak consensus (step 1) this implies that no correct players sent value $1 - y_i$ during step 2, i.e., every correct player received value y_i strictly more than t times but received value $1 - y_i$ at most t times, and hence decides on $y_j = y_i$ by majority. □

As will follow from the rest of the construction, the achievability of graded consensus implies broadcast independently of the number of corrupted players, i.e., for $t < n$. Since this is of fundamental importance for the rest of this text, from now on, we simply assume a protocol for graded consensus to be given secure for some arbitrary (fixed) threshold t, $0 \leq t < n$, and show how this protocol can be turned into a protocol for broadcast with respect to the same threshold t. In other words, broadcast with respect to any threshold t is reduced to graded consensus with respect to the same threshold.

We now proceed by transforming Protocol 4.4 into king consensus, as given by Protocol 4.5. The players first perform an invocation of graded consensus on their inputs followed by an "echo round" where player p_k, the king, sends to all players his particular outcome of graded consensus. This has the effect that a correct king can enforce agreement at the end of the protocol whereas a corrupted king has no effect whenever the correct players already start in the state of agreement on some value.

Protocol 4.5 `KingConsensus`(P, p_k, x_i)

1. $(y_i, g_i) :=$ `GradedConsensus` (P, x_i);
2. `if` $i = k$ `then` `SendToAll`(y_i) `fi`; `Receive`(y_i^k);
3. `if` $g_i = 0$ `then` $y_i := y_i^k$ `fi`;
4. return y_i

Lemma 4.6 (King Consensus). *Let* `GradedConsensus` *be a protocol that achieves graded consensus among n players secure against any fixed number t ($0 \leq t < n$) of corrupted players. Then, in model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.5 achieves king consensus among n players secure against t corrupted players.*

Proof.
Validity: If all correct players hold the same value v at the beginning of the protocol then, by the validity property of graded consensus, all correct players p_i decide on $y_i = v$ and compute $g_i = 1$, and hence they all ignore the king p_k and stick to $y_i = v$.

Consistency: Let p_k be correct and assume that all correct players p_i adopt the king's value y_i^k in step 3 because they all compute grade $g_i = 0$ — then all decide on the same value $y_i = y_i^k$ since p_k is correct and distributes the same value to all players during step 2.

On the other hand, let p_k be correct and assume that some correct player p_i ignores the king because he computes grade $g_i = 1$. Then, by the consistency property of graded consensus, every other correct player p_j received $y_j = y_i$ in step 1. Hence, p_k distributes $y_k = y_i$ during step 2, and all correct players decide on y_i independently of whether or not they adopt p_k's value. □

Finally, king consensus secure against t corrupted players can be turned into broadcast with respect to t. This is achieved by having the sender distribute his input value, followed by t invocations of king consensus — with the restriction that the sender and the king players of each phase must all be pairwise distinct. Note that the sender can be seen as an initial king player. For convenience, broadcast Protocol 4.6 is stated with respect to sender p_1. For easier references in the rest of this text, Protocol 4.5 for king consensus (marked KC) is directly in-lined into the following protocol.

Protocol 4.6 `Broadcast` (P, p_1, x_1)

```
1.  if i = 1 then SendToAll(x_1) fi;     Receive(y_i);
2.  for k = 2 to t + 1 do
3.    (y_i, g_i) := GradedConsensus (P, y_i);          } KC
4.    if i = k then SendToAll(y_i) fi;  Receive(y_i^k); }
5.    if g_i = 0 then y_i := y_i^k fi;
6.  od; return y_i
```

Lemma 4.7. *Let* `GradedConsensus` *be a protocol that achieves graded consensus among n players secure against any fixed number t ($0 \leq t < n$) of corrupted*

players. Then, in model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.6 achieves broadcast among n players secure against t corrupted players.

Let $\mathcal{R}_{\mathrm{gc}}$ be the round complexity and $\mathcal{B}_{\mathrm{gc}}$ be the bit complexity of Protocol `GradedConsensus`*. The resulting broadcast protocol requires a round complexity of $\mathcal{R} = t \cdot \mathcal{R}_{\mathrm{gc}} + t + 1$ and a bit complexity of $\mathcal{B} = O(n^2) + t \cdot \mathcal{B}_{\mathrm{gc}}$.*

Proof.
Validity: If sender p_1 is correct then all correct players p_i enter step 2 with the same value $y_i = x_1$, and validity follows from the validity property of king consensus.

Consistency: For the case that sender p_1 is correct, consistency is implied by the validity property of broadcast. If p_1 is corrupted then, among the t phases of king consensus at least one involves a correct king. At the end of this phase, all players agree on a value by the consistency property of king consensus. Finally, by the validity property of king consensus, all players will stay with this value, and consistency follows.

Complexity: The round and bit complexities of the protocol can be easily verified by code inspection. □

Instantiating `GradedConsensus` with Protocol 4.4, we immediately get

Theorem 4.8 (Broadcast [BGP89]). *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.6 achieves broadcast among n players perfectly secure against $t < n/3$ corrupted players. Its round complexity is $\mathcal{R} = 3t + 1$ and its bit complexity is $\mathcal{B} = O(tn^2)$.*

4.1.2 Probabilistic broadcast protocols

In this section we present two basic probabilistic broadcast protocols for model $\mathcal{M}_{\mathrm{sec}}$ (pairwise secure channels). Both protocols provide unconditional security against an active adversary who corrupts up to $t < n/3$ players. Note that probabilistic protocols typically require secure channels (instead of only authenticated ones) in order to "hide" the randomness from the adversary. As an exception, the protocols in [Ben83, Bra87b] use randomness only for local decisions and hence only require authenticated channels.

Randomization allows to beat the lower bound of $t + 1$ rounds for deterministic protocols. It is possible to achieve broadcast in a (small) constant expected number of rounds with error probability 0 but requiring arbitrary many rounds in the worst case, i.e., protocols of type "Las Vegas". However, the probability of non-termination within r rounds can be made exponentially small in r which, loosely speaking, in fact ensures

fast termination. On the other hand, this fast decrease allows for protocols of type "Monte Carlo" by running the protocol for a fixed number of rounds r guaranteeing termination within r rounds but accepting an error probability exponentially small in r.

This section is restricted to protocols of type "Monte Carlo" since these protocols have the important advantage to guarantee that all players terminate the protocol within the same communication round, i.e., they achieve SBA (cf. Section 3.1.4, "Simultaneous versus eventual Byzantine agreement"), implying that, after any broadcast invocation within a larger protocol, all players would simultaneously continue the rest of the protocol during the same round.

4.1.2.1 Collective coin-tossing

Probabilistic protocols typically make use of a binary *shared coin* protocol, i.e., a protocol that allows the good players to produce, by exchanging messages, a reasonably unpredictable coin toss. We use the following definition in [FM97]:

Definition 4.2. *Let $\mathcal{C} = \{0,1\}$ be the set of possible outcomes, and Ψ a protocol in which each player p_i computes an output value $c_i \in \mathcal{C}$. Then Ψ is a* q-fair shared coin *protocol if and only if, $\forall x \in \mathcal{C}$, in a random execution of C with at most t players being corrupted*

$$\Pr(\forall \text{ correct players } p_i : \ c_i = x) \geq q.$$

◇

In other words, q-fairness guarantees that, for each bit $x \in \mathcal{C}$, the probability that all correct players will agree on x at the end of the protocol is at least q. It is important to note that this definition does not guarantee that the correct players always agree on the outcome.

As for unconditional security, shared coin protocols have for example been developed in [FM97, MR91, BGR96]. The protocol in [FM97] works in standard model $\mathcal{M}_{\text{sec}}$ and tolerates $t < n/3$. Their protocol terminates in a fixed constant number of rounds providing q-fairness for $q > 0.35$.

4.1.2.2 Feldman-Micali protocol

The archetypal form of randomized Byzantine agreement, due to Rabin [Rab83a], is to combine some alleviated form of consensus like weak consensus with a shared coin in the following way. First the sender distributes his input value. Then, similar to phase-king Protocol 4.6, several

phases are run wherein a protocol for alleviated consensus is followed by a protocol implementing a shared coin. In each phase, a player who "dislikes" the outcome of alleviated consensus simply adopts the value of the shared coin. Thus the coin simply takes the role of the king player of Protocol 4.6.[10]

For the "Las Vegas" variant of randomized Byzantine agreement graded consensus (in its original form with ternary grade values $g_i \in \{0,1,2\}$ [FM97]) can be used as the alleviated form of consensus mentioned above. In each phase, a player p_i who gets grade $g_i = 0$ in graded consensus adopts the coin; if he gets $g_i = 1$ then he stays with his value; and if he gets $g_i = 2$ then he terminates the protocol. This guarantees agreement after the first phase wherein all correct players agree on the outcome of the coin c and some correct player computed $x_i = c$ as the output value of graded consensus; and termination after one additional round. Using the shared-coin protocol in [FM97], this protocol is efficient and terminates in a constant expected number of rounds.

For the "Monte Carlo" variant, Protocol 4.7 below, weak consensus can be used as the alleviated form of consensus. In each phase, every player p_i who outputs $x_i = \perp$ in weak consensus adopts the coin. This guarantees agreement after the first phase wherein all correct players agree on the outcome of the coin, and either all correct players computed $x_i = \perp$ or the outcome of the coin is equal to the value x_i of any correct player p_i.

Let $\texttt{CoinFlip}(\{0,1\})$ denote the binary shared-coin protocol in [FM97] and let κ be a security parameter.

Protocol 4.7 $\texttt{Broadcast}(P, p_s, x_s)$

1. `if` $i = s$ `then` $\texttt{SendToAll}(x_s)$ `fi;` $\texttt{Receive}(x_i)$;
2. `for` $k = 1$ `to` κ `do`
3. $\quad x_i := \texttt{WeakConsensus}\,(P, x_i)$
4. $\quad c_i := \texttt{CoinFlip}(P, \{0,1\})$;
5. $\quad$ `if` $x_i = \perp$ `then` $x_i := c_i$ `fi`;
6. `od;` $y_i := x_i$;
7. return y_i

Theorem 4.9 (Broadcast [FM97]). *In model $\mathcal{M}_{\text{sec}}$, Protocol 4.7 achieves unconditionally secure broadcast among n players secure against $t < n/3$ corrupted players in $O(\kappa)$ rounds with an error probability of at most $\left(\frac{2}{3}\right)^{\kappa}$.*

[10] In fact, in contrast to the way presented here, phase-king is actually based on this paradigm whereby the coin is replaced by the vote of the king player.

Proof.
Validity: If the sender p_s is correct then all correct players start the first phase with the same value x_s and hence validity follows by the validity property of weak consensus.

Consistency: Consider any phase where the coin is common, i.e., that for all correct players p_i, p_j, it holds that $c_i = c_j$. If every correct player computed $x_i = \perp$ in step 3 then all correct players adopt the coin, and consistency follows. If any correct player computed $x_i \neq \perp$ in step 3 and $c_i = x_i$ then, by the consistency property of weak consensus, every correct player p_j with $x_j \neq x_i$ adopts the coin, and consistency follows.

The shared coin guarantees that, with a probability of at least $\frac{1}{3}$, all players compute the same coin, and additionally, that $c_i = x_i$ if some correct player p_i computed $x_i \neq \perp$ in step 3. Thus, the probability that consistency never occurs during all κ phases is at most $\left(\frac{2}{3}\right)^{\kappa}$. □

Furthermore, the applied shared-coin protocol is efficient and hence also the resulting broadcast protocol is. In particular, an analysis of the protocol in [FM97] yields a bit complexity of $\mathcal{B} = O(n^7\kappa)$ (assuming that $t = \Omega(n)$).

4.1.2.3 Probabilistic phase-king protocol

The protocol in [FM97] has the disadvantage that running ℓ independent invocations of it in parallel requires $\kappa + \Omega(\log \ell)$ phases per protocol in order to achieve an error probability that is still negligible in κ, as was observed in [BE03]. There an efficient protocol was proposed that overcomes this problem, i.e., an $O(\kappa)$-round protocol for any (polynomial) number of parallel broadcasts such that the overall error probability is negligible in κ.

The probabilistic-phase-king paradigm allows for the parallelization of broadcast in a simpler way than in [BE03]. Essentially, this protocol is obtained from Protocol 4.6 by electing the "king" at random during every phase. This can be achieved by the leader-election protocol in [FM97] that terminates in a (fixed) constant number of rounds. It guarantees that, with probability $\frac{1}{3}$, all correct players choose the same correct player as "king" by a random selection process.

Let `LeadElect`(P) denote the leader-election protocol of [FM97]. Note that during each phase of the final broadcast protocol, all players must act as a potential "king" already *before* the election process since, otherwise, an adaptive adversary could successively corrupt all "phase kings" for at least $t - 1$ rounds.

Protocol 4.8 `Broadcast` (P, p_s, x_s)

1. `if` $i = s$ `then` `SendToAll`(x_s) `fi`; `Receive`(x_i);
2. `for` $k = 1$ `to` κ `do`
3. $(x_i, g_i) :=$ `GradedConsensus` (P, x_i);
4. `SendToAll`(x_i); `Receive`$(x_i^1, \ldots, x_i^n)$;
5. $k :=$ `LeadElect`(P);
6. `if` $g_i = 0$ `then` $x_i := x_i^k$ `fi`;
7. `od`; $y_i := x_i$;
8. return y_i

Theorem 4.10 (Broadcast [FG02]). *In model $\mathcal{M}_{\mathrm{sec}}$, Protocol 4.8 achieves broadcast among n players secure against $t < n/3$ corrupted players in $O(\kappa)$ rounds with an error probability of at most $\left(\frac{2}{3}\right)^{\kappa}$.*

Proof.
Validity: If the sender p_s is correct then all correct players start the first phase with the same value x_s. By the validity property of graded consensus, at the end of each phase, every correct player p_i computes $g_i = 1$ and $x_i = x_s$, and validity follows.

Consistency: Note that the steps 3 to 6 simply constitute a probabilistic version of king consensus. Protocol `LeadElect`(P) guarantees that, with probability $\frac{1}{3}$, all correct players agree on the same correct king p_k. Hence, the probability that they never agree on a common correct king is at most $\left(\frac{2}{3}\right)^{\kappa}$, and by the consistency property of king consensus, consistency is satisfied with an error probability of at most $\left(\frac{2}{3}\right)^{\kappa}$. □

Since the applied leader-election protocol is efficient [FM97], also the resulting broadcast protocol is. Furthermore, with respect to the same error probability, an arbitrary number of broadcasts can now be run in parallel by merging their leader-election invocations into one single instance in order to elect one central king to deliver his default values with respect to all parallel protocols.

4.1.3 Generic reductions

This section treats the reducibility between broadcast, consensus, and their weakened variants. The question is about which primitives can be built from others with respect to which threshold. All reductions are given with respect to unconditional security. By simple arguments, it can be shown that the pairs broadcast and consensus, graded broadcast and graded consensus, and, weak broadcast and weak consensus, can all be

used to simulate each other if $t < n/2$. As a maybe more surprising fact, it can also be shown that weak broadcast implies broadcast if $t < n/2$, and hence, that all mentioned problems are equivalent with respect to $t < n/2$.

At a first sight it might seem not to make sense to make these considerations with respect to any threshold $t \geq n/3$ since, for this case, broadcast is not achievable anyway. Nevertheless, these reductions will be of some importance in the sequel. More general reductions along these lines will be given in Chapter 5.

4.1.3.1 Broadcast and consensus

Proposition 4.11. *In models $\mathcal{M}_{\mathrm{aut}}$ and $\mathcal{M}_{\mathrm{sec}}$, if $t < n/2$ then (efficient) broadcast is achievable if and only if (efficient) consensus is achievable.*

Proof.
"$\Rightarrow$": Suppose that broadcast is achievable. Then consensus can be simulated by having every player broadcast his input value and having every player decide on the majority of received values. Since all values are distributed by broadcast and a majority of the players is correct, this protocol achieves consensus.

"$\Leftarrow$": Suppose that consensus is achievable. Then broadcast can be simulated by having the sender distribute (multi-send) his input value to all players and having all players run a consensus protocol on the values received from the sender. Since a majority of the players is correct, this protocol achieves broadcast. □

The implicit constructions in the proof of Proposition 4.11 immediately show how the previously mentioned broadcast protocols can be turned into protocols for consensus in a generic way. However, specifically modifying such a protocol directly usually yields a more efficient solution. For example, Protocol 4.6 can be directly turned into a consensus protocol by removing step 1 but having every player start with his own input, $y_i := x_i$, and appending one more round of king consensus — saving a factor of n in bit complexity compared to the generic reduction involving n parallel broadcast protocols.

Analogously to Proposition 4.11 we get the following

Proposition 4.12. *In models $\mathcal{M}_{\mathrm{aut}}$ or $\mathcal{M}_{\mathrm{sec}}$, if $t < n/2$ then (efficient) weak broadcast is achievable if and only if (efficient) weak consensus is achievable, and*

(efficient) graded broadcast is achievable if and only if (efficient) graded consensus is achievable.

Proof.
Weak case.
"$\Rightarrow$": Suppose that weak broadcast is achievable. Then weak consensus can be simulated by having each player distribute his input by weak broadcast, followed by having him decide on value $v \in \{0,1\}$ if he received v at least $n-t$ times and else on $\perp$.

"$\Leftarrow$": Suppose that weak consensus is achievable. Then weak broadcast can be simulated by having the sender multi-send his input to every player, followed by a protocol of weak consensus where everybody inputs the received value.

Graded case.
"$\Rightarrow$": Suppose that graded broadcast is achievable. Then graded consensus can be simulated by having each player distribute his input by graded broadcast. Each player now decides on value $y_i = v$ if he received v at least $n-t$ times, and on $y_i = 0$ otherwise. Finally he computes $g_i = 1$ if y_i was received at least $n-t$ times with grade 1.

"$\Leftarrow$": Suppose that graded consensus is achievable. Then graded broadcast can be simulated by having the sender multi-send his input to every player, followed by a protocol of graded consensus where everybody inputs the received value. □

4.1.3.2 Weak broadcast and broadcast

We now show how to reduce broadcast (and consensus) to weak broadcast for any $t < n/2$. This will be of use for the constructions in Section 4.2.2 and Chapter 5. That broadcast can be generically reduced to weak broadcast for any $t < n/3$ was already mentioned in [Dol82].

Recall Lemma 4.7. It states that, with respect to any threshold t, graded consensus implies broadcast with the same resilience. Hence, it is sufficient to show that weak broadcast implies graded consensus if $t < n/2$. This is demonstrated by Protocol 4.9 which basically consists of two consecutive rounds wherein each player weak-broadcasts a value. Note that, in step 4 of the protocol, the domain of weak broadcast is ternary, namely $\{0,1,\perp\}$. Following the restriction to focus on protocols with binary domains we can simply interpret such a protocol as being simulated by two parallel invocations of binary weak broadcast.

Protocol 4.9 `GradedConsensus`(P, x_i)

1. $\forall j \in \{1, \ldots, n\} : x_i^j :=$ `WeakBroadcast` (P, p_j, x_j);
2. $S_i^0 := \{j \in \{1, \ldots, n\} | x_i^j = 0\}$; $S_i^1 := \{j \in \{1, \ldots, n\} | x_i^j = 1\}$;
3. `if` $|S_i^{x_i}| \geq n - t$ `then` $z_i := x_i$ `else` $z_i := \perp$ `fi`;
4. $\forall j \in \{1, \ldots, n\} : z_i^j :=$ `WeakBroadcast` (P, p_j, z_j);
5. $T_i^0 := \{j \in \{1, \ldots, n\} | z_i^j = 0\}$; $T_i^1 := \{j \in \{1, \ldots, n\} | z_i^j = 1\}$;
6. `if` $|T_i^0| > |T_i^1|$ `then` $y_i := 0$ `else` $y_i := 1$ `fi`;
7. `if` $|T_i^{y_i}| \geq n - t$ `then` $g_i := 1$ `else` $g_i = 0$ `fi`;
8. return (y_i, g_i)

Lemma 4.13 (Graded Consensus). *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 4.9 achieves graded consensus secure against $t < n/2$ corrupted players.*

Let $\mathcal{R}_{\mathrm{wbc}}$ be the round complexity and $\mathcal{B}_{\mathrm{wbc}}$ be the bit complexity of the given primitive for weak broadcast. Then Protocol 4.9 requires a round complexity of $\mathcal{R} = 2\mathcal{R}_{\mathrm{wbc}}$ and a bit complexity of $\mathcal{B} = 3n \cdot \mathcal{B}_{\mathrm{wbc}}$.

Proof.
Validity: If all correct players hold the same value v at the beginning of the protocol then, by the validity property of weak broadcast in step 1, every correct player p_i gets value v at least $n - t$ times but value $1 - v$ at most $t < n - t$ times, and hence computes $z_i = v$. By the same argument with respect to weak broadcast in step 4, every correct player finally computes $y_i = v$ and $g_i = 1$, and validity follows.

Consistency: Note that steps 1 to 3 constitute an instance of weak consensus (cf. proof of Proposition 4.12). In particular, all correct players p_ℓ that do not set $z_\ell := \perp$ in step 3 compute the same value for $z_\ell \in \{0, 1\}$.

Now, let p_i and p_j be two correct players and suppose that p_i decides on $y_i = v \in \{0, 1\}$ and $g_i = 1$, i.e., $|T_i^v| \geq n - t$. Then, at least one correct player p_j must have sent $z_j = v$ in step 4 — but no correct player p_ℓ can have sent $z_\ell = 1 - v$ as follows from the above remark.

Let f_v be the number of corrupted players whom p_i received value v from in step 4. Then, for every correct player p_j, it holds that $|T_j^v| \geq n - t - f_v$. Furthermore, by the validity property of weak broadcast, p_j can have received value $1 - v$ from at most $t - f_v$ (corrupted) players, and $|T_j^{1-v}| \leq t - f_v$. Since $n > 2t$, we get $|T_j^v| \geq n - t - f_v > t - f_v \geq |T_j^{1-v}|$, and p_j decides on $y_j = v = y_i$ in step 6. Hence, consistency follows.

Complexity: The stated complexities can be easily verified by code inspection (regarding that the second invocations of weak broadcast are ternary and must hence be simulated by two parallel invocations of binary weak broadcast). □

Theorem 4.14 (Weak Broadcast Implies Broadcast). *In model $\mathcal{M}_{\mathrm{aut}}$, if $t < n/2$ then (efficient) weak broadcast implies (efficient) broadcast.*

Let $\mathcal{R}_{\mathrm{wbc}}$ be the round complexity and $\mathcal{B}_{\mathrm{wbc}}$ be the bit complexity of the given primitive for weak broadcast. Then there is a broadcast protocol that requires round complexity $\mathcal{R} = 2t \cdot \mathcal{R}_{\mathrm{wbc}} + t + 1$ and bit complexity $\mathcal{B} = O(n^2) + 3nt \cdot \mathcal{B}_{\mathrm{wbc}}$.

Proof. The theorem follows from Lemma 4.7 (broadcast from graded consensus for any t) and Lemma 4.13 (graded consensus from weak broadcast for $t < n/2$). □

4.1.4 Impossibility result

In this section, we prove that broadcast among $n \geq 3$ players is impossible if $t \geq n/3$ with respect to both unconditional and computational security. From Proposition 4.11 it then immediately follows that also consensus is impossible for this bound. As a warm-up, we first prove the impossibility of the special case $n = 3$ and $t \geq 1$, and then generalize it to arbitrary numbers $n \geq 3$ and $t \geq n/3$.

For model $\mathcal{M}_{\mathrm{aut}}$ with $n = 3$ and $t \geq 1$, Graham and Yao [GY89] proved a tight bound of $\frac{\sqrt{5}-1}{2}$ for the possible success probability of Byzantine agreement. In contrast, our impossibility proof is given along the lines of [FLM86] where it is shown that broadcast for $t \geq n/3$ is not achievable in the more powerful model $\mathcal{M}_{\mathrm{sec}}$, which immediately implies unachieveability in model $\mathcal{M}_{\mathrm{aut}}$ as well. We prove that every protocol can only succeed with a probability of at most $\frac{2}{3}$ in the unconditional case and with a probability of at most $\frac{5}{6}$ in the computational case. For model $\mathcal{M}_{\mathrm{aut}}[u]$ this is a slightly weaker bound than the one given in [GY89].

The idea of the proof is to assume that there exists a broadcast protocol for three players secure against one corrupted player which then can be used to build a different system with contradictory behavior, hence proving that such a protocol cannot exist.

Lemma 4.15. *In models $\mathcal{M}_{\mathrm{aut}}$ or $\mathcal{M}_{\mathrm{sec}}$, broadcast among $n = 3$ players $\{p_0, p_1, p_2\}$ is not achievable if $t \geq 1$. For every protocol there exists a value $x_0 \in \{0, 1\}$ such that, when the sender holds input x_0, the adversary can make the protocol fail with a probability of at least $\frac{1}{6}$.*

Proof. Suppose Ψ to be any broadcast protocol for the three players p_0, p_1, and p_2, with sender p_0. Let π_0, π_1, π_2 denote the players' corresponding processors with their local programs and, for each $k \in \{0, 1, 2\}$ let π_{k+3} be an identical copy of processor π_k.

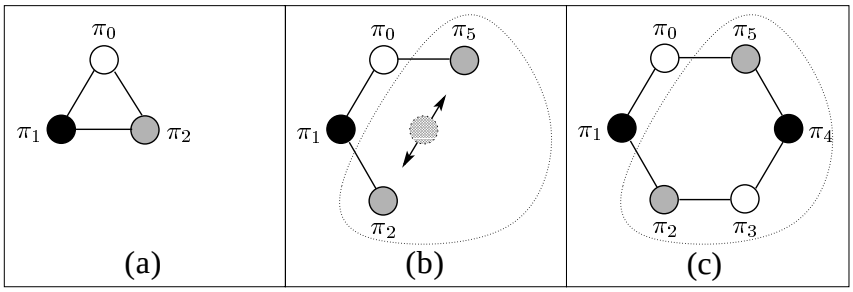

Figure 4.1: *Rearrangement of processors in proof of Lemma 4.15.*

Instead of connecting the original processors as required for broadcast, we build a network requiring all six processors (i.e., the original ones together with their copies) by arranging them in a circle, i.e., each processor π_k ($k \in \{0, \ldots, 5\}$) is (exactly) connected by one channel with $\pi_{(k-1) \bmod 6}$ and one with $\pi_{(k+1) \bmod 6}$.

We now prove that, in the new system and without the presence of an adversary, for every pair of adjacent processors π_k and $\pi_{(k+1) \bmod 6}$ their common view is indistinguishable from their view as two processors $\pi_{k \bmod 3}$ and $\pi_{(k+1) \bmod 3}$ in the original system with respect to an adversary who corrupts the remaining processor $\pi_{(k+2) \bmod 3}$ in an admissible way. Refer to Figure 4.1. The original system is depicted in Figure 4.1-(a). With respect to the processors π_0 and π_1, π_2 is "split" into two different copies, π_2 and π_5, where π_0 is connected to π_5 and π_1 is connected to π_2 (Figure 4.1-(b)). By assumption, when running this system, the processors π_0 and π_1 achieve broadcast independently from the behavior of the processors π_2 and π_5. Furthermore, by arranging the six processors in a circle as described above and shown in Figure 4.1-(c), this "splitting" is simultaneously achieved with respect to every pair π_k and $\pi_{(k+1) \bmod 6}$. Hence, for every such pair, their joint view is indistinguishable from their view (as processors $\pi_{k \bmod 3}$ and $\pi_{(k+1) \bmod 3}$) in the original system where the adversary corrupts $\pi_{(k+2) \bmod 3}$ by simply simulating all the remaining processors of the new system. For example, with respect to pair (π_0, π_1) the corresponding adversary strategy for the original system is to corrupt π_2, simulate the correct processors $\pi_2, \ldots, \pi_5$ in the new system, and make π_2 behave to π_1 like π_2 in the new system and to π_0 like π_5 in the new system — in other words, the adversary simulates the subsystem encircled by the dotted line in Figure 4.1-(c).

The new system involves two processors of the type corresponding to the sender, namely, π_0 and π_3, and these are the only processors that enter

an input. Let now π_0 and π_3 be initialized with different inputs, i.e., let's assume that π_0 has input $x_0 \in \{0,1\}$ and that π_3 has input $x_3 = 1 - x_0$.

We now show that there are at least two pairs of adjacent processors in the new system, i.e., one third among all six such pairs, for which the broadcast conditions are not satisfied despite being completely consistent with two correct processors in the original system. First, suppose that consistency holds with respect to every pair on, wlog, the value x_0. Then the validity condition is violated with respect to both pairs involving processor π_3 since $x_3 \neq x_0$. Suppose, otherwise, that the consistency condition is violated with respect to at least one pair. Then there must be at least two such pairs because the processors are arranged in a circle.

Hence, for any possible run of the new system on inputs $x_0 = 0$ and $x_3 = 1$ it holds that, chosen a pair $(\pi_k, \pi_{(k+1) \bmod 6})$ of adjacent processors uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$. Otherwise, there would exist invocations of the new system where strictly less than two pairs fail. In particular, either

A) Chosen a pair from $\{(\pi_5, \pi_0), (\pi_0, \pi_1), (\pi_1, \pi_2)\}$ uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$; or

B) Chosen a pair from $\{(\pi_2, \pi_3), (\pi_3, \pi_4), (\pi_4, \pi_5)\}$ uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$.

If case (A) holds then, on sender input $x_0 = 0$, if the adversary selects a pair $(\pi_k, \pi_{(k+1) \bmod 6})$ of the new system at random and corrupts $\pi_{(k+2) \bmod 3}$ in the original system by simulating the processors in $\Sigma = \left\{\pi_{(k+\ell) \bmod 6} | \ell \in \{2, \ldots, 5\}\right\}$ then the protocol fails with a probability of at least $\frac{1}{6}$. If case (B) holds then the same holds for sender input $x_0 = 1$. Hence the lemma follows. □

The following theorem proceeds in the same way as Lemma 4.15. It is assumed that there exists a broadcast protocol for n players secure against $t \geq n/3$ corrupted players which then can be used to build a different system with contradictory behavior.

Theorem 4.16 (Impossibility [LSP82, KY84, FLM86]). *In models $\mathcal{M}_{\text{aut}}$ or $\mathcal{M}_{\text{sec}}$, broadcast among a set of $n \geq 3$ players $P = \{p_0, \ldots, p_{n-1}\}$ is not achievable if $n \leq 3t$. For every protocol there exists a value $x_0 \in \{0,1\}$ such that, when the sender holds input x_0, the adversary can make the protocol fail*

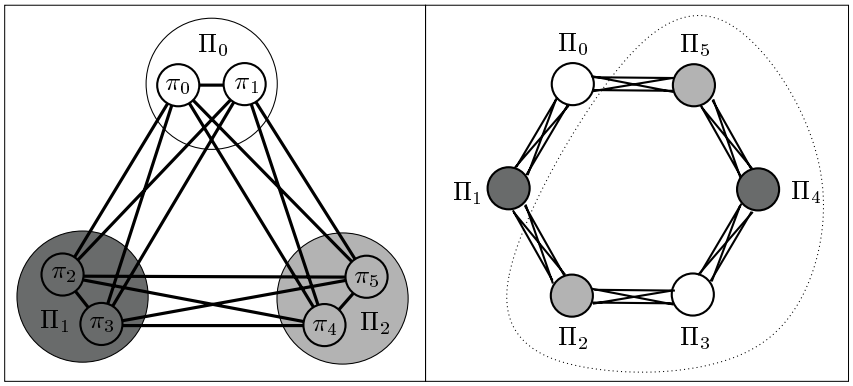

Figure 4.2: *Rearrangement of processors in proof of Theorem 4.16 for the special case $n = 6$ and $t = 2$.*

- *with a probability of at least $\frac{1}{6}$ if she is computationally bounded, and*
- *with a probability of at least $\frac{1}{3}$ if she is computationally unbounded.*

Proof. Assume Ψ to be a broadcast protocol for n players $p_0, \ldots, p_{n-1}$ with sender p_0 that tolerates $t \geq n/3$ corrupted players.

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs and let $\Pi_0 \dot{\cup} \Pi_1 \dot{\cup} \Pi_2 = \Pi$ be a partition of Π such that, for each set Π_k, it holds that $1 \leq |\Pi_k| \leq t$. Furthermore, for each $i \in \{0, \ldots, n-1\}$, let π_{i+n} be an identical copy of processor π_i. For every π_i ($0 \leq i \leq 2n-1$) let the *type* of processor π_i be defined as the number $i \bmod n$. Finally, for each $k \in \{0, 1, 2\}$, let $\Pi_{k+3} = \{\pi_{i+n} \mid \pi_i \in \Pi_k\}$ form identical copies of the sets Π_k.

Instead of connecting the original processors as required for broadcast, we build a network involving all $2n$ processors (i.e., the original ones and their copies) by arranging the six processor sets Π_k in a circle — as done with the single processors in the proof of Lemma 4.15 – see Figure 4.2. In particular, for all sets Π_k ($0 \leq k \leq 5$), every processor $\pi_i \in \Pi_k$ is connected (exactly) by one channel with all processors in $\Pi_k \setminus \{\pi_i\}$, $\Pi_{(k-1) \bmod 6}$, and $\Pi_{(k+1) \bmod 6}$. Hence, each processor π_i in the new system is symmetrically connected with exactly one processor of each type (different from his own one) as in the original system. We say that Π_k and Π_ℓ are *adjacent processor sets* if and only if $\ell \equiv k \pm 1 \pmod 6$.

Analogously to the proof of Lemma 4.15, for every set $\Pi_k \cup \Pi_{(k+1) \bmod 6}$ ($0 \leq k \leq 5$) in the new system and without the presence of an adversary, their common view is indistinguishable from their view as the set

of processors $\Pi_{k \bmod 3} \cup \Pi_{(k+1) \bmod 3}$ in the original system with respect to an adversary who corrupts all processors of the remaining processor set $\Pi_{(k+2) \bmod 3}$ in an admissible way whereas $|\Pi_{(k+2) \bmod 3}| \leq t$ by construction.

Let now π_0 and π_n be initialized with different inputs. For any possible run of the new system on inputs $x_0 = 0$ and $x_n = 1$ it holds that, chosen a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ of adjacent processor sets uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$. In particular, either

A) Chosen a pair from $\{(\Pi_5, \Pi_0), (\Pi_0, \Pi_1), (\Pi_1, \Pi_2)\}$ uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$; or

B) Chosen a pair from $\{(\Pi_2, \Pi_3), (\Pi_3, \Pi_4), (\Pi_4, \Pi_5)\}$ uniformly at random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{3}$.

If case (A) holds then, on sender input $x_0 = 0$, if the adversary selects a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ of the new system at random and corrupts $\Pi_{(k+2) \bmod 3}$ in the original system by simulating the processors in $\Sigma = \bigcup_{\ell=2}^{5} \Pi_{(k+\ell) \bmod 6}$ then the protocol fails with a probability of at least $\frac{1}{6}$. If case (B) holds then the same holds for sender input $x_0 = 1$. Hence the theorem holds for the computational case.

Furthermore, if the adversary is unbounded then, given any protocol Ψ, she can compute such a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ for which the conditions for broadcast are violated with a probability of at least $\frac{1}{3}$, accordingly corrupt the processors in $\Pi_{(k+2) \bmod 3}$, and hence forcing the protocol to fail on input

$$x_0 = \begin{cases} 0 & \text{, if } 0 \in \{k, k+1\} \text{ , and} \\ 1 & \text{, else ,} \end{cases}$$

with a probability of at least $\frac{1}{3}$. □

4.2 Broadcast from consistently shared data

Consider model $\mathcal{M}_{\text{aut}}^{\text{pki}}$, i.e., the players are connected by pairwise authenticated channels and they share a consistent PKI. As proven in [DS83], in this model, broadcast secure against any number $t < n$ of corrupted players can be efficiently achieved — and thus consensus secure against $t < n/2$.

Theorem 4.17 (Tight Bounds for Broadcast [LSP82, DS83]). *In standard models $\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki}}$ and $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki}}$, efficient broadcast as secure as the underlying PKI is possible for any $t < n$ and (efficient) consensus as secure as the underlying PKI is achievable if and only if $t < n/2$.*

Proof. The theorem follows from Theorem 4.18 (achievability of broadcast) and Proposition 4.11 (equivalence of broadcast and consensus for $t < n/2$), and Proposition 3.1 (impossibility of consensus for $t \geq n/2$). □

Thus, in contrast to unconditional security, the security of such a protocol is conditioned by the security of the underlying PKI. Such a PKI is typically set up with respect to a *computationally secure* digital signature scheme (e.g. [RSA78]). For this case, the protocol in [DS83] achieves computational security (more precisely, is as secure as the underlying signature scheme) and is very efficient. Less typically, a PKI can also be set up with respect to a pseudo-signature scheme that provides *unconditional security* with an exponentially small error probability (cf. Section 2.2.2.2). For this case the protocol in [DS83] provides unconditional security but is less efficient (though polynomial). The Dolev-Strong protocol [DS83] is given in the following section. In Section 4.2.2, a protocol with hybrid security is given: as secure as the underlying signature scheme for some threshold t_σ but still perfectly secure for some smaller threshold t_u.

4.2.1 Dolev-Strong protocol

The Dolev-Strong protocol [DS83] can be based on any PKI according to Definition 2.5, e.g., based on the computational signature scheme secure against adaptive chosen-message attacks in [GMR88] or based on the unconditionally secure pseudo-signature scheme in [PW96]. Independently of the used signature scheme, making the protocol fail implies that the adversary can forge signatures. In other words, the protocol is as secure as the used signature scheme.

The original protocol in [DS83] requires that messages are recursively resigned by different players. Here, we apply a modification in [PW96] that requires values to be signed only once.

For simplicity, let $\sigma_i(v)$ denote a signature by player p_i on value v as generated by p_i's signing algorithm and $V_i^j(v,\sigma)$ denote the verification of a signature σ on value v with respect to signer p_j as performed by player p_i: $V_i^j(v,\sigma) := V(v,\sigma,\mathrm{PK}^j)$ according to Definition 2.3 where PK^j is player p_j's public key or $V_i^j(v,\sigma) := V(v,\sigma,\mathrm{PK}_i^j,\ell)$ where PK_i^j is

player p_i's version of p_j's public key and ℓ is the transfer level according to Definition 2.4.

Let p_s be the sender with input x_s. Every player p_i maintains a set A_i of accepted values that, at the end, is either $\emptyset$, $\{0\}$, $\{1\}$, or $\{0,1\}$. Furthermore, every player p_i maintains two sets $S_i[0]$ and $S_i[1]$ where he collects signatures by the other players, signatures on 0 in $S_i[0]$ and signatures on 1 in $S_i[1]$. Informally, a player accepts a value if he has seen a valid signature on it by the sender and enough other players confirm having seen a valid signature by the sender — whereas a confirmation is itself given by signing the respective value. At the end of the protocol, every player p_i computes his output y_i depending on the set A_i of accepted values.

Protocol 4.10 `Broadcast` (P, p_s, x_s)

The whole protocol proceeds for $t+1$ rounds. In a first round, p_s computes a signature $\sigma_s(x_s)$ on his input, sends the pair $(x_s, \{\sigma_s(x_s)\})$ to every other player, computes $y_s := x_s$, and halts. During rounds $r = 1, \ldots, t+1$ every player p_i ($i \neq s$) performs the following actions where, initially, $A_i = \emptyset$:

- If any value $v \in \{0,1\}$ has been *newly* added to the set of accepted values A_i during round $r-1$ then p_i computes $\sigma_i(v)$ and sends the pair $(v, S_i[v] \cup \{\sigma_i(v)\})$ to everybody.
- If a pair (v, S) is received from any player p_j such that $v \in \{0,1\}$ and the set S contains valid signatures on v by at least r distinct players including the sender p_s then v is added to A_i, $A_i := A_i \cup \{v\}$, and the set S is memorized, $S_i[v] := S$.

At the end of the protocol, every player p_i computes output $y_i = 1$ if his set of accepted values is $A_i = \{1\}$ and $y_i = 0$ otherwise (where 0 serves as a default decision value for the cases $A_i = \emptyset$ and $A_i = \{0,1\}$).

Note that, in the case of unconditional pseudo-signatures, the signatures checked during round r must be valid with respect to level $\ell = r$, i.e., it must hold that $V(v, \sigma, \mathrm{PK}, r) = 1$. Furthermore, note that the signatures in the sets $S_i[\cdot]$ can be sent "loosely", i.e., without specifying the corresponding signers, since the recipient can simply verify each signature with respect to every player. This does not affect security in any significant way.

Theorem 4.18 (Broadcast [DS83, PW96]). *In model $\mathcal{M}^{\mathrm{pki}}_{\mathrm{aut}}$, given player set $P = \{p_1, \ldots, p_n\}$, Protocol 4.10 achieves broadcast for any number $t < n$ of corrupted players.*

The protocol requires a round complexity of $\mathcal{R} = t+1$ and a bit complexity

of $\mathcal{B} = O(n^3)|\sigma|$ where $|\sigma|$ is the maximal length of a signature including possible random padding and encoding of a session ID (cf. Section 3.1.4 "multiple invocations of Byzantine agreement").

Proof.
Validity: If the sender p_s is correct then, for every correct player p_i, $A_i = \{x_s\}$ at the end of the protocol since p_i accepts x_s after the first communication round but never accepts the value $1 - x_s$ since p_s never signs it and hence $\nexists\, \sigma_s \in S_i[1 - x_s] : V_i^s(1 - x_s, \sigma_s) = 1$. Or, in other words, any correct p_i accepting value $1 - x_s$ implies that the adversary can forge signatures. Hence, all correct players p_i decide on $y_i = x_s$.

Consistency: Assume players p_i and p_j to be correct. We show that p_i and p_j decide on the same value $y_i = y_j$ by showing that $A_i = A_j$ at the end of the protocol.

Consider any value $v \in A_i$. If p_i accepts v first during a round $r \in \{1, \ldots, t\}$ then p_i is in possession of valid signatures on v by r distinct players including p_s but excluding himself (assuming that the adversary cannot forge signatures on behalf of p_i). Hence, during phase $r + 1$, he sends $r + 1$ valid signatures by distinct players including the one by p_s to p_j who in turn accepts, and hence $v \in A_j$. On the other hand, if p_i accepts v only during phase $t+1$ then some player p_ℓ sent him $t+1$ valid signatures on v by $t + 1$ distinct players including p_s. One such player p_ℓ must be correct and hence sent the same information to every correct player, especially to p_j, and hence $v \in A_j$.

Complexity: The round complexity is evident. In the worst case, each player sends to each other two lists of $O(n)$ signatures (one list for each possible $v \in \{0, 1\}$), and hence $\mathcal{R} = O(n^3|\sigma|)$. □

Evidently, the same principle can also be used to construct protocols for consensus secure against $t < n/2$ corrupted players. Furthermore, probabilistic broadcast protocols as in Section 4.1.2 with an error probability that decreases exponentially in the number of communication rounds can also be achieved for the case that $t < n/2$ by combining the protocol in [Tou84] with the shared-coin protocol in [BS94].

4.2.2 Hybrid security

Suppose that the player set P shares a PKI according to Definition 2.5. A natural question is whether, for Byzantine agreement, unconditional security can be combined with the (typically only computational) security of the PKI's corresponding signature scheme (cf. Section 3.2.2 "Ex-

tended adversary models"). Of course, demanding both with respect to the same threshold t makes little sense since unconditional security would subsume the security of the underlying signature scheme. However, simultaneously achieving unconditional security for some "small" bound t_u while achieving the signature scheme's security for some larger bound t_σ can be strictly more powerful than any protocol described in the previous sections since then, in order to make the protocol fail, the adversary would have to either corrupt more than t_σ players — or corrupt more than t_u players even if she is able to break the underlying signature scheme or the given PKI is inconsistent.

The first such hybrid protocol to combine unconditional security and conditional security is "adaptive Byzantine agreement" by Waidner and Pfitzmann [WP89]. There, assuming a consistent PKI with respect to a so called fail-stop signature scheme, their protocol tolerates $t_\sigma < n$ with respect to computational security and $t_u < n/3$ with respect to (non-perfect) unconditional security. However, unconditional security for $t_u < n/3$ is only guaranteed if the PKI is indeed consistent. Furthermore, note that the protocol achieves strictly less than Protocol 4.10 unconditionally tolerating $t < n$ when assuming a consistent PKI with respect to unconditional pseudo-signatures [PW96].

In contrast, our construction is generic in the sense that it works for arbitrary signature schemes. Moreover, security with respect to t_u is guaranteed even if the shared PKI is inconsistent. Given a PKI with respect to a *computationally secure* signature scheme (e.g. [GMR88]), our hybrid protocol provides perfect security for threshold t_u and, given the PKI is consistent, computational security for threshold t_σ. For this case our protocols are very efficient. Given a PKI with respect to an *unconditionally secure* pseudo-signature scheme [PW96], the hybrid protocol provides perfect security for threshold t_u and, given the PKI is consistent, non-perfect unconditional security for threshold t_σ. For this case our protocols are less efficient (though polynomial).

It turns out that, in this model, broadcast is possible if and only if $2t_u + t_\sigma < n$ (whereas $t_u \leq t_\sigma$) as is proven in [Hol01]. Efficient protocols are only known for the case that $2t_\sigma < n$ is additionally satisfied, i.e., when guaranteed that the correct players form a majority. Regarding the construction of protocols, we restrict ourselves to this special case.

Theorem 4.19 (Bounds for Hybrid Broadcast [FH02]). *In model $\mathcal{M}_{\text{aut}}^{\text{pki}}$, (efficient) broadcast (or consensus) as secure as the underlying PKI for t_σ and perfectly secure for t_u is achievable for all t_σ and t_u such that $t_u \leq t_\sigma$, $2t_\sigma < n$, and $2t_u + t_\sigma < n$. If $2t_u + t_\sigma \geq n$ then there is no such protocol.*

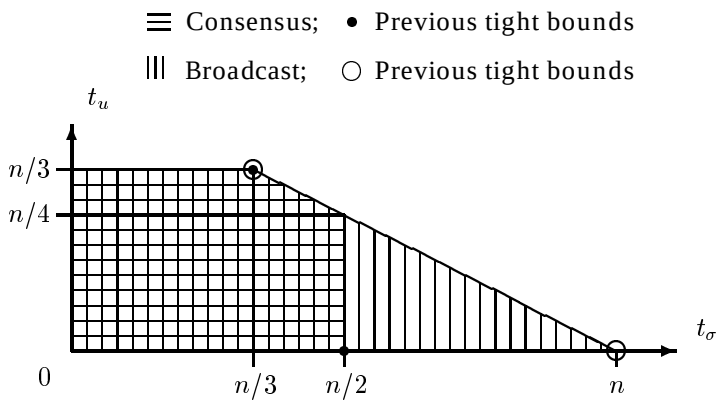

Figure 4.3: Theorem 4.19: tight bounds for broadcast and consensus with hybrid security.

Proof. The theorem follows from Theorem 4.21 (broadcast achievability), Theorem 4.22 (impossibility of broadcast), and Proposition 4.11 (equivalence of broadcast and consensus for $t < n/2$). □

Consider for example consensus. Consensus is only possible if $t < n/2$ even in model $\mathcal{M}_{\text{sec}}^{\text{pki}}[\text{c}]$. The above bound implies that optimally resilient consensus with respect to the (typically computational) security of the signature scheme, i.e., $t_\sigma < n/2$, can additionally guarantee perfect security with respect to $t \leq n/4$ "for free". Furthermore, this allows for computationally secure multi-party computation in $\mathcal{M}_{\text{sec}}^{\text{pki}}$ with optimal resilience that, at the same time, is unconditionally secure against $t \leq n/4$ corrupted players — which is strictly more powerful than previously achievable by any MPC protocol based on a consistent PKI.

4.2.2.1 Model

We consider a set of n players in model $\mathcal{M}_{\text{aut}}$, additionally sharing a (possibly inconsistent) PKI. We refer to this model as $\mathcal{M}_{\text{aut}}^{\text{pki?}}$.

As there are two bounds t_u and t_σ on the number of players an adversary possibly corrupts, it is necessary to distinguish between these bounds and the actual number f of players that are corrupted at the end of the protocol — as the adversary may be adaptive, the number of corrupted players might increase during the execution of the protocol, and reach its maximum at the end.

For the sequel of this section we make the following

Assumption 4.1. *Let n be the number of players, and t_u and t_σ be two integers such that $t_u \leq t_\sigma$, $n > 2t_\sigma$, and $n > 2t_u + t_\sigma$. Let f be the number of players corrupted by the adversary by the end of the protocol. It is assumed that:*

- *The adversary cannot corrupt more than t_σ players: $f \leq t_\sigma$.*
- *If the adversary corrupts more than t_u players then she cannot forge signatures and the given PKI is consistent.*

4.2.2.2 A phase-king protocol

Recall Theorem 4.14 in Section 4.1.3. There, with respect to unconditional security, it is shown that the achievability of weak broadcast implies the achievability of broadcast for $n > 2t$. Since $n > 2t_\sigma$ holds by Assumption 4.1, it is hence sufficient to give a protocol for weak broadcast for $n > 2t_\sigma$, $n > 2t_u + t_\sigma$, and $t_u \leq t_\sigma$. This protocol for weak broadcast can then be plugged into Protocol 4.9 for graded consensus which, in turn, can be plugged into broadcast Protocol 4.6 — maintaining the security level of the underlying protocol for weak broadcast.

As in Section 4.2.1, let again $\sigma_i(v)$ denote a signature by player p_i on value v as generated by p_i's signing algorithm and $V_i^j(v, \sigma)$ the verification of a signature σ on value v with respect to signer p_j as computed by p_i.

It is important to note that, even if the adversary is able to forge signatures, the signature by a correct player p_j is always valid. Moreover the signature $\sigma_j(v)$ by a correct player p_j on a value $v \in \{0, 1\}$ can always be assumed to guarantee that $V_i^j(v, \sigma_j(v)) = 1$ and $V_i^j(1 - v, \sigma_j(v)) = 0$.

The protocol works as follows. The sender p_s signs his input value and sends his input together with its signature to every other player: $(x_s, \sigma_s(x_s))$. Every player except for the sender now redistributes this information to everybody (but without signing this new message himself). Now, every player received n values, one from every player. Each player p_i now decides on the outcome of the protocol:

- If he received some bit v together with a valid signature by p_s at least $n - t_u$ times from different players then he computes output v.
- Otherwise, if he received a bit v together with a valid signature by p_s at least $n - t_\sigma$ times but no single correct signature by p_s for bit $1 - v$ then he decides on v.
- Otherwise, he decides on $\perp$.

Protocol 4.11 `WeakBroadcast` (P, p_s, x_s)

1. `if` $i = s$ `then` `SendToAll`$(x_s, \sigma_s(x_s))$ `fi`; `Receive`(x_i^s, σ_i^s);
2. `if` $i \neq s$ `then` `SendToAll`(x_i^s, σ_i^s) `fi`; $\forall j \neq s$: `Receive`(x_i^j, σ_i^j);
3. $S_i^0 := \{p_j \in P | x_i^j = 0 \ \wedge \ V_i^s(0, \sigma_i^j) = 1\}$;
   $S_i^1 := \{p_j \in P | x_i^j = 1 \ \wedge \ V_i^s(1, \sigma_i^j) = 1\}$;
4. `if` $\exists b \in \{0,1\} : |S_i^b| \geq n - t_u$ `then` $y_i := b$ (A)
   `elseif` $\exists b \in \{0,1\} : |S_i^b| \geq n - t_\sigma \ \wedge \ S_i^{(1-b)} = \emptyset$ `then` $y_i := b$ (B)
   `else` $y_i := \perp$ `fi`; (C)
5. return y_i

Note that, in case of unconditional pseudo-signatures, only transferability $\lambda = 2$ is required. Thus, a simpler scheme than the one in [PW96] could be used, e.g., the information checking protocol in [CDD+99].

Lemma 4.20. *In model $\mathcal{M}_{\text{aut}}^{\text{pki?}}$, under Assumption 4.1, Protocol 4.11 among the players $P = \{p_1, \ldots, p_n\}$ with sender $p_s \in P$ achieves weak broadcast.*

Its round complexity is $\mathcal{R} = 2$ and its bit complexity is $\mathcal{B} = O(n^2)|\sigma|$ where $|\sigma|$ is the maximal length of a signature including possible random padding etc. (cf. Section 3.1.4).

Proof. We show that the validity and consistency properties are satisfied. For this, let f be the number of corrupted players at the end of the protocol.

Validity: Suppose that the sender p_s is correct. Hence, every correct player p_i receives the sender's input $v := x_s$ during step 1 of the protocol, $x_i^s = v$, and a valid signature σ_i^s, i.e., $V_i^s(v, \sigma_i^s) = 1$ and $V_i^s(1 - v, \sigma_i^s) = 0$.

If $f \leq t_u$ then every correct player p_i receives the value v together with a valid signature by p_s from at least $n - t_u$ different players during steps 1 and 2. Hence, $|S_i^v| \geq n - t_u$, and p_i decides according to condition (A) in step 4. Furthermore, p_i receives the value $1 - v$ at most $t_u < n - t_u$ times which implies that $|S_i^{1-v}| < n - t_u$, and thus p_i (uniquely) computes output $y_i = v$ according to condition (A).

If $t_u < f \leq t_\sigma$ then every correct player p_i receives the value v together with a correct signature by p_s from at least $n - t_\sigma$ different players during steps 1 and 2. Hence, $|S_i^v| \geq n - t_\sigma$ and $S_i^{1-v} = \emptyset$ since the adversary cannot forge signatures in this case (cf. Assumption 4.1). Hence, p_i computes $y_i = v$ according to either condition (A) or (B).

Consistency: Suppose that some correct player p_i computes output $v := y_i \neq \perp$. We have to show that hence, every correct player p_j computes an output $y_j \in \{v, \perp\}$.

Suppose first, that p_i decides according to condition (A) in step 4, i.e., $|S_i^v| \geq n - t_u$. For p_j this implies that $|S_j^v| \geq |S_i^v| - t_\sigma \geq n - t_u - t_\sigma > t_u$

and hence that $|S_j^{1-v}| < n - t_u$ and $S_j^v \neq \emptyset$. Hence, p_j cannot compute $y_j = 1 - v$, neither according to condition (A) nor to condition (B).

On the other hand, suppose that p_i decides according to condition (B) in step 4, i.e., $|S_i^v| \geq n - t_\sigma$ and $S_i^{1-v} = \emptyset$. Since $n > 2t_\sigma$ this implies that at least one correct player relayed a correct signature by p_s on v, and hence, $S_j^v \neq \emptyset$ and $|S_j^{1-v}| \leq t_\sigma < n - t_\sigma \leq n - t_u$. Hence, if p_j computes $y_j \neq \perp$ according to condition (A) then $y_j = v$ because of the argument of the last paragraph; and if p_j computes $y_j \neq \perp$ according to condition (B) then $y_j = v$ since $S_j^v \neq \emptyset$.

Complexity: Round and bit complexities can be easily verified by code inspection. □

Theorem 4.21 (Broadcast [FH02]). *In model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{pki?}}$, under Assumption 4.1, broadcast is efficiently achievable, i.e., broadcast as secure as the underlying PKI with respect to t_σ and perfectly secure with respect to t_u where $t_u \leq t_\sigma$, $2t_\sigma < n$, and $2t_u + t_\sigma < n$.*

Thereby a round complexity of $\mathcal{R} = 5t + 1$ and a bit complexity of $\mathcal{B} = O(n^4)|\sigma|$ can be achieved where $|\sigma|$ is the maximal length of a signature including possible random padding etc. (cf. Section 3.1.4).

Proof. Achievability and complexity follow from Lemma 4.20 and Theorem 4.14. □

4.2.2.3 Impossibility result

It now remains to show that the bound $2t_u + t_\sigma < n$ is tight when $t_u > 0$. Note that, for this impossibility proof, it is not assumed that $2t_\sigma < n$. The impossibility result even holds for model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki?}}$, i.e., in model $\mathcal{M}_{\mathrm{sec}}$ with an additional, possibly inconsistent PKI.

Theorem 4.22 (Impossibility [FH02]). *Let t_u and t_σ be bounds such that $0 < t_u \leq t_\sigma$, and $2t_u + t_\sigma \geq n$. Then, in model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{pki?}}$, there is no broadcast protocol among n players that is as secure as the PKI with respect to t_σ and unconditionally secure with respect to t_u.*

For every protocol there exists a sender input $x_0 \in \{0, 1\}$ such that the adversary can make the protocol fail with a probability of at least $\frac{1}{3}$ — either being unbounded and corrupting t_u players or being able to forge signatures (or having made for an inconsistent PKI) and corrupting t_σ players.

Proof. The proof proceeds similarly to the proof of Theorem 4.16. For the sake of contradiction, we assume such a broadcast protocol to exist.

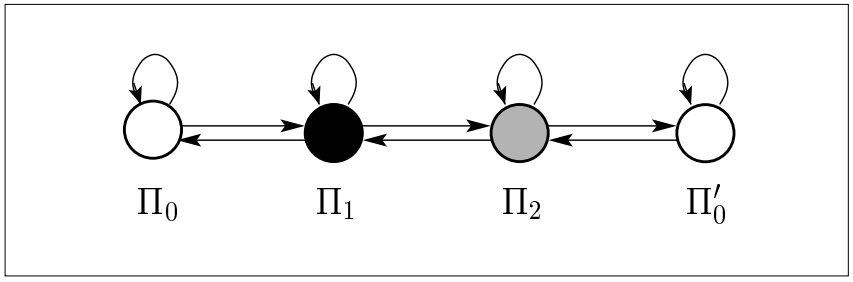

Figure 4.4: *Rearrangement of processors in proof of Theorem 4.22.*

This protocol is then used to build a different system with contradictory behavior.

Assume any protocol Ψ for a player set P with $|P| = n \geq 3$ that tolerates $2t_u + t_\sigma \geq n$ (with $t_u > 0$), i.e., that is secure for the case that either the adversary corrupts up to t_u players, or that the adversary corrupts up to t_σ players but is unable to forge signatures (and the PKI is consistent).

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs. Since $0 < t_u \leq t_\sigma$, it is possible to partition the processors into three sets, $\Pi_0 \dot{\cup} \Pi_1 \dot{\cup} \Pi_2 = \Pi$, such that $1 \leq |\Pi_0| \leq t_\sigma$, $1 \leq |\Pi_1| \leq t_u$, and $1 \leq |\Pi_2| \leq t_u$. Note that, hence, $|\Pi_0 \cup \Pi_1| \geq n - t_u$, $|\Pi_1 \cup \Pi_2| \geq n - t_\sigma$, and $|\Pi_2 \cup \Pi_0| \geq n - t_u$.

Furthermore, for each $\pi_i \in \Pi_0$, let π'_i be an identical copy of processor π_i. Let the number i denote the *type* of any processor π_i (or π'_i, respectively). Finally, let $\Pi'_0 = \{\pi'_i \mid \pi_i \in \Pi_0\}$ form an identical copy of set Π_0.

Instead of connecting the original processors as required for broadcast, we build a network involving all processors in $\Pi_0 \cup \Pi_1 \cup \Pi_2 \cup \Pi'_0$ with their pairwise communication channels connected in a way such that each processor π_i (or π'_i) communicates with exactly one processor of each type $j \in \{1, \ldots, n\} \setminus \{i\}$. Consider Figure 4.4. Exactly all pairs in $(\Pi_0 \cup \Pi_1) \times (\Pi_0 \cup \Pi_1)$, $(\Pi_1 \cup \Pi_2) \times (\Pi_1 \cup \Pi_2)$, and $(\Pi_2 \cup \Pi'_0) \times (\Pi_2 \cup \Pi'_0)$ are connected by pairwise channels (excluding reflexive connections). There are no connections between the sets Π_0 and Π_2, and no connections between the sets Π_1 and Π'_0. Messages that originally would have been sent from a processor in Π_0 to a processor in Π_2 are discarded. Messages that originally would have been sent from a processor in Π_2 to a processor in Π_0 are delivered to the corresponding processor in Π'_0. Messages sent from a processor in Π'_0 to a processor in Π_1 are discarded.

We now show that for the sets $\Pi_0 \cup \Pi_1$, $\Pi_1 \cup \Pi_2$, and $\Pi_2 \cup \Pi'_0$, their joint view is indistinguishable from their view in the original setting for

an adversary corrupting the remaining processors in an admissible way.

The adversary can corrupt all processors in Π_0 in the original system by simulating the processors in Π_0 and Π_0' of the new system. This is possible since $|\Pi_0| \leq t_\sigma$ and since, by corrupting the processors, she learns the secret keys of all corresponding players with processors in Π_0. Thus the joint view of the processors in $\Pi_1 \cup \Pi_2$ in the original system is exactly the same as their view in the new system.

Analogously, the adversary can corrupt all the processors in Π_1 in the original system by simulating the processors in $\Pi_0 \cup \Pi_1$ of the new system — where Π_0 simply is an imaginary set for the adversary's strategy. This is possible since $|\Pi_1| \leq t_u$, since she knows all public keys, and since she can forge signatures in this case. Thus the joint view of the processors in $\Pi_2 \cup \Pi_0$ in the original system is exactly the same as the view of the processors in $\Pi_2 \cup \Pi_0'$ in the new system.

Finally, the adversary can corrupt all the processors in Π_2 in the original system by simulating the processors in $\Pi_2 \cup \Pi_0'$ of the new system. This is again possible since $|\Pi_2| \leq t_u$, she knows all public keys, and she can forge signatures in this case. Thus the joint view of the processors in $\Pi_0 \cup \Pi_1$ in the original system is exactly the same as their view in the new system.

Let now the sender in Π_0 be assigned input 0 and the sender's copy in Π_0' be assigned input 1. Hence, for each run of the new system, there must be a set among $S_2 = \Pi_0 \cup \Pi_1$, $S_0 = \Pi_1 \cup \Pi_2$, and $S_1 = \Pi_2 \cup \Pi_0'$, for whom either validity or consistency is violated although being completely consistent with a set of corresponding processors in the original system. In particular, there is a set S_i ($i \in \{0,1,2\}$) such that, over all possible such runs, the probability that the conditions of broadcast are violated for the processors in S_i is at least $\frac{1}{3}$.

If this holds for $i = 0$ then even a computationally bounded adversary can make the protocol fail with a probability of at least $\frac{1}{3}$ by corrupting the players in Π_0 and acting accordingly (simulating Π_0 with respect to sender input 0 and simulating Π_0' with respect to sender input 1).

On the other hand, if this holds for some $i \in \{1,2\}$ then a computationally unbounded adversary can make the protocol fail with a probability of at least $\frac{1}{3}$ by corrupting the players in Π_i (for input $x_0 = 1$ if $i = 1$ and for input $x_0 = 0$ if $i = 2$). Since the adversary is unbounded, given protocol Ψ, she can compute such an $i \in \{1,2\}$ and act accordingly. □

4.2.2.4 Applications

When additionally assuming pairwise *secure* channels instead of only authenticated ones, Theorem 4.21 immediately implies that also MPC is achievable with respect to the bounds $2t_\sigma < n$ and $2t_u + t_\sigma < n$.

Theorem 4.23 (Tight Bounds for MPC [FH02]). *In model $\mathcal{M}_{\text{sec}}^{\text{pki?}}$, MPC unconditionally secure with respect to t_u and as secure as the underlying PKI with respect to t_σ is efficiently achievable if $t_u \leq t_\sigma$, $2t_\sigma < n$, and $2t_u + t_\sigma < n$. This bound is tight — if either $2t_\sigma \geq n$ or $2t_u + t_\sigma \geq n$ then MPC is not achievable.*

Proof. The protocol in [CDD+99] for model $\mathcal{M}_{\text{sec}}^{\text{bc}}$ achieves unconditionally secure MPC with an error probability exponentially small in the security parameter. Substituting each invocation of a broadcast channel with hybrid broadcast along the lines of Theorem 4.21 thus immediately yields an MPC protocol for model $\mathcal{M}_{\text{sec}}^{\text{pki?}}$.

If the adversary corrupts at most t_u players then all broadcast invocations are unconditionally secure, and hence so is the MPC protocol. If the adversary corrupts at most t_σ players then the broadcast invocations are as secure as the underlying PKI and signature scheme, and hence also the MPC protocol, i.e., making the MPC protocol fail with non-negligible probability implies being able to forge signatures (or that the PKI is inconsistent).

It is well known that computationally secure MPC guaranteeing both privacy and robustness, is not possible if $t \geq n/2$ which implies the necessity of $2t_\sigma < n$. Since broadcast is the special case of an MPC, $2t_u + t_\sigma < n$ is also necessary for MPC as follows from Theorem 4.22. □

4.3 Consistently shared data from broadcast

A key ingredient for some of the following constructions are protocols that, given "temporary" broadcast, perform a precomputation among the players that allows for future broadcast secure against $t < n$ corrupted players even when this "temporary" broadcast is not available anymore. In other words, such a protocol preserves the functionality of broadcast for any later time. This can be done by computing consistently shared data among the players (cf. Section 2.3.4).

Definition 4.3 (Precomputation). *A protocol among n players where every player $p_i \in P$ computes some private data Δ_i achieves* precomputation for broadcast *(or* precomputation, *for short) if it satisfies*

Correctness: broadcast is achievable after an execution of this protocol.

Independence: A correct player's intended input value for any broadcast precomputed for need not be known at the time of the precomputation. ◇

Such a precomputation can only exist for some limited number b of later broadcasts. A precomputation protocol that allows for at least b later broadcasts is said to achieve *precomputation for b broadcasts*.

Independence implies two important properties: first, the precomputation may be done long before the actual broadcasts it is used for; and second, that the adversary gets no information about any future inputs by correct senders.

In order to get computational security, such a precomputation can be easily achieved (given that secure signature schemes exist):

Protocol 4.12 $\texttt{Precomp}(P)$

With respect to the given signature scheme, every player generates a secret-key / public-key pair, and, during the phase where temporary broadcast is achievable, every player broadcasts his public key to the other players.

Proposition 4.24. *In model $\mathcal{M}_{\mathrm{aut}}$, when additionally given temporary broadcast, Protocol 4.12 achieves precomputation for broadcast computationally secure against $t < n$ corrupted players (in plain model $\mathcal{M}_{\mathrm{aut}}$ again).*

Let $\mathcal{R}_{\mathrm{bc}}$ and $\mathcal{B}_{\mathrm{bc}}$ be the round and bit complexities of the given temporary broadcast (for broadcasting one bit). Then there is a precomputation protocol that requires $\mathcal{R} = \mathcal{R}_{\mathrm{bc}}$ rounds and a bit complexity of $\mathcal{B} = nk \cdot \mathcal{B}_{\mathrm{bc}}$ where k is the maximal size of a public-key. Independently of the precomputation, the protocol for future broadcast can be any protocol based on digital signatures.

Proof. The correctness of the protocol is obvious: After the protocol, they share a consistent PKI and can, for example, use Dolev-Strong Protocol 4.10 for broadcast that is secure against $t < n$ corrupted players. In the protocol, n players broadcast a public key in parallel, hence yielding the given complexities $\mathcal{R}$ and $\mathcal{B}$. □

In order to get unconditional security, more involved methods must be applied but still, the goal of the precomputation is the same: to set up a consistently shared PKI. The first such protocol was given by Baum-Waidner, Pfitzmann, and Waidner in [BPW91]. This protocol allows for later broadcast secure against $t < n/2$ corrupted players. In [PW96, Wai92], a protocol is constructed that allows for later broadcast secure

against $t < n$ corrupted players. More details about this protocol are given in the following section.

4.3.1 Pfitzmann-Waidner protocol

The Pfitzmann-Waidner Protocol uses temporary broadcast in order to set up a pseudo-signature scheme (Definition 2.4) among the players. The fact that pseudo-signatures are only transferable for a finite number of times allows for unconditional security. Finite transferability obviously is sufficient for broadcast. For example, the Dolev-Strong Protocol requires a transferability of $t + 1$ as can be easily verified.

The pseudo-signature scheme is set up with respect to every single player as a future signer as follows. Between the signer and every other player (the future verifiers), many instances of a scheme for unconditionally secure message authentication codes (e.g., [WC81]) is set up in a way that the signer does not learn which keys belong to which players. This can be done by the verifiers choosing their respective keys and anonymously sending them to the signer with the Dining-Cryptographers Protocol [Cha88, BB90]. Note that the Dining-Cryptographers protocol requires secure channels and broadcast, $\mathcal{M}_{\text{sec}}^{\text{bc}}$.

Later a signer will simply sign a value by computing the value's authentication code with respect to every single key. Verification depends on the "transfer level" of the signature. Getting the signature directly from the signer, a verifier accepts the signature if and only if all authentication codes with respect to the verifier's own keys are correct. The more intermediate players the signature is passed through, the fewer authentication codes are required to match with respect to the final verifier's keys.

Let s be the index of the sender of the future broadcast.

Protocol 4.13 $\texttt{Precomp}(P)$

- For $(j = s, A)$, and (j, A), (j, B) ($j \in \{1, \ldots, n\} \setminus \{s\}$) in parallel:
  - For $k = 1 \ldots m$ in parallel:
    - Repeat until everything "consistent" (less than $n^2/2$ times):
      - If $i \neq j$: select random authentication key κ_i;
      - For $\ell = 1 \ldots 2n$ in parallel:
        - $\forall p_h$ ($h \neq i$) agree on a pairwise key $K_{hi}^{(\ell)}$;
        - broadcast $\kappa_i^{2\ell-1} + \sum_{h \neq i} K_{hi}^{(\ell)}$;
        - If $i = j$: broadcast whether a fault occurred ("complaint");
        - broadcast information and eliminate players/keys;

Proposition 4.25 (Pfitzmann-Waidner [PW96]). *In model $\mathcal{M}_{\mathrm{sec}}$, for any integer $b > 0$ and security parameter $\kappa > 0$, there is a protocol that, additionally given temporary broadcast, achieves precomputation for b broadcasts unconditionally secure against $t < n$ corrupted players (in plain model $\mathcal{M}_{\mathrm{sec}}$ again — or even model $\mathcal{M}_{\mathrm{aut}}$). Thereby the error probability of each one of the b later broadcasts is $\varepsilon < 2^{-\kappa}$.*

Let $\mathcal{R}_{\mathrm{bc}}$ and $\mathcal{B}_{\mathrm{bc}}$ be the round and bit complexities of the given broadcast channel (for broadcasting one single bit). There is a precomputation protocol requiring round complexity $\mathcal{R} \leq \frac{n^2}{2} + \frac{3}{2}n^2 \cdot \mathcal{R}_{\mathrm{bc}}$ and bit complexity $\mathcal{B} = O(n^8 b \log\log|\mathcal{D}|(\kappa + \log n + \log\log\log|\mathcal{D}|)^2 \cdot (1 + \mathcal{B}_{\mathrm{bc}}))$.

Independently of the precomputation, the protocol for future broadcast can be any protocol based on digital signatures (but using pseudo-signatures instead). In particular, Protocol 4.10 then has round complexity $\mathcal{R}' = t + 1$ and bit complexity $\mathcal{B}' = O(n^2 \log|\mathcal{D}| + n^6(\kappa + \log n)^2)$.

The correct players all terminate the precomputation protocol and every single later broadcast protocol during the same communication rounds.

Proof. The proposition follows from the analysis in [PW96] for the precomputation for one single broadcast of an element in domain $\mathcal{D}$ — their protocol is simply run b times in parallel. □

In [PW96, Wai92], also a regeneration technique is described that allows to precompute for b later broadcasts requiring a communication complexity that depends on b only in polylogarithmic order (as compared to logarithmic in the computational case). The idea is to initially precompute for a "sufficient" number of broadcasts that allows to run a new precomputation protocol plus at least one additional broadcast to be used for its "application", i.e., together with each one of the b later broadcast protocols an additional regeneration protocol is run. The disadvantage of this solution is that the round complexity for later broadcast basically grows to n^3 and that the bit complexity depends on n in order of $\Omega(n^{17})$. However we shall keep in mind the following

Proposition 4.26 (Pfitzmann-Waidner [PW96]). *In model $\mathcal{M}_{\mathrm{sec}}$, for any integer $b > 0$ and security parameter $\kappa > 0$, there is a protocol that, additionally given temporary broadcast, achieves precomputation for b broadcasts unconditionally secure against $t < n$ corrupted players (in plain model $\mathcal{M}_{\mathrm{sec}}$ again). Thereby the error probability of each one of the b later broadcasts is $\varepsilon < 2^{-\kappa}$.*

If temporary broadcast is efficient then the precomputation and the b later broadcast protocols have round complexities polynomial in n and bit complexities polynomial in n, κ, $\log|\mathcal{D}|$, and $\log b$.

The correct players all terminate the precomputation protocol and every single later broadcast protocol during the same communication rounds.

Chapter 5

Extended Communication Models

5.1 Introduction

Opposed to the standard communication models $\mathcal{M}_{\mathrm{aut}}$, $\mathcal{M}_{\mathrm{sec}}$, $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}}$, or $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}}$, several "non-standard" communication models have been studied in context of different problems.

Incomplete networks. Dolev [Dol82] considered the problem of general incomplete networks where not necessarily every pair of players is connected by a channel. He proved that, in this model, Byzantine agreement unconditionally secure against an active adversary requires the additional property that, between every pair of nodes, there exist $2t + 1$ node-disjoint paths. Dolev, Dwork, Waarts, and Yung [DDWY93] studied the problem of perfectly secure communication over general (incomplete) point-to-point communication networks in presence of a mixed-type adversary with simultaneous active and passive corruption.

Partial broadcast. Franklin and Yung [FY95] considered the problem of unconditionally secure private point-to-point communication in the presence of a passive adversary, given partial-broadcast but not necessarily private communication channels among pairs of players. Franklin and Wright [FW00], and Wang and Desmedt [WD01], considered unconditionally secure point-to-point communication over local-broadcast net-

works in the presence of an active adversary.

External information sources. An external information source is a device that repeatedly submits information to the involved players. Such primitives are of special interest since they are non-interactive as, for instance, opposed to a primitive to achieve partial-broadcast. An external information source could for instance be realized by a satellite. Rabin [Rab83b] considered problems such as contract signing with fair exchange with help of an external information source to distribute random integers to the involved parties. Blum, Feldman, and Micali [BFM88] showed that, in this model, non-interactive zero-knowledge is achievable. Maurer [Mau93] considered unconditionally secure two-party secret-key agreement with help of authenticated channels and an external random source whose signal cannot be perfectly received, neither by the parties nor by the adversary.

This chapter focuses on two particular models, both based on model either $\mathcal{M}_{\mathrm{aut}}$ or $\mathcal{M}_{\mathrm{sec}}$. In the first model, $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$ or $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}_b}$, besides of the pairwise channels, it is assumed that every set of b players can reliably broadcast information to each other (either with help of a protocol, tamper proof hardware, or whatever). In the second model, $\mathcal{M}_{\mathrm{aut}}^{\mathrm{q}}$ or $\mathcal{M}_{\mathrm{sec}}^{\mathrm{q}}$, besides of the pairwise channels, it is assumed that the players have access to a common, external information source. Throughout this chapter, we assume no data to be consistently shared among the players. All constructions provide unconditional security whereas the impossibility proofs are given with respect to computational security.

One of the main results is that, in $\mathcal{M}_{*}^{\mathrm{bc}_3}$ (the minimal extension over pairwise communication), global broadcast is achievable if and only if $t < n/2$. In context of multi-party computation this implies that for the protocols in [Bea89, RB89, CDD+99] global broadcast is not required but that broadcast among triplets of players is sufficient. Furthermore, in general model $\mathcal{M}_{*}^{\mathrm{bc}_b}$ ($b > 3$), global broadcast is achievable even against faulty majorities.

5.2 Motivation

Although it is arguable how realistic such "non-standard" models might be, from a theoretical point of view, it can be interesting for several reasons. First of all, it is a natural question how much partial consistency

among subsets of players is required in order to achieve global consistency among all players. Second, the generic reduction of complex tasks to simple ones is a useful tool in order to prove whether or not a task is achievable under given conditions, only requiring a construction for the simple task in order to prove the achievability of the complex one, and only requiring to show the impossibility of the complex task in order to prove the unachieveability of the simple one. For example, such a reduction together with an application was already given in Chapter 4. Theorem 4.14 states that weak broadcast implies broadcast if $t < n/2$. In Section 4.2.2, this theorem was applied in order to show that broadcast with hybrid security is achievable if $2t_u + t_\sigma < n$ and $2t_\sigma < n$ by merely giving a construction for weak broadcast. Third, in view of the fact that communication takes place in a physical world where some level of consistency is guaranteed, for instance by exploiting the proximity of entities communicating by radio channels, it is interesting to investigate how such given minimal consistency guarantees can be exploited and amplified. While the models given here can perhaps not directly be motivated as being available in the physical world, other models might be.

5.3 Partial broadcast

We start with an efficient, optimally resilient protocol for model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$ and some further considerations related to this model (Section 5.3.1). In Section 5.3.2 a generic, optimally resilient protocol for model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$ for any $b \geq 2$ is given. That protocol, however, generally is not efficient. Finally, optimal resilience of the given protocols for models $\mathcal{M}_{*}^{\mathrm{bc}_b}$ is proven in Section 5.3.3. The general tight bounds are anticipated by the following

Theorem 5.1. *In model $\mathcal{M}_{*}^{\mathrm{bc}_b}$, global broadcast among $n > 2$ players is achievable if and only if $t < \frac{b-1}{b+1}n$.*

Proof. The theorem follows from Theorems 5.10 and 5.11. □

5.3.1 3-broadcast

5.3.1.1 Model

We consider model $\mathcal{M}_{\mathrm{aut}}$ (pairwise authenticated channels) but, additionally, we assume that unconditionally secure, synchronous broadcast channels among each triple of players are available, i.e., that for each subset of three players ($S \subset P$, $|S| = 3$) and for any selection of a sender

among them there is an authenticated broadcast channel from the sender to the remaining two players. Such broadcast channels from a sender to two recipients will be denoted as *BC_3 channels*. The security of the BC_3 channels is not necessarily required to be perfectly secure but we assume their error probability ε_0 to be customizable to an arbitrarily small level. More precisely, we assume that, given any security parameter $\kappa > 0$, such a channel can be customized such that its error probability is bounded by $2^{-\kappa}$, $\varepsilon < 2^{-\kappa}$. We will refer to this model as $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$.

5.3.1.2 A phase-king protocol

In this section, we give the construction of an efficient broadcast protocol that tolerates $t < n/2$ corrupted players. As follows from Theorem 4.14 (reduction from broadcast to weak broadcast for $n > 2t$), it is sufficient to give an efficient protocol for weak broadcast. By the construction behind this theorem, the final broadcast protocol will be a phase-king protocol.

Let p_s be the sender. The following protocol for weak broadcast is simple: p_s simultaneously sends his input value with all BC_3 channels he is involved in as a sender, i.e., all $\binom{n-1}{2}$ BC_3 channels with sender p_s and two distinct recipients in $P \setminus \{p_s\}$. Every recipient p_i then decides on $v \in \{0,1\}$ if he received v in each of the $n-2$ invocations of BC_3 he was involved in and on $\perp$, otherwise.

In the following protocol, let $\texttt{Broadcast}(\{p_s, p_i, p_j\}, p_s, x_s)$ denote an invocation of the BC_3 channel among $\{p_s, p_i, p_j\}$ with sender p_s where p_s holds input x_s.

Protocol 5.1 $\texttt{WeakBroadcast}(P, p_s, x_s)$

1. $\forall j \in \{1, \ldots, n\} \setminus \{s, i\}$: $v_i^j := \texttt{Broadcast}(\{p_s, p_i, p_j\}, p_s, x_s)$;
2. **if** $\exists v \in \{0,1\}\ \forall j : v_i^j = v$ **then** $y_i := v$ **else** $y_i := \perp$ **fi**;
3. **return** y_i

Lemma 5.2. *In model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$, Protocol 5.1 achieves efficient weak broadcast secure against any number of corrupted players.*

In order to achieve that the error probability satisfies $\varepsilon < 2^{-\kappa}$, the security parameter κ_{bc_3} of the underlying BC_3 channel can be set to $\kappa_{\mathrm{bc}_3} \geq \kappa + 2\log_2 n = \kappa + O(\log n)$.

Let $\mathcal{R}_{\mathrm{bc}_3}$ be the round complexity and $\mathcal{B}_{\mathrm{bc}_3}$ be the bit complexity of the given BC_3 channel. Then Protocol 5.1 requires a round complexity of $\mathcal{R} = \mathcal{R}_{\mathrm{bc}_3}$ and a bit complexity of $\mathcal{B} = \binom{n-1}{2} \cdot \mathcal{B}_{\mathrm{bc}_3} = O(n^2)\mathcal{B}_{\mathrm{bc}_3}$.

Proof. Consider Protocol 5.1. The validity property follows from the fact that a correct sender sends the same value x_s for all instances of BC_3, and hence a correct recipient p_i consistently receives $v_i^* \equiv x_s$, and decides $y_i = x_s$. The consistency property follows from the fact that any two correct players p_i and p_j share an instance of BC_3, namely the one among $\{p_s, p_i, p_j\}$ where both receive the same value. Hence, if p_i decides on $y_i \in \{0,1\}$ then p_j decides on $y_j \in \{y_i, \perp\}$.

There are $\binom{n-1}{2} = \frac{(n-1)(n-2)}{2} < \frac{n^2}{2}$ executions of BC_3 and hence, the overall error probability of Protocol 5.1 is at most $\varepsilon < \frac{n^2}{2} 2^{-\kappa_{\mathrm{bc}_3}}$ (union bound). Thus, demanding $\kappa_{\mathrm{bc}_3} \geq \kappa + 2\log_2 n \geq \kappa + \log_2(\frac{n^2}{2})$ yields

$$\varepsilon < \frac{n^2}{2} 2^{-\kappa_{\mathrm{bc}_3}} \leq 2^{-\kappa} .$$

The complexity of the protocol can be easily verified by code inspection. □

Theorem 5.3 (Broadcast [FM00b, FM00a]). *In model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$, global broadcast unconditionally secure against $t < n/2$ corrupted players is efficiently achievable.*

Let $\mathcal{R}_{\mathrm{bc}_3}$ and $\mathcal{B}_{\mathrm{bc}_3}$ be the round and bit complexities of the underlying BC_3 channels. For any $\kappa > 0$, given BC_3 channels with security parameter $\kappa_{\mathrm{bc}_3} \geq \kappa + \log_2(3tn^3) = \kappa + O(\log n)$, there is a broadcast protocol with security parameter κ (error probability $\varepsilon < 2^{-\kappa}$) requiring round complexity $\mathcal{R} = 2t \cdot \mathcal{R}_{\mathrm{bc}_3} + t + 1$ and bit complexity $\mathcal{B} = O(n^4)\mathcal{B}_{\mathrm{bc}_3}$.

Proof. Consider the broadcast protocol resulting from plugging weak broadcast Protocol 5.1 into graded consensus Protocol 4.9 and, in turn, plugging Protocol 4.9 into Protocol 4.6. By Lemmas 5.2, 4.13, and 4.7 this protocol obviously tolerates $t < n/2$ corrupted players and requires the stated round and bit complexities. The overall number of BC_3 invocations is upper-bounded by $\ell = 3tn^3$, and hence the error probability can be estimated as

$$\varepsilon < 3tn^3 2^{-\kappa_{\mathrm{bc}_3}} \leq 2^{-\kappa} .$$

□

Note that the bilateral authenticated channels of model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$ are not required in the given construction. The same correctness argumentation still holds when replacing each send from a player p_i to a player p_j by a

respective broadcast on the BC_3 channel from p_i to player p_j and some arbitrary other player $p_k \notin \{p_i, p_j\}$.

For the given constructions, we assumed the underlying BC_3 channels to be reliable independently of the number of corrupted players involved. Consider broadcast among three players secure against only one corrupted player, denoted by $\mathrm{BC}_{3,1}$ (standing for $\mathrm{BC}_{n=3,t=1}$). $\mathrm{BC}_{3,1}$ immediately implies BC_3, i.e., resilience against any number of corrupted players, since, in the presence of more than one corrupted player, the only condition to be satisfied is that a correct sender decides on his own input value. This can be easily guaranteed. Moreover, weak broadcast among three players secure against one corrupted player, denoted by $\mathrm{WBC}_{3,1}$, implies $\mathrm{BC}_{3,1}$, as follows from the general reduction of Theorem 4.14 — a more efficient reduction for the special case $n = 3$ is implied by the proof of the following lemma.

Lemma 5.4. *In model $\mathcal{M}_{\mathrm{aut}}$, $\mathrm{WBC}_{3,1}$ implies BC_3.*

Given that the $\mathrm{WBC}_{3,1}$ channels involve security parameter κ_{wbc_3}, round complexity $\mathcal{R}_{\mathrm{wbc}_3}$, and bit complexity $\mathcal{B}_{\mathrm{wbc}_3}$, there is a protocol for BC_3 with security parameter $\kappa = \kappa_{\mathrm{wbc}_3}$, round complexity $\mathcal{R} = \mathcal{R}_{\mathrm{wbc}_3} + 1$ and bit complexity $\mathcal{B} = \mathcal{B}_{\mathrm{wbc}_3} + O(1)$.

Proof. Given $\mathrm{WBC}_{3,1}$, BC_3 can be implemented as follows: First, the sender distributes his value with $\mathrm{WBC}_{3,1}$. Then both recipients exchange the values they have received from the sender. A recipient who received a value $x \neq \perp$ from the sender sticks to this value, $y := x$, whereas in the other case ($x = \perp$), he decides on $y := x'$ where x' is the value received from the other recipient (during the second round). Finally, if $y = \perp$, y is replaced by a default value, e.g., $y := 0$.

Hence, if the sender is correct, a correct recipient always decides on the sender's value. On the other hand, if the sender is corrupted, then two correct recipients either receive the same value $x \in \{0, 1\}$ or at least one of them receives $x = \perp$ (by the consistency property of weak broadcast) which makes him adopt the other players' value x' if $x' \in \{0, 1\}$. Finally, if the weak broadcast results in $x = \perp$ for both recipients then both of them replace their values by the default bit. □

There remains the natural question of what happens "between" the extremes of $\mathrm{WBC}_{3,1}$ and global broadcast secure against $t < n/2$ corrupted players. More generally, suppose that we are given a protocol Ψ for weak broadcast among ν players where $\nu \geq 3$ (but not necessarily $\nu \leq n$). How much resilience is necessary and sufficient for Ψ in order to allow for broadcast among the n players with respect to some fixed

resilience $t \in [\lceil n/3 \rceil, n/2)$? It is not hard to see that any protocol for weak broadcast among $\nu \geq 3$ players with $\tau \in [\lceil n/3 \rceil, n)$ ($\mathsf{WBC}_{\nu,\tau}$) allows for broadcast among $n \geq 3$ players with $t \in [\lceil n/3 \rceil, n/2)$ ($\mathsf{BC}_{n,t}$) and vice versa, i.e., that all respective primitives are equivalent. This fact is stated in the following theorem.

Theorem 5.5. *In any model implying synchronous authenticated communication, (efficient) $\mathsf{WBC}_{\nu,\lceil \nu/3 \rceil}$ implies*

- *(efficient) $\mathsf{WBC}_{n,n-1}$, and*
- *(efficient) $\mathsf{BC}_{n,t}$ if and only if $t \leq \lfloor \frac{n-1}{2} \rfloor$.*

Proof. The lemma follows from the following four efficient reductions, and from Theorem 5.11 which implies that, in model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}_3}$, broadcast for $t \geq n/2$ is impossible.

1. $\mathsf{WBC}_{\nu,\lceil \nu/3 \rceil} \Rightarrow \mathsf{WBC}_{3,1}$: Assume that the three players $\{p_1, p_2, p_3\}$ want to achieve weak broadcast with sender p_1 secure against one corrupted player. For this, each player p_i simulates k_i of the players in the given $\mathsf{WBC}_{\nu,\lceil \nu/3 \rceil}$-protocol such that $1 \leq k_i \leq \lceil \nu/3 \rceil$ and such that the sender of the given protocol is simulated by player p_1. At the end, every player p_i decides on the output of any one of the players he simulates. If at most one player in $\{p_1, p_2, p_3\}$ is corrupted then, in the given protocol for weak broadcast, no more players are simulated by a corrupted player than tolerated by the protocol. Hence, this simulation achieves $\mathsf{WBC}_{3,1}$ among the players $\{p_1, p_2, p_3\}$.
2. $\mathsf{WBC}_{3,1} \Rightarrow \mathsf{BC}_3$: This reduction is given by Lemma 5.4.
3. $\mathsf{BC}_3 \Rightarrow \mathsf{WBC}_{n,n-1}$: This reduction is given by Lemma 5.2.
4. $\mathsf{BC}_3 \Rightarrow \mathsf{BC}_{n,\lfloor \frac{n-1}{2} \rfloor}$: This reduction is given by Theorem 5.3.

□

5.3.2 General b-broadcast

In this section, model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_3}$ is generalized to a model with broadcast among any number $b > 2$ of players and, depending on b, we completely characterize the number t of corrupted players that can be tolerated for global broadcast when given partial broadcast among b players.

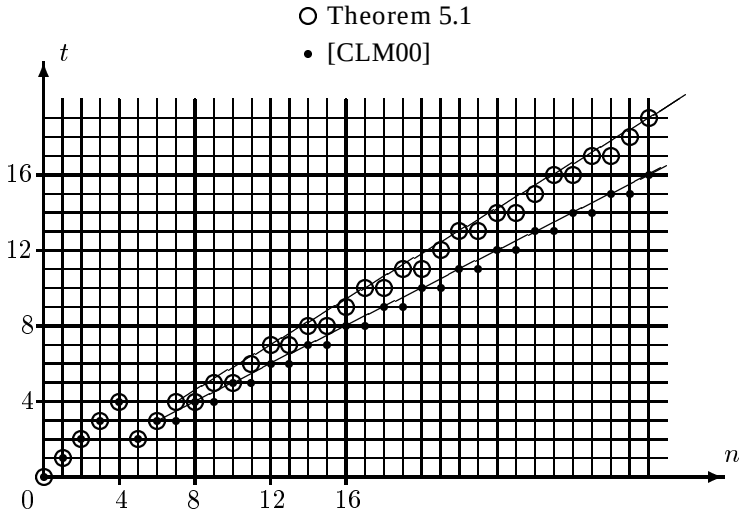

Figure 5.1: Comparison of Theorem 5.1 to [CLM00] for the special case of $b = 4$.

The first results for this general model were given in [CLM00] by Considine, Levin, and Metcalf. They showed that, for any integer $k \geq 1$, broadcast among $b = 2k$ players is necessary and sufficient for global broadcast secure against $t = \frac{k-1}{k}n$ corrupted players. However, their result does not give a complete picture of what is exactly achievable since only even integers $b = 2k$ ($k \geq 1$) and resiliences of the exact form $t = \frac{k-1}{k}n$ are considered. In contrast, we give a precise tight bound by showing that, in a model with broadcast among $b \geq 2$ players, global broadcast is achievable if and only if $t < \frac{b-1}{b+1}n$. Consider, for example, the case $b = 4$. Whereas after [CLM00] only $t \leq \frac{n}{2}$ is proven to be tolerable, the bound that is given here states that actually up to $t < \frac{3}{5}n$ corrupted players can be tolerated. Figure 5.1 gives a graphical comparison. A similar gap (though decreasing in function of b) appears for every b.

It is important to note that, in this section, we deviate from the popular notation for arbitrary resilience, $t < n$ (c.f. Section 3.1), and instead write $t < n - 1$ (or $t \leq n - 2$) which describes the largest non-trivial case — implying the achievability for $t \geq n - 1$ as well. The reason is the derived tight bound $t < \frac{b-1}{b+1}n$ that, for the special case $n = b$, satisfies $b - 2 < \frac{b-1}{b+1}b < b - 1$ and thus yields the tight bound $t < b - 1$. Thus, for

notational convenience and without any further consequence, we define arbitrary resilience to be $t < n - 1$.

5.3.2.1 Model

We consider model $\mathcal{M}_{\mathrm{aut}}$ (pairwise authenticated channels) but, additionally, we assume that unconditionally secure, synchronous broadcast channels among each b-tuple of players are available, i.e., that for each subset of b players ($S \subset P$, $|S| = b$) and for any selection of a sender among them there is an authenticated broadcast channel from the sender to the remaining $b - 1$ players. Such broadcast channels from a sender to $b - 1$ recipients will be denoted as *BC_b channels*. The security of the BC_b channels is not necessarily required to be perfectly secure but we assume their error probability ε_0 to be customizable to an arbitrarily small level (as for the special case of BC_3 of the previous section). We will refer to this model as $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$.

5.3.2.2 An IG protocol

In this section, we give a (generally inefficient) IG-protocol for global broadcast in model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$, unconditionally secure against $t < \frac{b-1}{b+1}n$ corrupted players. The protocol is efficient if and only if n and b differ by a constant, i.e., $n - b = O(1)$. Along the lines of Section 4.1.1.1 we start with a protocol for TTBC among player set $S \subseteq P$ ($|S| = n$) with respect to thresholds t_v and t_c.

For any predicate Q, we define $\bigwedge_{k=1}^{0} Q :=$ true; and, in step 4 of the protocol, let "min" denote any minimal set that satisfies the given condition. Furthermore, note that it is a binary broadcast protocol. Thus the recursion in step 7 does not only branch in order of n ($n-1$ sub-calls) but also in order $\log b$ since $\ell_i \in \{0, \ldots, b-1\}$ must be processed bit-wise.

Protocol 5.2 $\mathtt{TTBC}\,(S, p_s, x_s, t_v, t_c)$

1. $\forall S_{b-2} \subseteq S \backslash \{p_s, p_i\},\ |S_{b-2}| = b-2$:
   $v_i^{S_{b-2}} :=$ `Broadcast` $(S_{b-2} \cup \{p_s, p_i\}, p_s, x_s)$ `fi`;
2. `if` $i = s$ `then` $y_i := x_s$; return y_i `fi`;
3. `if` $t_c = 0 \vee b = n$ `then` $y_i := v_i^*$; return y_i `fi`; ["any received value"]
4. $\forall z \in \{0,1\}$: `if` $\exists S_{b-2} : v_i^{S_{b-2}} = z$ `then`
   $T_i[z] := \min(T \subseteq S \backslash \{p_s, p_i\} | \forall S_{b-2} \supseteq T :\ v_i^{S_{b-2}} = z)$;
   `else` $T_i[z] := S$ `fi`; [assign a sentinel]

5. `if` $|T_i[0]| < |T_i[1]|$ `then` $v_i := 0$ `else` $v_i := 1$ `fi`;
6. `if` $v_i = 0$ `then` $\ell_i := \min(|T_i[v_i]|, \lfloor \frac{b-1}{2} \rfloor)$
   `else` $\ell_i := b - 1 - \min(|T_i[v_i]|, \lfloor \frac{b-1}{2} \rfloor)$ `fi`;
7. $\forall p_j \in S \setminus \{p_s\}$: $\ell_i^j :=$ `TTBC` $(S \setminus \{p_s\}, p_j, \ell_j, t_v, t_c - 1)$ `fi`;
8. $\forall \ell \in [0, b-1] : L_i[\ell] := \big|\{p_j \in S \setminus \{p_s\} \mid \ell_i^j = \ell\}\big|$;
9. `if` $\bigwedge_{k=1}^{\ell_i}(L_i[k-1] + L_i[k] \geq n - t_c) \wedge (L_i[0] \geq n - t_v - 1)$ `then`
10. $y_i := 0$ `else` $y_i := 1$
11. `fi`; return y_i

Lemma 5.6. *Consider Protocol 5.2 in model* $\mathcal{M}_{\text{aut}}^{\text{bc}_b}$ *with* $n > b$*. For every correct player* p_i *it holds that* $|T_i[0]| + |T_i[1]| \geq b - 1$.

Proof. Let p_i be a correct player and let $k_0 := |T_i[0]|$ and $k_1 := |T_i[1]|$. Hence, there are k_0 distinct players $q_1, \ldots, q_{k_0} \notin \{p_s, p_i\}$ such that all instances of BC_b in step 1 involving the players in $\{p_s, p_i, q_1, \ldots, q_{k_0}\}$ resulted in output value $v_i = 0$.

Now, assume that $|T_i[0]| + |T_i[1]| < b - 1$ and consider any k_1 distinct players $r_1, \ldots, r_{k_1} \notin \{p_s, p_i\}$ and the set $U := \{p_s, p_i\} \cup \{q_1, \ldots, q_{k_0}\} \cup \{r_1, \ldots, r_{k_1}\}$. Since $|U| \leq b$, there is at least one instance of BC_b involving all the players in U. This instance cannot have resulted in $v_i = 0$ and $v_i = 1$ at the same time, and thus the lemma follows. □

Lemma 5.7. *Consider Protocol 5.2 in model* $\mathcal{M}_{\text{aut}}^{\text{bc}_b}$ *with* $n > b$*, and consider any two correct players* p_i *and* p_j*. For any* $v_i \in \{0, 1\}$, $|T_i[v_i]| \leq b - 3$ *implies* $|T_j[v_i]| \leq |T_i[v_i]| + 1$.

Proof. Assume that $|T_i[v_i]| \leq b - 3$. Hence, there are $k \leq b - 3$ distinct players $q_1, \ldots, q_k \notin \{p_s, p_i\}$ such that all instances of BC_b in step 1 involving the at most $b - 1$ players in $\{p_s, p_i, q_1, \ldots, q_k\}$ resulted in output value v_i, in particular, all instances involving the at most b players in $\{p_s, p_i, p_j, q_1, \ldots, q_k\}$. Hence, all instances of BC_b involving the players in $\{p_s, p_j, p_i, q_1, \ldots, q_k\}$ resulted in output value v_i, and thus $|T_j[v_i]| \leq k+1$, and the lemma follows. □

Lemma 5.8. *Consider Protocol 5.2 in model* $\mathcal{M}_{\text{aut}}^{\text{bc}_b}$ *with* $n > b$*. If* p_i *is correct then, for some* $c \in \{-1, 1\}$*, every correct player* p_j *computes* $\ell_j \in \{\ell_i, \ell_i + c\}$.

Proof. Consider a correct player p_i with a minimal set $T_i[v_i]$, i.e., such that for all $v_j \in \{0, 1\}$ and all correct players p_j it holds that $|T_j[v_j]| \geq |T_i[v_i]|$.

If $|T_i[v_i]| \geq \lfloor \frac{b-1}{2} \rfloor$ then, by the minimality of $|T_i[v_i]|$, every correct player p_j computes $\ell_j \in \{\lfloor \frac{b-1}{2} \rfloor, b-1-\lfloor \frac{b-1}{2} \rfloor\} = \{\lfloor \frac{b-1}{2} \rfloor, \lceil \frac{b-1}{2} \rceil\}$, and the lemma follows.

If $|T_i[v_i]| < \lfloor \frac{b-1}{2} \rfloor$ then, by Lemma 5.7, it holds that $|T_j[v_i]| \leq |T_i[v_i]| + 1 \leq \lfloor \frac{b-1}{2} \rfloor$. Furthermore, by Lemma 5.6, it holds that $|T_j[1-v_i]| \geq b-1-|T_j[v_i]|$.

We distinguish three cases:

- If $|T_j[v_i]| < \lfloor \frac{b-1}{2} \rfloor$ then $|T_j[1-v_i]| \geq b-1-|T_j[v_i]| > b-1-\lfloor \frac{b-1}{2} \rfloor = \lceil \frac{b-1}{2} \rceil > |T_j[v_i]|$.
- If $|T_j[v_i]| = \lfloor \frac{b-1}{2} \rfloor$ and b is even then $|T_j[1-v_i]| \geq b-1-|T_j[v_i]| = \lceil \frac{b-1}{2} \rceil > \lfloor \frac{b-1}{2} \rfloor = |T_j[v_i]|$.
- If $|T_j[v_i]| = \lfloor \frac{b-1}{2} \rfloor$ and b is odd then $|T_j[1-v_i]| \geq b-1-|T_j[v_i]| = \lceil \frac{b-1}{2} \rceil = \lfloor \frac{b-1}{2} \rfloor = |T_j[v_i]|$.

The first two cases imply that $v_j = v_i$ since $|T_j[v_i]| < |T_j[1-v_i]|$ (see step 5). Furthermore, by Lemma 5.7, it holds that $|T_j[v_i]| \leq |T_i[v_i]| + 1$ and thus that p_j computes $\ell_j \in \{\ell_i, \ell_i + 1\}$ if $v_i = 0$ and $\ell_j \in \{\ell_i, \ell_i - 1\}$ if $v_i = 1$.

The third case implies that $|T_j[v_i]| = |T_j[1-v_i]| = \lfloor \frac{b-1}{2} \rfloor$. Since b is odd, it holds that $\lfloor \frac{b-1}{2} \rfloor = b-1-\lfloor \frac{b-1}{2} \rfloor$, and thus that p_j computes $\ell_j = \ell_i + 1$ if $v_i = 0$ and $\ell_j = \ell_i - 1$ if $v_i = 1$. □

Lemma 5.9. *Consider Protocol 5.2 in model $\mathcal{M}^{\mathrm{bc}_b}_{\mathrm{aut}}$. If $(b-1)n > (b-1)t_c + 2t_v$ and $t_c \leq t_v$ then the protocol achieves TTBC with respect to thresholds t_v and t_c.*

Proof. The proof proceeds by backward induction over n. Thus, assume that Protocol 5.2 achieves TTBC among $n' = n-1$ players whenever $(b-1)n' > (b-1)t'_c + 2t'_v$, and hence, achieves TTBC for the special case that $n' = n-1$, $t'_v = t_v$, and $t'_c = t_c - 1$, since $(b-1)n > (b-1)t_c + 2t_v$ implies $(b-1)n' = (b-1)(n-1) > (b-1)(t_c-1) + 2t_v = (b-1)t'_c + 2t'_v$.

Validity: Assume that at most t_v players are corrupted and that the sender p_s is correct. If $t_c = 0$ or $b = n$ then validity is trivially satisfied (step 3) since p_s distributes the same value x_s with all BC_b channels — this case constitutes the induction base. Thus, assume that $t_c > 0$, $b < n$, and by induction, that the protocol achieves validity with respect to $n' = n-1$, $t'_v = t_v$, and $t'_c = t_c - 1$.

Since correct p_s distributes the same value x_s with all BC_b channels, all correct players p_i compute $T_i[x_s] = \emptyset$ and thus $\ell_i = x_s \cdot (b-1)$. By assumption, in step 7, every correct player receives this ℓ_i from every one of the at least $n - t_v - 1$ remaining correct players in $S \setminus \{p_s\}$.

If $x_s = 0$ then every correct player p_i computes $L_i[0] \geq n - t_v - 1$ and thus $y_i = 0 = x_s$. If $x_s = 1$ then, since $(b-1)n > (b-1)t_c + 2t_v$ implies $\frac{(b-1)(n-t_c)}{2} > t_v$, $\bigwedge_{k=1}^{\ell_i}(L_i[k-1]+L_i[k] \geq n-t_c) \wedge (L_i[0] \geq n-t_v-1)$ would imply that $\sum_{\ell=0}^{b-1} L_i[\ell] \geq \frac{(b-1)(n-t_c)}{2} + n - t_v - 1 > t_v + n - t_v - 1 > n - 1$ in contradiction to the fact that $n' = n - 1$, and thus it must hold that $\sum_{\ell=0}^{b-1} L_i[\ell] \leq n - 1$, and every correct player p_i computes $y_i = 1 = x_s$.

Consistency: Assume that at most t_c players are corrupted. The induction base consists of the case that either $b = n$ (then the protocol achieves broadcast by definition) or that the sender is correct (then, since $t_v \geq t_c$, consistency directly follows from the validity property proven above — especially, this is the case if $t_c = 0$).

Thus, assume that the sender p_s is corrupted (and hence $t_c > 0$), $b < n$, and by induction, that the protocol achieves TTBC with respect to $n' = n - 1$, $t'_v = t_v$, and $t'_c = t_c - 1$.

Since the sender is corrupted, at most $t_c - 1 = t'_c$ corrupted players remain in $S \setminus \{p_s\}$, and are involved in step 7. Hence, by induction, every invocation of the protocol in step 7 achieves consistency. Furthermore, since $t'_v \geq t'_c$, also validity is achieved, i.e., all invocations of the protocol in step 7 achieve broadcast. This implies that two correct players p_i and p_j compute exactly the same sets $L_i[0] = L_j[0] =: L[0], \ldots, L_i[b-1] = L_j[b-1] =: L[b-1]$.

Let p_i be a correct player with minimal ℓ-value, i.e., such that for all other correct players p_j: $\ell_i \leq \ell_j$. By Lemma 5.8, it holds that $\ell_j \in \{\ell_i, \ell_i + 1\}$. We now show that all correct players p_j compute $y_j = y_i$. If $\ell_j = \ell_i$ then both players have exactly the same view and hence decide in the same way, $y_j = y_i$. Thus, suppose that $\ell_j = \ell_i + 1$.

- If p_i computes $y_i = 0$ then $\bigwedge_{k=1}^{\ell_i}(L[k-1] + L[k] \geq n - t_c) \wedge (L[0] \geq n - t_v - 1)$, and by Lemma 5.8 it also holds that $L[\ell_i] + L[\ell_i + 1] \geq n - t_c$. Hence, $\bigwedge_{k=1}^{\ell_i+1}(L_{k-1} + L_k \geq n - t_c) \wedge (L_i[0] \geq n - t_v - 1)$ and p_j computes $y_j = 0 = y_i$.
- If p_i computes $y_i = 1$ then $\neg \bigwedge_{k=1}^{\ell_i}(L[k-1]+L[k] \geq n-t_c) \vee \neg(L[0] \geq n - t_v - 1)$, and thus $\neg \bigwedge_{k=1}^{\ell_i+1}(L[k-1] + L[k] \geq n - t_c) \vee \neg(L[0] \geq n - t_v - 1)$, and p_j computes $y_j = 1 = y_i$. □

Protocol 5.3 `Broadcast` (P, p_s, x_s)

1. $y_i := TTBC\,(P, p_s, x_s, t, t)$;
2. return y_i

Theorem 5.10. *In model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$, Protocol 5.3 achieves broadcast perfectly secure against $t < \frac{b-1}{b+1}n$ corrupted players. Its round complexity is $\mathcal{R} = \min(t+1, n-b+1)$ and its computational and bit complexities are polynomial in n if $n - b = O(1)$.*

Proof. That Protocol 5.3 achieves broadcast follows from Definition 4.1, Lemma 5.9, and the fact that $t_v = t_c = t$ and thus $(b-1)n > (b-1)t_c + 2t_v = (b+1)t$.

The round complexity can be easily verified by code inspection. The bit complexity satisfies $\mathcal{B} \leq \sum_{i=0}^{\min(t,n-b)} \binom{n-1-i}{b-1} \frac{(n-1)!}{(n-1-i)!} \lceil \log_2 b \rceil^i$. Given that $n - b = O(1)$ we get

$$\mathcal{B} \leq \sum_{i=0}^{n-b} n^{n-b-i} n^i \lceil \log_2 b \rceil^i \leq n^{n-b} \log_2^{n-b} n = \mathrm{Poly}(n).$$

Furthermore, if $n - b = O(1)$ then the sets $T_i[0]$ and $T_i[1]$ can be efficiently computed by a breadth-first-search technique, and thus also the computational complexity of the protocol is polynomial for this case. □

5.3.3 Impossibility result

The following theorem, generalizing Theorem 4.16, states that global broadcast among n players secure against $t \geq \frac{b-1}{b+1}n$ corrupted players is impossible in model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{bc}_b}$ (or model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}_b}$ where *secure* bilateral channels are given). Recall that, for notational ease, we use the bound $t < n-1$ for arbitrary resilience.

Theorem 5.11 ([FGMO02]). *Let $b \geq 2$. In model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}_b}$, broadcast among a set of $n > b$ players $P = \{p_0, \ldots, p_{n-1}\}$ is not achievable if $t \geq \lceil \frac{b-1}{b+1}\, n \rceil$. For every protocol there exists a value $x_0 \in \{0,1\}$ such that, when the sender holds input x_0, the adversary can make the protocol fail*

- *with a probability of at least $\frac{1}{2b+2}$ if she is computationally bounded, and*
- *with a probability of at least $\frac{1}{b+1}$ if she is computationally unbounded.*

Proof. For the sake of contradiction we assume that, in model $\mathcal{M}_{\mathrm{sec}}^{\mathrm{bc}_b}$ for some $b \geq 2$, there is a broadcast protocol Ψ among a player set $P =$

$\{p_0, \ldots, p_{n-1}\}$ that tolerates $t \geq \lceil \frac{b-1}{b+1} n \rceil$. This protocol is used to build a different system with contradictory behavior.

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs and let $\Pi_0 \dot{\cup} \Pi_1 \dot{\cup} \ldots \dot{\cup} \Pi_b = \Pi$ be a partition of Π into $b+1$ sets of cardinalities $|\Pi_{k_i}| \in \{\lfloor \frac{n}{b+1} \rfloor, \lceil \frac{n}{b+1} \rceil\}$. In other words, $\ell = n \bmod (b+1)$ sets of cardinality $\lceil \frac{n}{b+1} \rceil$ and $b+1-\ell$ sets of cardinality $\lfloor \frac{n}{b+1} \rfloor$ with the following property: if $\ell \geq \frac{b+1}{2}$ then, of any two adjacent sets Π_k and $\Pi_{(k+1) \bmod (b+1)}$, at least one is of cardinality $\lceil \frac{n}{b+1} \rceil$. Furthermore, for each $i \in \{0, \ldots, n-1\}$, let π_{i+n} be an identical copy of processor π_i. Finally, for each $k \in \{0, \ldots, b\}$, let $\Pi_{k+n} = \{\pi_{i+n} \mid \pi_i \in \Pi_k\}$ form an identical copy of set Π_k.

Instead of connecting the original processors as required for broadcast, we build a network involving all $2n$ processors by connecting them with instances of BC_b such that for every set $\Pi_k \cup \Pi_{(k+1) \bmod (2b+2)}$ of processors in the new system, and without the presence of an adversary, their common view is indistinguishable from their view as processors in $\Pi_{k \bmod (b+1)} \cup \Pi_{(k+1) \bmod (b+1)}$ in the original system with an adversary that corrupts the processors in $\Pi \setminus (\Pi_{k \bmod (b+1)} \cup \Pi_{(k+1) \bmod (b+1)})$ in an admissible way.

Note that for every BC_b in the original system there is a set Π_k such that no processor in Π_k is connected to it. This is because there are $b+1$ different sets Π_k. The BC_b's are now reconnected in the following way: For each BC_b that originally connects a set S of b processors in $\Pi \setminus \Pi_k$ ($k \in \{0, \ldots, b\}$), there are now two BC_b's, one connecting the processors $\{\pi_i \in \Pi_{k+1} \cup \cdots \cup \Pi_{k+n-1} \mid \pi_{i \bmod (b+1)} \in S\}$ and one connecting the processors $\{\pi_i \in \Pi_{k+1+n} \cup \cdots \cup \Pi_{k+n-1+n} \mid \pi_{i \bmod (b+1)} \in S\}$.

We now show that every set $\Pi_k \cup \Pi_{(k+1) \bmod (2b+2)}$ of processors contains at least $n-t$ correct processors, implying that protocol Ψ satisfies the broadcast conditions with respect to all processors in the union of two such adjacent processor sets. We have two cases:

$$\ell < \tfrac{b+1}{2} : |\Pi_k \cup \Pi_{(k+1) \bmod (2b+2)}| \geq 2 \left\lfloor \tfrac{n}{b+1} \right\rfloor = \left\lfloor \tfrac{2n}{b+1} \right\rfloor = n + \left\lfloor \tfrac{2n-(b+1)n}{b+1} \right\rfloor$$
$$= n - \left\lfloor \tfrac{b-1}{b+1} n \right\rfloor \geq n - \left\lceil \tfrac{b-1}{b+1} n \right\rceil \geq n - t \text{ , and}$$

$$\ell \geq \tfrac{b+1}{2} : |\Pi_k \cup \Pi_{(k+1) \bmod (2b+2)}| \geq \left\lfloor \tfrac{n}{b+1} \right\rfloor + \left\lceil \tfrac{n}{b+1} \right\rceil \geq \left\lfloor \tfrac{2n}{b+1} \right\rfloor \geq n - t \text{ .}$$

Let now π_0 and π_n be initialized with different inputs. For any possible run of the new system on inputs x_0 and $x_n = 1 - x_0$ it holds that, chosen a pair $(\Pi_k, \Pi_{(k+1) \bmod (2b+2)})$ of adjacent processor sets uniformly at

random, the probability that the conditions for broadcast are violated for this pair is at least $\frac{1}{b+1}$.

In particular, there is a pair $(\Pi_k, \Pi_{(k+1) \bmod (2b+2)})$ in the new system such that, over all possible runs on inputs $x_0 = 0$ and $x_n = 1$ the probability that the conditions of broadcast are violated for $(\Pi_k, \Pi_{(k+1) \bmod (2b+2)})$ is at least $\frac{1}{b+1}$.

If the adversary is unbounded, given any protocol Ψ, she can compute such a pair $(\Pi_k, \Pi_{(k+1) \bmod (2b+2)})$ and act accordingly by corrupting the processors in $\Pi_{(k+2) \bmod (b+1)} \cup \cdots \cup \Pi_{(k+b) \bmod (b+1)}$ in the original system, hence forcing the protocol to fail on input

$$x_0 = \begin{cases} 0 & \text{, if } 0 \in \{k, k+1\} \text{ , and} \\ 1 & \text{, else ,} \end{cases}$$

with a probability of at least $\frac{1}{b+1}$.

If the adversary is computationally bounded then she can still make the protocol fail with a probability of at least $\frac{1}{2b+2}$ as follows along the lines of the proof of Theorem 4.16. □

5.3.4 A hierarchy of partial consistency

Recall the tight bound for global broadcast when given broadcast among b players: $t < \frac{b-1}{b+1}n$. The constructions in Section 5.3.2 relied on the fact that BC_b allows to distribute a value such that, even when the sender is corrupted, all correct players' views are "adjacent" (Lemma 5.8). More precisely, by having the sender distribute his input bit by all BC_b channels he is involved in, a protocol was achieved that solves the following problem.

Definition 5.1. *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and $b > 0$ an integer. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \{0, 1\}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \{0, \ldots, b-1\}$ achieves b-*set-neighboring *(N_n^b, for short) with respect to P and p_s if it satisfies the following conditions:*

- *Validity: If the sender is correct with input $x_s \in \{0, 1\}$ then every correct player p_i decides on $y_i = x_s \cdot (b-1)$.*
- *Consistency: There is a value $\ell \in \{0, \ldots, b-2\}$ such that every correct player p_i decides on either $y_i = \ell$ or $y_i = \ell + 1$.*

◇

N_n^b with respect to $b = 2$ achieves simply the same as the multi-send of a bit: if the sender is correct then all players decide on the sender's

input value and, trivially, all correct players always decide on either 0 or 1. Recall Definition 3.4 of weak broadcast. Substituting the recipients' output range with $0 \mapsto 0$, $\perp \mapsto 1$ and $1 \mapsto 2$ exactly yields Definition 5.1 with respect to $b = 3$. Recall Definition 3.6 of graded broadcast. Substituting the recipients' output range with $(y_i = 0, g_i = 1) \mapsto 0$, $(0,0) \mapsto 1$, $(1,0) \mapsto 2$, and $(1,1) \mapsto 3$ exactly yields Definition 5.1 with respect to $b = 4$. Finally, the original version of graded broadcast in [FM97] yields Definition 5.1 with respect to $b = 5$. Thus, N_n^b is simply a generalization along the lines of these four well-known special cases. In [Rot00], von Rotz already observed that b-set-neighboring implies broadcast among $n = b$ players tolerating any number of corrupted players, $t < n$.

Proposition 5.12 ([Rot00]). *For any $b \geq 2$, BC_b and N_b^b are equivalent.*

Proof.
$BC_b \Rightarrow N_b^b$*:* N_b^b can be achieved from BC_b by having the sender broadcast his bit and having the recipients decide on 0 on receiving 0, and on $b-1$, otherwise.

$N_b^b \Rightarrow BC_b$*:* The proof proceeds by induction. First, note the trivial fact that N_b^b ($b \geq 2$) implies $\mathrm{N}_{b'}^{b'}$ for any b', $1 \leq b' \leq b$, since N_{b-1}^{b-1} can be obtained from N_b^b by removing a player and mapping both output values $y_i \in \{b-2, b-1\}$ to $y_i := b-2$.

That N_2^2 implies BC_2 is the induction base (both functionalities achieve exactly the same). We now show how to achieve BC_b from BC_{b-1} and N_b^b. First, the b players perform an instance of N_b^b where the sender p_s inputs his value $x_s \in \{0,1\}$ to be broadcast, decides on $y_s := x_s$, and terminates. Second, all $b-1$ remaining players p_i broadcast among each other their received values $y_i \in \{0, \ldots, b-1\}$ with help of BC_{b-1} (in a bitwise manner). Finally, a player p_i decides on 0 if every value $v \in [0, y_i]$ was broadcast by at least one of the $b-1$ players $p_j \neq p_s$ — and on 1, otherwise. The correctness of this construction follows from the facts

- that all correct players $p_i \neq p_s$ end up with the same $b-1$ values y_j ($j \neq s$) since these values are redistributed by broadcast, and
- that at least one of the b values in $\{0, \ldots, b-1\}$ is not redistributed, and
- that, for any two correct players p_i and p_j, it holds that $y_i = y_j \pm 1$.

□

Theorem 5.13. *For any $b \geq 2$, N_k^b ($b \leq k \leq n$) implies global broadcast if and only if $t < \frac{b-1}{b+1} n$.*

b	Primitive			Full Res.		General n
2	Multi-Send:	N_n^2	$\Leftrightarrow$	BC_2	$\Leftrightarrow$	$t < n/3$
3	Weak Broadcast (WBC):	N_n^3	$\Leftrightarrow$	BC_3	$\Leftrightarrow$	$t < n/2$
4	Graded Broadcast (GBC):	N_n^4	$\Leftrightarrow$	BC_4	$\Leftrightarrow$	$t < 3n/5$
5	[FM97]-GBC:	N_n^5	$\Leftrightarrow$	BC_5	$\Leftrightarrow$	$t < 2n/3$
$\vdots$		$\vdots$				
$n-1$	-	N_n^{n-1}	$\Leftrightarrow$	BC_{n-1}	$\Leftrightarrow$	$t < n-2$
n	-	N_n^n	$\Leftrightarrow$	BC_n	$\Leftrightarrow$	$t < n$

Table 5.1: *Hierarchy of partial consistency; partial broadcast with full resilience, and global broadcast with resilience t.*

Proof. From Lemma 5.8 it follows that BC_b implies N_n^b with arbitrary resilience. N_n^b trivially implies N_k^b ($b \leq k \leq n$) as is mentioned in the proof of Proposition 5.12. By Proposition 5.12, N_b^b implies BC_b. Finally, by Theorems 5.10 and 5.11, BC_b implies global broadcast if and only if $t < \frac{b-1}{b+1}n$. □

In particular, the observations made in this section imply the well-known fact that multi-send for $t < n$ is achievable if and only if pairwise authenticated channels are achievable and, together with Theorem 4.1, that multi-send implies global broadcast with resilience $t < n/3$. Weak broadcast for $t < n$ is achievable if and only if BC_3 is achievable and, together with Theorem 5.10, weak broadcast implies global broadcast with resilience $t < n/2$ (as separately stated by Theorem 4.14). Graded broadcast for $t < n$ is achievable if and only if BC_4 is achievable and, together with Theorem 5.10, graded broadcast implies global broadcast with resilience $t < 3n/5$. See Table 5.1 for the full picture.

5.4 External information sources

It can be easily proven that an additional global random source (i.e., a beacon) does not help to improve the classical bound of $t < n/3$ for broadcast in standard models $\mathcal{M}_{\mathrm{aut}}$ or $\mathcal{M}_{\mathrm{sec}}$, namely by simply assuming such a source in the proof of Theorem 4.16 — still with exactly the same reasoning. However, by slightly modifying the functionality of such a random source, as described in the following section, it does. Note that a straightforward solution could be achieved with the help of an external information source that simulates the whole protocol in [BPW91]

or [PW96] in order to establish broadcast for the future (cf. Section 4.3.1). However, this would be a rather complex task to be performed by an information source requiring a lot of mathematical structure. In contrast, the following solution is based on very simple correlated information.

5.4.1 The Q-flip model

We assume model $\mathcal{M}_{\mathrm{aut}}$ (pairwise authenticated channels). But on top, some additional primitive is assumed among each triple of players, called *Q-Flip*. This model is denoted as model $\mathcal{M}_{\mathrm{aut}}^{\mathrm{q}}$.

Q-Flip, as described for the three players p_1, p_2, and p_3, is a random generator that (for every invocation), with uniform distribution, generates a random permutation on the elements $\{0,1,2\}$, $(x_1,x_2,x_3) \in \{(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)\}$, and sends element x_i ($i \in \{1,2,3\}$) to player p_i. No single player p_i learns more about the permutation than the value x_i which he receives, i.e., a single player does not learn how the remaining two values are assigned to the other players.

The Q-Flip primitive was originally motivated by quantum entanglement considerations about the Byzantine agreement problem. A detailed description of these quantum physical aspects is given in [FGM01].

5.4.2 Efficient protocol

We construct an efficient protocol for broadcast that tolerates $t < n/2$ corrupted players. As follows from Lemma 5.5, it is sufficient to show that $\mathrm{WBC}_{3,1}$ can be efficiently achieved among every triplet of players, i.e., weak broadcast among three, secure against one corrupted player. We now describe such a protocol for a sender s and two recipients r_0 and r_1.

Let $x_s \in \{0,1\}$ be the input of sender s, and y_0 and y_1 be the final outputs of the players r_0 and r_1 (whereas s always implicitly decides on $y_s = x_s$). The primitive Q-Flip is invoked some m times and each player receives a sequence of m elements in $\{0,1,2\}$, i.e., s receives $Q_s = (Q_s[1],\ldots,Q_s[m])$, r_0 receives $Q_0 = (Q_0[1],\ldots,Q_0[m])$, and r_1 receives $Q_1 = (Q_1[1],\ldots,Q_1[m])$, where the triplet $(Q_s[i],Q_0[i],Q_1[i])$ represents the outcome of the i-th invocation of Q-Flip. The protocol now proceeds as follows:

First, s sends to r_0 and r_1 his input bit x_s and the set σ_s of all indices $i \in \{1,\ldots,m\}$ such that s received the complement of x_s for the i-th

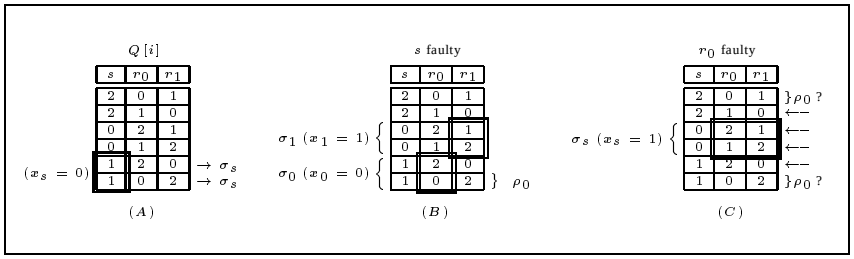

Figure 5.2: *(A) Possible outcomes of Q-Flip and selection of σ_s; (B,C) Basic cheating strategies for s and r_0.*

invocation of Q-Flip (see Figure 5.2–A):

$$\sigma_s = \{i \in \{1, \ldots, m\} : Q_s[i] = 1 - x_s\} \ . \tag{5.1}$$

If this is done correctly then σ_s is of large size (i.e., approximately $m/3$, which is important for good statistics on the corresponding Q-Flips) and both recipients r_k ($k \in \{0,1\}$) never received the value $1 - x_s$ for any Q-Flip invocation with respect to σ_s: $\{i \in \sigma_s : Q_k[i] = 1 - x_s\} = \emptyset$.

Let x_k and σ_k ($k \in \{0,1\}$) be the information that (potentially faulty) s actually sent to recipient r_k. The recipients now decide on $y_k = x_k$ if and only if σ_k is of large size and the value $1 - x_k$ was never received with respect to σ_k:

$$y_k := \begin{cases} x_k & \text{, if } (\sigma_k \text{ large}) \ \wedge \ (\{i \in \sigma_k : Q_k[i] = 1 - x_k\} = \emptyset) \\ \perp & \text{, else .} \end{cases} \tag{5.2}$$

Now, r_0 sends to r_1 his value y_0 and the set ρ_0 of all indices $i \in \sigma_0$ such that he received x_0 for the corresponding Q-Flip invocation:

$$\rho_0 = \{i \in \sigma_0 : Q_0[i] = x_0\} \ . \tag{5.3}$$

If $y_0 \neq y_1$ but $y_0, y_1 \in \{0,1\}$, r_1 now redecides in the following way:

$$y_1 := \begin{cases} y_0 & \text{, if } (\rho_0 \text{ large}) \ \wedge \ (\{i \in \rho_0 \setminus \sigma_1 : Q_1[i] = 2\} \stackrel{\text{almost}}{=} \rho_0) \\ y_1 & \text{, else .} \end{cases} \tag{5.4}$$

Before describing the protocol in detail, we give a rough argumentation why none of the players can make the protocol fail.

r_1 faulty: r_1 cannot significantly misbehave since r_1 is silent.

s faulty: In order to make the protocol fail, s must achieve that r_0 and r_1 decide on distinct values $y_0 \neq y_1$ such that $y_0, y_1 \in \{0,1\}$ (Equation (5.2)) and that r_1 does not redecide on $y_1 = y_0$ according to Equation (5.4). The only way to achieve this is to select $x_0 \in \{0,1\}$ and $x_1 = 1 - x_0$, and to basically compose large sets σ_0 and σ_1 as shown in Figure 5.2–B (as shown with respect to $x_0 = 0$)[11]. But then, r_0 learns a large set ρ_0 of indices i (Figure 5.2–B, last row) such that, mainly, $i \notin \sigma_1$ and $Q_1[i] = 2$ which will "convince" r_1 to redecide on $y_1 = y_0$ according to Equation (5.4).

r_0 faulty: In order to make the protocol fail, r_0 must achieve that r_1 redecides on $y_1 = y_0 \neq x_s$ according to Equation (5.4). Since correct s sends $\sigma_s = \{i \in \{1,\ldots,m\} : Q_s[i] = x_s\}$ to both players, r_0 cannot come up with a large set ρ_0 of indices i such that most of them satisfy $i \notin \sigma_s$ and $Q_1[i] = 2$ (see Figure 5.2–C) since r_0 cannot distinguish between the outcomes corresponding to the first and the last row.[12]

From this argumentation, it can be seen that the Q-Flip primitive is simply used in order to set up a pseudo-signature scheme with transferability 2.

Protocol 5.4 `WeakBroadcast`$(\{s, r_0, r_1\}, s, x_s)$

1. s, r_0, r_1: invoke primitive Q-Flip m times;
2. $s \overset{\text{send}}{\rightarrow} r_0, r_1 \colon x_s, \sigma_s = \{i \in \{1,\ldots,m\} : Q_s[i] = 1 - x_s\}$; $\quad r_k$: `Receive`(x_k, σ_k)
   $s \colon y_s := x_s$;
3. r_k: `if` $(|\{i \in \sigma_k : Q_k[i] = x_k\}| \geq m_0) \wedge (\{i \in \sigma_k : Q_k[i] = 1 - x_k\} = \emptyset)$ `then` $y_k := x_k$ `else` $y_k := \perp$ `fi`;
4. $r_0 \overset{\text{send}}{\rightarrow} r_1 \colon y_0,\ \rho_0 = \{i \in \sigma_0 : Q_0[i] = y_0\}$; $\quad r_1$: `Receive`(y_{01}, ρ_{01})
5. r_1: `if` $(\perp \neq y_{01} \neq y_1 \neq \perp) \wedge (|\rho_{01}| \geq m_0) \wedge$
   $(|\{i \in \rho_{01} \setminus \sigma_1 : Q_1[i] = 2\}| \geq \lambda |\rho_{01}|)$ `then` $y_1 := y_{01}$ `fi`

The detailed protocol is described by Protocol 5.4. There, two free protocol parameters are introduced, m_0 ($m_0 = \Omega(m)$; $m_0 < m/6$) for

[11] Note that for every selection $i \in \sigma_k$ ($k \in \{0,1\}$) such that $Q_s[i] \neq 1 - x_k$, it holds with a probability of $\frac{1}{2}$ that $Q_k[i] = 1 - x_k$, which makes r_k decide on $\perp$ (or $1 - x_k$) according to Equation (5.2).

[12] Note, that r_0 completely learns all instances indicated by an arrow but nothing more about the other instances.

asserting that the sets σ_s and ρ_0 are of sufficiently large cardinality, and λ ($\frac{1}{2} < \lambda < 1$) for the test according to Equation (5.4). Both parameters will be fixed for the final analysis of the protocol.

Lemma 5.14. *Let $\lambda_0 < 1$ and $\lambda_1 > 1$. The probability Pr_{stats} that of m invocations of Q-Flip any one of the six possible outcomes in $\{(0,1,2),\ldots,(2,1,0)\}$ occurs either less than $m_0 = \lambda_0 \frac{m}{6} < \frac{m}{6}$ or more than $m_1 = \lambda_1 m > \frac{m}{6}$ times satisfies $Pr_{stats} \leq 6 \max(\mathcal{C}_\downarrow(\frac{1}{6}, m, \lambda_0), \mathcal{C}_\uparrow(\frac{1}{6}, m, \lambda_1))$.*[13]

Proof. The respective probability for each particular outcome, e.g. $(0,1,2)$, can be independently estimated by the Chernoff bound (with random variable X_i representing the i-th Q-Flip). The overall probability is at most as large as the sum of these six probabilities (union bound). □

Lemma 5.15 (All players correct). *If all players are correct and each possible outcome of Q-Flip appears at least m_0 times, then Protocol 5.4 achieves $WBC_{3,1}$.*

Proof. By the given assumptions we have $\sigma_s = \sigma_0 = \sigma_1$ and $x_s = x_0 = x_1$ after step 1 of the protocol. Since $\sigma_s = \{i \in \{1,\ldots,m\} : Q_s[i] = 1 - x_s\}$, it holds for both recipients r_k that

$$\{i \in \sigma_k : Q_k[i] = 1 - x_k\} = \emptyset \ \wedge \ |\{i \in \sigma_k : Q_k[i] = x_k\}| \geq m_0 \ .$$

Hence $y_0 = y_1 = x_s$ after step 3, and thus also at the end of the protocol. □

Since r_1 does not actively participate in the protocol, i.e., since he does not send a single message, we immediately get

Corollary 5.16 (r_1 possibly faulty). *If the players s and r_0 are correct and each possible outcome of Q-Flip appears at least m_0 times, then Protocol 5.4 achieves $WBC_{3,1}$.*

It now remains to determine upper bounds on the error probabilities for the cases that s is faulty ($\Pr_s$) or that r_0 is faulty ($\Pr_{r_0}$). The following lemma will be used for the analysis of the former case in the proof of Lemma 5.18.

Lemma 5.17. *If each possible outcome of Q-Flip appears at least m_0 times and at most m_1 times and if (faulty) sender s submits $x_0 \in \{0,1\}$ to r_0 and $x_1 = 1 - x_0$ to r_1 and selects k indices $i \in \{1,\ldots,m\}$ such that either*

- *$i \in \sigma_0 \cap \sigma_1$, or*

[13] See Section 2.5 for the definition of $\mathcal{C}_\downarrow$ and $\mathcal{C}_\uparrow$.

- $i \in \sigma_0 \setminus \sigma_1$ *such that* $Q_s[i] = 2$

then $y_0 = x_0$ *and* $y_1 = 1 - x_0$ *(i.e., disagreement) holds at the end of the protocol with a probability of at most* $(\frac{m_1}{m_0+m_1})^k$.

Proof. If s submits one single index $i \in \sigma_0 \cap \sigma_1$ then either $Q_0[i] = 1 - x_0$ or $Q_1[i] = 1 - x_1$ with a probability of at least $\frac{m_0}{m_0+m_1}$ (since the only information by s is $Q_s[i]$ and hence s risks, with the respective probability, to produce a "collision" on either recipient's side which makes that one decide on $\perp$).

On the other hand, if s submits one single index $i \in \sigma_0 \setminus \sigma_1$ such that $Q_s[i] = 2$ (the only possibility in order to achieve that $Q_0[i] = x_0$ and $Q_1[i] \neq 2$) then r_0 decides on $\perp$ with a probability of at least $\frac{m_0}{m_0+m_1}$ (since s risks, with the respective probability, to produce a "collision" on r_0's side which makes that one decide on $\perp$).

Finally, in order to achieve that $y_0 = x_0$ and $y_1 = 1 - x_0$ holds at the end of the protocol, s must prevent any single collision for all k index selections. This can be achieved with a probability of at most $(\frac{m_1}{m_0+m_1})^k$. □

Lemma 5.18 (s possibly faulty). *If the players* r_0 *and* r_1 *are correct and each possible outcome of Q-Flip appears at least* m_0 *times and at most* m_1 *times, then Protocol 5.4 fails to achieve* $WBC_{3,1}$ *with probability at most* $Pr_s < \left(\frac{m_1}{m_0+m_1}\right)^{(1-\lambda)m_0}$.

Proof. The only way for s to make the protocol fail is to force the recipients to decide on distinct bits, i.e., $y_0 = x_0 = b \in \{0, 1\}$ and $y_1 = x_1 = 1 - b$. Hence both recipients must already decide on those values during step 3 of the protocol which implies $|\rho_{01}| = |\rho_0| \geq m_0$ since, otherwise, r_0 would set $y_0 = \perp$. Furthermore, r_0 must not be able to convince r_1 to redecide on $y_1 = y_{01} = y_0$ during step 5 of the protocol. Since the first two conditions according to step 5 are satisfied, i.e.,

- $\perp \neq y_0 = y_{01} \neq y_1$ (since the recipients hold distinct bits), and
- $|\rho_{01}| \geq m_0$ (see above),

the last condition must be violated, i.e., it must hold that

$$|\{i \in \rho_{01} \setminus \sigma_1 : Q_1[i] = 2\}| < \lambda \, |\rho_{01}| \; .$$

Hence s must find some $\ell > (1 - \lambda) \, |\rho_{01}| \geq (1 - \lambda)m_0$ indices i such that either

- $i \in \rho_{01} \cap \sigma_1 \;\; (\subseteq \sigma_0 \cap \sigma_1)$, *or*
- $i \in \rho_{01} \setminus \sigma_1 \;\; \wedge \;\; Q_1[i] \neq 2$ (and hence $Q_s[i] = 2$) ,

such that no collision occurs. By Lemma 5.17 this happens with probability at most $\Pr_s < \left(\frac{m_1}{m_0+m_1}\right)^{(1-\lambda)m_0}$. □

Lemma 5.19 (r_0 possibly faulty). *If the players s and r_1 are correct and each possible outcome of Q-Flip appears at least m_0 times and at most m_1 times then, for any $\lambda > \frac{m_1}{m_0+m_1}$, Protocol 5.4 fails to achieve $WBC_{3,1}$ with a probability of at most $Pr_{r_0} \leq \mathcal{H}(m, m_1, m_0, \lambda m_0)$.*[14]

Proof. The only way for r_0 to make the protocol fail is to make r_1 adopt $y_1 := y_{01} \neq x_s$ during step 5 of the protocol. Hence the following conditions must hold:

- $y_{01} = 1 - y_1 = 1 - x_s$,
- $|\rho_{01}| \geq m_0$,
- $|\{i \in \rho_{01} \setminus \sigma_1 : Q_1[i] = 2\}| \geq \lambda\,|\rho_{01}|$.

Let $u = |\rho_{01}| \geq m_0$. Since s is correct and hence

$$\{i \in \{1, \ldots, m\} : Q_s[i] = 1 - x_s = y_{01} \ \wedge\ Q_1[i] = 2\} \ \subseteq\ \sigma_s = \sigma_1 \ ,$$

r_0 must select u indices i such that for λu of them it holds that $i \notin \sigma_s$, and $Q_0[i] = y_{01}$, and $Q_1[i] = 2$. An optimal strategy in order to achieve this is by randomly selecting m_0 indices i such that $Q_0[i] = 1 - x_s$ (corresponding to random selections from the first and the last row in Figure 5.2–C).

This process corresponds to a hyper-geometric distribution with $N = m$, $K = m_1$, and $n = m_0$ (see Section 2.5). The probability for r_0 to succeed is hence given by the tail of this distribution according to $k \geq \lambda m_0$. By Equation (2.2), for any $\lambda > \frac{m_1}{m_0+m_1}$, this probability can be estimated as

$$\Pr_{r_0} \leq \mathcal{H}(m, m_1, u, \lambda u) \leq \mathcal{H}(m, m_1, m_0, \lambda m_0) \leq e^{-2\left(\lambda - \frac{m_1}{m_0+m_1}\right)^2 m_0} \ .$$

□

Lemma 5.20. *For every desired security parameter $\kappa > 0$ there exist parameters m, m_0, and λ such that Protocol 5.4 has communication and computation complexities polynomial in κ and achieves $WBC_{3,1}$ in model $\mathcal{M}^{\mathrm{q}}_{\mathrm{aut}}$ with an error probability of at most $\varepsilon < 2^{-\kappa}$.*

Let $\mathcal{R}_{\mathrm{q}}$ and $\mathcal{B}_{\mathrm{q}}$ be the round and bit complexities of the underlying Q-Flip primitive, and let κ be the desired security parameter. Then Protocol 5.4 requires a round complexity of $\mathcal{R} = \mathcal{R}_{\mathrm{q}} + 2$ and a bit complexity of $\mathcal{B} = O(\kappa)\mathcal{B}_{\mathrm{q}}$.

[14] See Section 2.5 for the definition of $\mathcal{H}$.

Proof. We let $\lambda_0 = \frac{3}{4}$ and $\lambda_1 = \frac{5}{4}$, and fix the parameterization of the protocol as follows such that there is only one free parameter left, namely m, the number of Q-Flip invocations:

$$\boxed{\begin{array}{lcll} m_0 & = & \lambda_0 \frac{m}{6} & = \frac{m}{8} \\ m_1 & = & \lambda_1 \frac{m}{6} & = \frac{5m}{24} \\ \lambda & = & \lambda_0 & = \frac{3}{4} \end{array}}$$

Now, as a function of security parameter κ, let $m \geq 288\,(\kappa + 2)$. According to Corollary 5.16 and Lemmas 5.14, 5.18, and 5.19 we get the following estimations:

$$\mathrm{Pr}_{\mathsf{stats}} \leq 6 \max\left(\mathcal{C}_{\downarrow}(\frac{1}{6}, m, \lambda_0), \mathcal{C}_{\uparrow}(\frac{1}{6}, m, \lambda_1)\right) \leq 6e^{-\frac{m}{288}} ,$$

$$\mathrm{Pr}_{r_1} = 0 ,$$

$$\mathrm{Pr}_{s} \leq \left(\frac{m_1}{m_0 + m_1}\right)^{(1-\lambda)m_0} = \left(\frac{\lambda_1}{\lambda_0 + \lambda_1}\right)^{(1-\lambda)\lambda_0 m} = \left(\frac{8}{5}\right)^{-\frac{m}{32}} < e^{-\frac{m}{69}} ,$$

$$\mathrm{Pr}_{r_0} \leq e^{-2\left(\lambda - \frac{m_1}{m_0+m_1}\right)^2 m_0} = e^{-2\left(\lambda - \frac{\lambda_1}{\lambda_0+\lambda_1}\right)^2 m_0} = e^{-\frac{3m}{128}} \leq e^{-\frac{m}{43}} .$$

Finally, the overall error probability can be estimated by the sum of the probabilities that either the statistics of Q-Flip fail, i.e., that at least one of the six possible outcomes appears less than m_0 or more than m_1 times, or that, given good statistics, a faulty player can nevertheless successfully misbehave:

$$\begin{aligned} \varepsilon &\leq \mathrm{Pr}_{\mathsf{stats}} + \max(P_{r_1}, P_s, P_{r_0}) \leq 6e^{-\frac{m}{288}} + e^{-\frac{m}{69}} \\ &< 7e^{-\frac{m}{288}} < e^2 e^{-\frac{m}{288}} < e^{-\frac{m-576}{288}} \leq e^{-\kappa} . \end{aligned}$$

The protocol requires two additional communication rounds after the invocations of the Q-Flip primitive, hence the stated round complexity follows. During these rounds, overall, three subsets of $\{1, \ldots, m\}$ (which can be encoded with m bits, each) and three bits are sent. Since $m = 288(\kappa + 2) = O(\kappa)$ guarantees an error probability of $\varepsilon < e^{-\kappa}$, we get a bit complexity of $\mathcal{B} = O(m)\mathcal{B}_{\mathrm{q}} + O(m) = O(\kappa)\mathcal{B}_{\mathrm{q}}$. □

Theorem 5.21 (Broadcast [FGMR02]). *In model $\mathcal{M}^{\mathrm{q}}_{\mathrm{aut}}$, unconditionally secure broadcast is efficiently achievable for $t < n/2$.*

Let $\mathcal{R}_{\mathrm{q}}$ and $\mathcal{B}_{\mathrm{q}}$ be the round and bit complexities of the underlying Q-Flip primitive. Then, for any $\kappa > 0$, there is a broadcast protocol with security parameter κ that requires a round complexity of $\mathcal{R} = 2t \cdot \mathcal{R}_{\mathrm{q}} + 5t + 1$ and a bit complexity of $\mathcal{B} = O(n^4(\kappa + \log n))\mathcal{B}_{\mathrm{q}}$.

Proof. First, note that Protocol 5.1 achieves global weak broadcast even when replacing BC_3 by $\mathrm{WBC}_{3,1}$ in step 1 of the protocol. Thus, consider the broadcast protocol resulting from transforming Protocol 5.4 for $\mathrm{WBC}_{3,1}$ into a protocol for weak broadcast with respect to n and $t < n/2$ according to Protocol 5.1 and, in turn, transforming this protocol for weak broadcast into a broadcast protocol with respect to n and $t < n/2$ according to Theorem 4.14.

By Lemma 5.20, in order to achieve $\mathrm{WBC}_{3,1}$ with security parameter $\kappa_{\mathrm{wbc_3}}$, Protocol 5.4 requires $\mathcal{R}_{\mathrm{wbc_3}} = \mathcal{R}_{\mathrm{q}} + 2$ rounds and a bit complexity of $\mathcal{B}_{\mathrm{wbc_3}} = O(\kappa_{\mathrm{wbc_3}})\mathcal{B}_{\mathrm{q}}$.

By Lemma 5.2 (including the above remark), turning this $\mathrm{WBC}_{3,1}$ protocol into weak broadcast with respect to n and $t < n/2$ results in a protocol that requires $\mathcal{R}_{\mathrm{wbc}} = \mathcal{R}_{\mathrm{wbc_3}} = \mathcal{R}_{\mathrm{q}} + 2$ and $\mathcal{B}_{\mathrm{wbc}} = O(n^2)\mathcal{B}_{\mathrm{wbc_3}} = O(n^2\kappa_{\mathrm{wbc_3}})\mathcal{B}_{\mathrm{q}}$, and involves a security parameter of $\kappa_{\mathrm{wbc}} \geq \kappa_{\mathrm{wbc_3}} - \log_2(n^2) = \kappa_{\mathrm{wbc_3}} - 2\log_2 n$.

By Theorem 4.14, turning this weak broadcast protocol with respect to n and $t < n/2$ into broadcast with respect to n and $t < n/2$ results in a protocol that requires $\mathcal{R} = 2t\mathcal{R}_{\mathrm{wbc}} + t + 1 = 2t\mathcal{R}_{\mathrm{q}} + 5t + 1$ and $\mathcal{B} = O(n^2) + 3nt\mathcal{B}_{\mathrm{wbc}} = O(n^4\kappa_{\mathrm{wbc_3}})\mathcal{B}_{\mathrm{q}}$, and involves a security parameter of $\kappa \geq \kappa_{\mathrm{wbc}} - \log_2 t^2 \geq \kappa_{\mathrm{wbc_3}} - 4\log_2 n$.

Thus, in order to achieve security parameter κ, $\kappa_{\mathrm{wbc_3}}$ can be customized to $\kappa_{\mathrm{wbc_3}} = \kappa + O(\log n)$. Finally, we get the stated complexities $\mathcal{R} = 2t \cdot \mathcal{R}_{\mathrm{q}} + 5t + 1$ and $\mathcal{B} = O(n^4(\kappa + \log n))\mathcal{B}_{\mathrm{q}}$. □

5.5 Applications and open problems

As follows from Theorem 5.3, the protocols in [Bea89, RB89, CDD+99] for unconditionally secure MPC in model $\mathcal{M}^{\mathrm{bc}}_{\mathrm{sec}}$ do not necessarily require the availability of global broadcast but also work in model $\mathcal{M}^{\mathrm{bc_3}}_{\mathrm{sec}}$. Note that, although global broadcast is achievable in model $\mathcal{M}^{\mathrm{bc_3}}_{\mathrm{aut}}$ (BC_3 and authenticated channels), for MPC secure channels are required besides.

Theorem 5.22. *In model $\mathcal{M}^{\mathrm{bc_3}}_{\mathrm{sec}}$, MPC unconditionally secure against $t < n/2$ corrupted players is efficiently achievable.*

Proof. Such an MPC protocol can be constructed by modifying the protocol in [CDD+99]: each invocation of a broadcast channel is simply simulated with help of BC_3 along the lines of Theorem 5.3. □

The same can also be achieved in model $\mathcal{M}^{\mathrm{q}}_{\mathrm{aut}}$, even without requiring the additional condition of secure channels since the Q-Flip primitive helps to build pairwise one-time pads among the players:

Annegret and Beat can generate a one-time pad (OTP) of length approximately k by using $3k$ Q-Flip invocations shared with an arbitrary third player Charlotte and reporting to each other where they got a value different from 2. By this exchange of information Charlotte does not get any additional information about the actual outcome of the Q-Flip invocations. Finally, the OTP is formed by those Q-Flip instances where both Annegret and Beat, got either 0 or 1, e.g., by Annegret's respective bits — which are the complements of Beat's.

Theorem 5.23. *In model $\mathcal{M}^{\mathrm{q}}_{\mathrm{aut}}$, MPC unconditionally secure against $t < n/2$ corrupted players is efficiently achievable.*

Proof. Such an MPC protocol can also be constructed by modifying the protocol in [CDD+99]. Each invocation of a broadcast channel is simulated with help of Q-Flip along the lines of Theorem 5.21. Furthermore, secure pairwise communication is substituted by one-time-pad encrypted communication over the pairwise authenticated channels whereby the one-time pads are extracted from Q-Flip as shown above. □

The result behind Theorem 5.22 turned out to be useful for the characterization of complete primitives. More generally than assuming pairwise communication channels or broadcast channels among subsets of players, it can be assumed that, for some cardinality $k \leq n$, each subset of k players shares a communication primitive of "some kind". Such a primitive would take a secret input from each player, perform some computation, and secretly return an output to each player. Such a primitive is called *complete* if MPC can be achieved among the n players having access only to such primitives and local computation. Following a series of work by Kilian, and Beimel, Malkin, and Micali [Kil88, Kil91, BMM99], Kilian [Kil00] completely characterized which primitives of cardinality $k = 2$ are complete for unconditionally secure two-party computations ($n = 2$) for both the passive and the active model. In [FGMO01], it is proven that, in order to achieve MPC unconditionally secure against $t < n/2$ actively corrupted players, a primitive of cardinality at least $k = 3$ is required. There, primitives of cardinality $k = 3$ are proven to

be complete by showing that the primitives allow to simulate pairwise secure channels and BC_3 channels, and applying Theorem 5.22.

Finally, although the achievability of global broadcast when given BC_b channels was fully characterized in Section 5.3 by showing that global broadcast is achievable if and only if $t < \frac{b-1}{b+1}n$, generally, it remains an open problem whether efficient protocols exist that achieve this bound. A related question is whether always full connectivity of partial broadcast is required among the players in order to achieve global broadcast, i.e., whether partial broadcast among all $\binom{n}{b}$ subsets is necessary.

Chapter 6

Generalized Security Definitions

6.1 Introduction

Consider protocols among n players with respect to a threshold adversary. The standard way to define security is to make it depend on one threshold t: if the adversary corrupts at most t players then the protocol satisfies some required properties — but nothing is guaranteed for the case that the adversary corrupts more than t players, i.e., in this case the protocol may fail in any arbitrary way.

A natural way to eliminate this drawback would be a protocol that satisfies all desired properties if at most some t players are corrupted but in which all players safely abort the protocol when these properties cannot be satisfied, i.e., if more than t players are corrupted. This principle was applied by Bennett and Brassard [BB84] for key agreement among two players with help of a quantum channel. Their protocol reliably works if no eavesdropper is present but is aborted by both players if the quantum channel is being tapped.

Another approach is to specify different levels of correctness depending on the number of corrupted players being present. This principle was applied in [VP93] by Vaidya and Pradhan. They defined the "degradable agreement" problem with respect to two thresholds m and u, $u \geq m$, where broadcast must be achieved if up to m players are corrupted but only a very weak form of broadcast must be achieved if up to u players are corrupted. They proved that degradable agreement is achievable if

and only if $n > 2m + u$.

In this chapter, in a similar way as in [VP93], the notion of broadcast is generalized to two thresholds t_v and t_c in the sense that the validity property of classical Definition 3.1 is guaranteed as long as up to $f \leq t_v$ players are corrupted and the consistency property is guaranteed as long as up to $f \leq t_c$ players are corrupted. Note that this definition was already given by Definition 4.1 of two-threshold broadcast (TTBC), the purpose of which was to easily demonstrate the correctness of the EIG protocol for standard broadcast.

In other words, this definition requires that broadcast is achieved if $f \leq \min(t_v, t_c)$ but that still either validity or consistency is guaranteed if $f \leq \max(t_v, t_c)$. We distinguish the two cases:

- *Broadcast with extended validity (ExtValBC)*: $t_v \geq t_c$, and
- *Broadcast with extended consistency (ExtConsBC)*: $t_c \geq t_v$.

Furthermore, the version with extended consistency can additionally be defined to also achieve agreement about the fact whether or not validity is achieved (*validity detection*), called *ExtConsBC*$^+$. The version with extended validity can additionally be defined to also guarantee that either all players end up with the same output value or all players learn that no consistency has been achieved (*consistency detection*), This variant is called *ExtValBC*$^+$. Note that these variants with detection are related to the failure-discovery problem by Hadzilacos and Halpern in [HH93a], a variant of broadcast in the sense that the broadcast conditions are required to be satisfied only provided that no correct player discovered that "something went wrong". However, the difference is that, in our context, global failure discovery is required, i.e., the broadcast conditions are required to be satisfied provided that any single correct player did not discover that "something went wrong".

ExtValBC$^+$ is achievable if $t_c = 0$ or $t_c + 2t_v < n$ and ExtConsBC$^+$ is achievable if $t_v = 0$ or $t_v + 2t_c < n$. Furthermore, it can be shown that these bounds are tight even with respect to the variants without "detection". These bounds are depicted in Figure 6.1.

Theorem 6.1 ([Hol01, FHHW03]). *Two-threshold broadcast (TTBC) with "detection" is (efficiently) achievable if*

$$t_v = 0 \ \vee \ t_c = 0 \ \vee \ ((t_c + 2t_v < n) \wedge (t_v + 2t_c < n)) \, .$$

If this condition is violated then not even TTBC without "detection" is achievable.

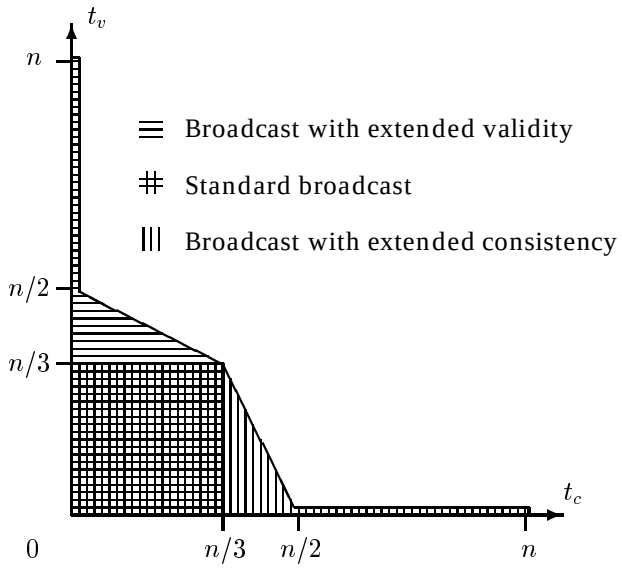

Figure 6.1: Tight bounds for two-threshold broadcast.

Proof. The theorem follows from Theorems 6.2 and 6.6 (achievability of broadcast with extended validity and consistency detection), Theorems 6.9 and 6.11 (achievability of broadcast with extended consistency and validity detection), and Theorem 6.12 (impossibility of TTBC). □

The protocols for broadcast with extended consistency and validity detection can be turned into "detectable" precomputation protocols along the lines of precomputation Protocol 4.12 or the Pfitzmann-Waidner protocol: either all players commonly accept implying that future broadcast is possible, or all players commonly reject. "Detectable precomputation" is of special interest with respect to multi-party computation: it immediately implies that any protocol in a model with broadcast (e.g., standard model $\mathcal{M}_{\text{sec}}^{\text{bc}}$) can be transformed into a non-robust protocol (cf. Section 3.2) in the corresponding model without broadcast (standard model $\mathcal{M}_{\text{sec}}$).

Broadcast with extended consistency for the special case $t_v = 0$ was introduced by Lamport [Lam83] under the name "weak Byzantine generals" (cf. Section 3.1.5). He proved that there is no deterministic protocol when $t_c \geq n/3$. Broadcast with extended consistency and validity detection for the special case $t_v = 0$ was introduced in [FGM01, FGMR02]

under the name "detectable broadcast", and shown there to be achievable if $t_c < n/2$. In [Hol01, FGH+02] it was shown that, for this special case where $t_v = 0$, broadcast with extended consistency and validity detection is even achievable if $t_c < n$. The above definitions with respect to general thresholds t_v and t_c were given in [Hol01, FHHW03].

6.2 Motivation

The mentioned bounds for ExtConsBC$^+$ demonstrate that, beyond the classical lower bounds, Byzantine agreement can nevertheless be run in an "optimistic" manner. Consider, for example, the case $n = 3$ where ordinary Byzantine agreement is not achievable for $t > 0$. The achievability of ExtConsBC$^+$ with respect to the thresholds $t_c < n$ and $t_v = 0$ shows that there is a protocol that achieves Byzantine agreement for the case that no player gets corrupted but, still, for any number of corrupted players, achieves that all players reliably detect whether or not the protocol has been successful.

This "optimistic" model is of particular interest when faults or corruption are expected to be rare but to appear in bursts. Virus infections of servers, for example, only occur from time to time but then it must be expected that many servers get infected at the same time. Repeatedly running invocations of detectable precomputation thus allows to reliably detect a first phase wherein only few servers are infected which can be exploited in order to "vaccinate" the system against future infections of any number of servers.

6.3 Model and definitions

For self-containedness, we start with the redefinition of two-threshold broadcast.

Definition 6.1 (Two-Threshold Broadcast — restated Definition 4.1). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ achieves* two-threshold broadcast *(*TTBC*, for short) with respect to P, p_s, $\mathcal{D}$, and thresholds t_v and t_c if it satisfies the following conditions:*

Validity: If at most $f \leq t_v$ players are corrupted and the sender p_s is correct then all correct players p_i decide on the sender's input value,

$y_i = x_s$.

Consistency: If at most $f \leq t_c$ players are corrupted then all correct players decide on the same output value.

◇

The case where $t_v \geq t_c$ can then be explicitly defined as follows:

Definition 6.2 (Broadcast with extended validity). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ achieves* broadcast with extended validity *(*ExtValBC*) with respect to P, p_s, $\mathcal{D}$, and thresholds t_v and t_c ($t_v \geq t_c$) if it satisfies the following conditions:*

Consistency: If at most $f \leq t_c$ players are corrupted then all correct players decide on the same output value.

Validity: If at most $f \leq t_v$ players are corrupted and the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$.

◇

The drawback of this definition is that the players do not learn whether or not consistency has been achieved. The next definition gives a stronger version which includes an additional grade output g_i for every player that allows to detect whether consistency has been achieved. However, for $t_v \geq n/3$, it is not possible that the players achieve agreement about whether or not consistency has been achieved — this would immediately imply standard broadcast for $t \geq n/3$. But still, it can be guaranteed that consistency is always detected if $f \leq t_c$ ("completeness") and never incorrectly detected if $f \leq t_v$ ("soundness"), i.e., $g_i = 1$ always implies reliable detection of consistency.

Definition 6.3 (Broadcast with extended validity and consistency detection). *Let P be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ and a grade value $g_i \in \{0, 1\}$ achieves* broadcast with extended validity and consistency detection *(*ExtValBC$^+$ *for short) with respect to P, p_s, $\mathcal{D}$, and thresholds t_v and t_c ($t_v \geq t_c$) if it satisfies the following conditions:*

Consistency: If at most $f \leq t_c$ players are corrupted then there is a value y such that every correct player p_i computes $y_i = y$ and $g_i = 1$.

Validity: If at most $f \leq t_v$ players are corrupted and the sender p_s is correct then every correct player p_i decides on the sender's input value, $y_i = x_s$.

Consistency Detection: If at most $f \leq t_v$ players are corrupted and any correct player p_i computes $g_i = 1$ then every correct player p_j computes $y_j = y_i$. ◇

The given protocol for ExtValBC$^+$ in Section 6.4 assumes standard model $\mathcal{M}_{\mathrm{aut}}[\mathrm{u}]$ whereas the respective impossibility proof for ExtValBC is even given with respect to standard model $\mathcal{M}_{\mathrm{sec}}[c]$.

Definition 6.4 (Broadcast with extended consistency). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ achieves* broadcast with extended consistency *with respect to P, p_s, $\mathcal{D}$, and thresholds t_c and t_v ($t_c \geq t_v$) if it satisfies the following conditions:*

Validity: If at most $f \leq t_v$ players are corrupted and the sender p_s is correct then all correct players p_i decide on the sender's input value, $y_i = x_s$.

Consistency: If at most $f \leq t_c$ players are corrupted then all correct players decide on the same output value. ◇

Analogously to ExtValBC, the drawback of this definition is that the players do not learn whether or not validity has been achieved. The next definition gives a stronger version which includes an additional grade output g_i for every player that allows to detect whether validity has been achieved. In contrast to the inherently non-common consistency detection in ExtValBC$^+$ for $t_v \geq n/3$, for ExtConsBC it is always possible that the players decide on the same grade output g_i. If $f \leq t_v$ then validity is always detected ("completeness"), and if $f \leq t_c$ then the detection of validity always implies validity ("soundness").

Definition 6.5 (Broadcast with extended consistency and validity detection aka Detectable Broadcast). *Let P be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where player $p_s \in P$ (called the* sender*) holds an input value $x_s \in \mathcal{D}$ and every player $p_i \in P$ finally decides on an output value $y_i \in \mathcal{D}$ and a grade value $g_i \in \{0, 1\}$ achieves* broadcast with extended consistency and validity detection *(*ExtConsBC$^+$*, for short) with respect to P, p_s, $\mathcal{D}$, and thresholds t_c and t_v ($t_c \geq t_v$) if it satisfies the following conditions:*

Validity: If at most $f \leq t_v$ players are corrupted then every correct player p_i computes grade $g_i = 1$. If, additionally, the sender p_s is correct then $y_i = x_s$.

Consistency: If at most $f \leq t_c$ players are corrupted then every correct player p_i decides on the same pair of outputs (y, g), $y_i = y$ and $g_i = g$.

Validity Detection: If at most $f \leq t_c$ players are corrupted, the sender p_s is correct, and any correct player p_i computes $g_i = 1$ then every correct player p_j computes $y_j = x_s$. ◇

Along the lines of [BPW91, PW96], "detectable broadcast" can be turned into a "detectable precomputation" (see Section 4.3) for future broadcast with arbitrary resilience ($t < n$).

Definition 6.6 (Detectable precomputation). *A protocol among n players where every player $p_i \in P$ computes some private data Δ_i and finally decides on a decision bit $g_i \in \{0, 1\}$ achieves* detectable precomputation for broadcast *(or* detectable precomputation*, for short) with respect to thresholds t_c and t_v ($t_c \geq t_v$) if it satisfies:*

Validity (or Robustness): If at most $f \leq t_v$ players are corrupted then the correct players accept ($g_i = 1$).

Consistency (or Correctness): If at most $f \leq t_c$ players are corrupted then all correct players commonly accept ($g_i = 1$) or commonly reject ($g_i = 0$) the protocol. If the private data Δ_i held by all correct players is inconsistent in the sense that it does not guarantee for arbitrarily resilient broadcast then the correct players reject ($g_i = 0$).

Independence: At the time of the precomputation, a correct player does not yet need to know the value to be broadcast later. ◇

The given protocols for ExtConsBC$^+$ in Section 6.5 assume standard model $\mathcal{M}_{\mathrm{sec}}[u]$ (or $\mathcal{M}_{\mathrm{aut}}[c]$) whereas the impossibility of ExtConsBC is even stated with respect to standard model $\mathcal{M}_{\mathrm{sec}}[c]$. Note that ExtConsBC with perfect security is not achievable if $t_c \geq n/3$, as follows from the impossibility of the "weak Byzantine generals problem" [Lam83].

All given definitions can be naturally adapted to the consensus variant of Byzantine agreement, yielding two-threshold consensus (TTC), consensus with extended validity (ExtValC), consensus with extended validity and consistency detection (ExtValC$^+$), etc.

6.4 Broadcast with extended validity

We directly present an efficient solution for broadcast with extended validity and consistency detection, ExtValBC$^+$, which is strictly stronger

than broadcast with extended validity. The model is $\mathcal{M}_{\mathrm{aut}}$.

6.4.1 Zero consistency

We start with the special case $t_c = 0$ and $t_v < n$, i.e., the case where consistency is only guaranteed if no player is corrupted. In this case, ExtValBC$^+$ can be achieved by a simple two-round protocol that was first given in [Hol01].

Protocol 6.1 $\texttt{ExtValBC}^+(P, p_s, x_s, t_v, t_c)$

1. `if` $i = s$ `then` $\texttt{SendToAll}(x_s)$ `fi`; $\texttt{Receive}(y_i)$;
2. $\texttt{SendToAll}(y_i)$; $\texttt{Receive}(y_i^1, \ldots, y_i^n)$;
3. `if` $y_i = y_i^1 = \cdots = y_i^n$ `then` $g_i := 1$ `else` $g_i := 0$ `fi`;
4. return (y_i, g_i)

Theorem 6.2 ([FGH$^+$02]). *In standard model $\mathcal{M}_{\mathrm{aut}}$, Protocol 6.1 achieves unconditionally secure ExtValBC$^+$ for thresholds $t_c = 0$ and $t_v < n$. Its round and bit complexities are $\mathcal{R} = 2$ and $\mathcal{B} = O(n^2 \log |\mathcal{D}|)$ where $\mathcal{D}$ is the value domain.*

Proof. Since a correct player p_i directly accepts the value y_i received from the sender, validity is trivially satisfied. That consistency holds if no player is corrupted can also be directly seen. Finally, if any correct player p_i computes $g_i = 1$ then all correct players sent value y_i during step 2. Thus, every correct player p_j holds the same vlaue $y_j = y_i$, and consistency detection follows. The stated complexities can be easily verified by code inspection. □

6.4.2 Non-zero consistency

ExtValBC$^+$ with non-zero consistency can be achieved by a phase-king protocol. For its construction, we can proceed in the same way as was done in Section 4.1.1.2 for the standard phase-king protocol but thereby distinguishing between validity with respect to threshold t_v and consistency with respect to threshold t_c. We directly start with a respective protocol for graded consensus including the implicit construction of a respective protocol for weak consensus.

For completeness, we first define the problems two-threshold weak consensus (TTWC) and two-threshold graded consensus (TTGC), which simply generalize the original definitions of weak and graded consensus with respect to separate thresholds t_v and t_c.

Definition 6.7 (Two-threshold variants).

TTWC: A protocol Ψ among P achieves two-threshold weak consensus *with respect to thresholds t_v and t_c if it achieves the validity condition of weak consensus if at most t_v players are corrupted and its consistency condition if at most t_c players are corrupted.*

TTGC: A protocol Ψ among P achieves two-threshold graded consensus *with respect to thresholds t_v and t_c if it achieves the validity condition of graded consensus if at most t_v players are corrupted and its consistency condition if at most t_c players are corrupted.* ◇

Protocol 6.2 TTGC (P, x_i, t_v, t_c)

1. `SendToAll`(x_i); `Receive`$(x_i^1, \ldots, x_i^n)$;
2. $S_i^0 := \{j \in \{1, \ldots, n\} \mid x_i^j = 0\}$; $S_i^1 := \{j \in \{1, \ldots, n\} \mid x_i^j = 1\}$;
3. `if` $|S_i^{x_i}| \geq n - t_v$ `then` $z_i := x_i$ `else` $z_i := \perp$ `fi`;
4. `SendToAll`(z_i); `Receive`$(z_i^1, \ldots, z_i^n)$;
5. $T_i^0 := \{j \in \{1, \ldots, n\} \mid z_i^j = 0\}$; $T_i^1 := \{j \in \{1, \ldots, n\} \mid z_i^j = 1\}$;
6. `if` $|T_i^0| \geq |T_i^1|$ `then` $y_i := 0$ `else` $y_i := 1$ `fi`;
7. `if` $|T_i^{y_i}| \geq n - t_v$ `then` $g_i := 1$ `else` $g_i := 0$ `fi`;
8. return (y_i, g_i)

Lemma 6.3. *In model $\mathcal{M}_{\text{aut}}$, if $t_c + 2t_v < n$ then the first three steps of Protocol 6.2 achieve TTWC for thresholds t_v and t_c (with output z_i).*

Proof.
Validity: If at most $f \leq t_v$ players are corrupted and every correct player p_i holds the same input value $x_i = x$ then, for every correct player p_i, it holds that $|S_i^x| \geq n - t_v$ and hence every such p_i computes $z_i = x_i = x$.

Consistency: If $f \leq t_c$ and any correct player p_i holds value $z_i \in \{0, 1\}$ after step 3 then $|S_i^{z_i}| \geq n - t_v$ and thus, for every correct player p_j, it holds that $|S_j^{z_i}| \geq n - t_v - t_c > t_v$ and thus $z_j \in \{z_i, \perp\}$ after step 3. □

Lemma 6.4. *In model $\mathcal{M}_{\text{aut}}$, if $t_c + 2t_v < n$ and $t_v \geq t_c$, Protocol 6.2 achieves TTGC with respect to thresholds t_v and t_c.*

Proof.
Validity: Suppose that $f \leq t_v$, and that every correct player p_i enters the protocol with the same input value $x_i = x$. Then, by Lemma 6.3 (validity), every correct player p_i holds value $z_i = x$ at the end of step 3

and thus value x is redistributed by all correct players in step 4. Thus $|T_i^x| \geq n - t_v > t_v$, and every correct player computes $y_i = x$ and $g_i = 1$.

Consistency: Suppose that $f \leq t_c$ and that some correct player p_i computes $g_i = 1$ and $y_i = y \in \{0,1\}$. Then $|T_i^y| \geq n - t_v$, and thus $|T_j^y| \geq n - t_v - t_c$ for every correct player p_j. Furthermore, at least one correct player p_k must have sent value $z_k = y$ in step 4. By Lemma 6.3 (consistency), this implies that $|T_j^{1-y}| \leq t_c < n - t_v - t_c \leq |T_j^y|$; and consistency follows. □

Protocol 6.2 could now be directly transformed into a phase-king protocol for broadcast with extended validity. However, in order to achieve broadcast with extended validity *and consistency detection* (ExtValBC$^+$), the definition of two-threshold graded consensus has to be augmented with consistency detection for the case that $t_c < f \leq t_v$ players are corrupted. This is achieved by extending the grade range to $g_i \in \{0,1,2\}$ whereas $g_i \geq 1$ implies consistency detection if at most $f \leq t_c$ players are corrupted and $g_i = 2$ implies consistency detection if at most $f \leq t_v$ players are corrupted, yielding the following definition for graded consensus with extended validity and consistency detection (ExtValGC$^+$, for short and forever).

Definition 6.8 (ExtValGC$^+$). *Let $P = \{p_1, \ldots, p_n\}$ be a set of n players and let $\mathcal{D}$ be a finite domain. A protocol Ψ among P where every player $p_i \in P$ holds an input value $x_i \in \mathcal{D}$ and finally decides on an output value $y_i \in \mathcal{D}$ and a grade $g_i \in \{0,1,2\}$ achieves* graded consensus with extended validity and consistency detection *(or* ExtValGC$^+$*, for short) with respect to P and $\mathcal{D}$ and thresholds t_v and t_c, if it satisfies the following conditions:*

- *Consistency: If $f \leq t_c$ and any correct player p_i computes $g_i \geq 1$ then every correct player p_j computes $y_j = y_i$.*
- *Validity: If at most $f \leq t_v$ players are corrupted and all correct players p_i enter the protocol with the same input value $x_i = x$ then every correct player p_i computes outputs $y_i = x$ and $g_i \geq 1$, and in particular $g_i = 2$ if at most $f \leq t_c$ players are corrupted.*
- *Consistency Detection: If $f \leq t_v$ and any correct player p_i computes $g_i = 2$ then every correct player p_j computes $y_j = y_i$.*

◇

ExtValGC$^+$ can be achieved by only refining the computation of grade g_i in Protocol 6.2, yielding the following protocol.

Protocol 6.3 $\texttt{ExtValGC}^+(P, x_i, t_v, t_c)$

1. $\texttt{SendToAll}(x_i)$; $\texttt{Receive}(x_i^1, \ldots, x_i^n)$;
2. $S_i^0 := \{j \in \{1, \ldots, n\} \mid x_i^j = 0\}$; $S_i^1 := \{j \in \{1, \ldots, n\} \mid x_i^j = 1\}$;
3. $\texttt{if } |S_i^{x_i}| \geq n - t_v \texttt{ then } z_i := x_i \texttt{ else } z_i := \perp \texttt{ fi}$;
4. $\texttt{SendToAll}(z_i)$; $\texttt{Receive}(z_i^1, \ldots, z_i^n)$;
5. $T_i^0 := \{j \in \{1, \ldots, n\} \mid z_i^j = 0\}$; $T_i^1 := \{j \in \{1, \ldots, n\} \mid z_i^j = 1\}$;
6. $\texttt{if } |T_i^0| \geq |T_i^1| \texttt{ then } y_i := 0 \texttt{ else } y_i := 1 \texttt{ fi}$;
7. $\texttt{if } |T_i^{y_i}| \geq n - t_c \texttt{ then } g_i := 2$
8. $\texttt{elseif } |T_i^{y_i}| \geq n - t_v \texttt{ then } g_i := 1$
9. $\texttt{else } g_i := 0 \texttt{ fi}$;
10. return (y_i, g_i)

Lemma 6.5. *If $t_c + 2t_v < n$ and $t_v \geq t_c$ then Protocol 6.3 achieves ExtValGC$^+$ with respect to t_v and t_c.*

Proof.

Consistency: Consistency follows from the proof of Lemma 6.4.

Validity: If $f \leq t_v$ then validity follows from Lemma 6.4. If $f \leq t_c$ then, for every correct player, it holds that $|T_i^{x_i}| \geq n - t_c$, and thus $g_i = 2$.

Consistency detection: Suppose that $f \leq t_v$ and that some correct player p_i computes $g_i = 2$ and $y_i = y \in \{0, 1\}$. Let $\mathcal{C}$ be the set of corrupted players, $\mathcal{S}^y$ be the set of correct players who sent value y in step 1, and let $\mathcal{T}^y$ be the set of correct players who sent value y in step 4. Note that $\mathcal{S}^y = S_j^y \backslash \mathcal{C}$ and $\mathcal{T}^y = T_j^y \backslash \mathcal{C}$ for any j.

Grade $g_i = 2$ implies $|T_i^y| \geq n - t_c$, and thus $|\mathcal{T}^y| \geq n - t_c - t_v$. Since $z_j \in \{x_j, \perp\}$ for every correct player p_j it follows that $|\mathcal{T}^y| \leq |\mathcal{S}^y|$. Hence, for every player p_j, $|S_j^y| \geq |\mathcal{S}^y| \geq n - t_c - t_v$. The bound $n > t_c + 2t_v$ now implies that $|S_j^{1-y}| \leq t_c + t_v < n - t_v$ and hence that $\mathcal{T}^{1-y} = \emptyset$. Thus, $|T_j^{1-y}| \leq |\mathcal{C}| \leq t_c$ and $|T_j^y| \geq |\mathcal{T}^y| \geq n - t_c - t_v > t_v$, and $y_j = y_i$. □

Protocol ExtValBC$^+$ can now be built along the lines of phase-king Protocol 4.6 by basically replacing graded consensus by TTGC. The only difference is an additional, final round of ExtValGC$^+$ in order to allow for consistency detection (note that the sender and all kings might be faulty if $f > t_c$). A more uniform way to present the protocol would be to interpret the sender as a first king and reversing the order between "king's vote" and TTGC. Nevertheless, in order to maintain compatibility with the previous chapters, we preserve the standard order.

Protocol 6.4 $\texttt{ExtValBC}^+(P, p_1, x_1, t_v, t_c)$

1. `if` $i = 1$ `then` $\texttt{SendToAll}(x_1)$ `fi`; $\texttt{Receive}(y_i)$;
2. `for` $k := 2$ `to` $t_c + 1$ `do`
3. $(y_i, h_i) := \texttt{TTGC}(P, y_i, t_v, t_c)$;
4. `if` $i = k$ `then` $\texttt{SendToAll}(y_i)$ `fi`; $\texttt{Receive}(y_i^k)$;
5. `if` $h_i = 0$ `then` $y_i := y_i^k$ `fi`;
6. `od`; $(y_i, h_i) := \texttt{ExtValGC}^+(P, y_i, t_v, t_c)$;
7. `if` $h_i = 2$ `then` $g_i := 1$ `else` $g_i := 0$ `fi`;
8. return (y_i, g_i)

Theorem 6.6 ([FHHW03]). *In model $\mathcal{M}_{\text{aut}}$, if $t_c + 2t_v < n$ and $t_v \geq t_c$ (and $t_c > 0$), Protocol 6.4 achieves efficient, perfectly secure ExtValBC$^+$ with respect to sender p_1 and thresholds t_v and t_c. The round and bit complexities are $\mathcal{R} = 3t_c + 3$ and $\mathcal{B} = O(n^3)$.*

Proof.

Validity: If $f \leq t_v$ and the sender p_1 is correct then, by the validity properties of TTGC and ExtValGC$^+$, every correct player p_i computes $y_i = x_1$ at the end of the protocol.

Consistency: If $f \leq t_c$ then there is a correct player $p_\ell \in \{p_1, \ldots, p_{t_c+1}\}$. At the end of phase $k = \ell$, every correct player p_i holds the same value $y_i = y_\ell = y$ which, by the validity properties of TTGC and ExtValGC$^+$, stays persistent until step 7 of the protocol and every correct player finally computes $y_i = y$, $h_i = 2$, and thus $g_i = 1$.

Consistency Detection: Suppose that $f \leq t_v$ and that some correct player p_i computes $g_i = 1$. This implies that $h_i = 2$ after step 6, and by the consistency-detection property of ExtValGC$^+$, that every correct player p_j computed $y_j = y_i$ during this invocation and thus terminated the protocol with $y_j = y_i$.

The stated complexities can be easily verified by code inspection. □

6.5 Broadcast with extended consistency

We directly present a construction for detectable precomputation which is strictly stronger than ExtConsBC$^+$. We first give a generic construction, followed by the case where $t_v = 0$ (zero validity), and then by the general case of $t_v + 2t_c < n$. The models are $\mathcal{M}_{\text{aut}}$ for the protocols with computational security, and $\mathcal{M}_{\text{sec}}$ for the protocols with unconditional security.

6.5.1 Generic construction

It turns out that any "reasonable" precomputation protocol according to Definition 4.3 based on temporarily available broadcast can be transformed into a protocol for detectable precomputation (or a protocol for detectable broadcast) in the same model but not requiring broadcast.

This generic construction is basically obtained by substituting each invocation of a broadcast channel by an invocation of ExtValBC$^+$ with respect to thresholds $t'_v = t_c$ and $t'_c = t_v$. Assume set $P = \{p_1, \ldots, p_n\}$ of players in model $\mathcal{M} \in \{\mathcal{M}_{\mathrm{aut}}, \mathcal{M}_{\mathrm{sec}}\}$ and let Ψ be a precomputation protocol for model $\mathcal{M}$ where, additionally, players are assumed to be able to broadcast messages; and let $B = \{\beta_1, \ldots, \beta_n\}$ be a set of protocols for model $\mathcal{M}$ where each protocol β_i achieves broadcast with sender p_i when based on the information exchanged during an execution of Ψ. Furthermore, assume that Ψ satisfies the independence property of Definition 6.6 with respect to the protocols in B.

If protocol Ψ always efficiently terminates even if all involved broadcast invocations fail and all protocols β_i always efficiently terminate even when based on any incorrectly precomputed information then a protocol for detectable precomputation can be achieved from Ψ and B as follows:

Protocol 6.5 `DetPrecomp`(P)

1. Run protocol Ψ wherein each invocation of broadcast is replaced by an invocation of ExtValBC$^+$ (Protocol 6.1 or 6.4) with the same sender with respect to thresholds $t'_v = t_c$ and $t'_c = t_v$.
2. Compute the logical AND of the grades gotten during all ℓ invocations of ExtValBC$^+$ in (modified) Protocol Ψ, $\Gamma_i := G_i := \bigwedge_{k=1}^{\ell} g_i^k$.
3. Send value G_i to each other player; receive the values $G_i^1, \ldots, G_i^n$.
4. For each player $p_j \in P$, participate in an invocation of Protocol β_j wherein p_j inputs Γ_j; receive the values $\Gamma_i^1, \ldots, \Gamma_i^n$, and set $\Gamma_i^i := \Gamma_i$.
5. Compute $g_i = 1$ if $|\{j \mid G_i^j = 1\}| \geq n - t_v - t_c \wedge |\{j \mid \Gamma_i^j = 1\}| \geq n - t_v$ and $g_i = 0$, otherwise.

Note that the protocols β_i in step 4 do not necessarily achieve broadcast since the invocation of Ψ during step 1 might have failed. However, they will always efficiently terminate by assumption. We now argue that the resulting protocol achieves detectable precomputation.

Lemma 6.7. *Consider model $\mathcal{M} \in \{\mathcal{M}_{\mathrm{aut}}, \mathcal{M}_{\mathrm{sec}}\}$. Let Ψ be a precomputation protocol for model $\mathcal{M}$ where the players are additionally assumed to be able to broadcast messages, and let $B = \{\beta_i | i \in \{1, \ldots, n\}\}$ be a set of protocols for model $\mathcal{M}$ where each protocol β_i achieves broadcast with sender p_i when based*

on the information Δ computed during the execution of Ψ.

If protocol Ψ always efficiently terminates even if all broadcast invocations fail and all protocols β_i always efficiently terminate even when based on arbitrary precomputed information then Protocol 6.5 achieves detectable precomputation.

Proof.
Consistency: Suppose that $f \leq t_c$ players are corrupted. If every correct player p_i rejects by computing $g_i = 0$ then consistency is satisfied.

Thus, suppose that some correct player p_i accepts by computing $g_i = 1$. We first show that, according to the definition of ExtValBC$^+$, this implies that all invocations of Protocol 6.1 (or 6.4, respectively) achieved broadcast (when neglecting the grade outputs), thus implying that the players share a PKI according to Definition 2.5:

- $t_v = 0$: $g_i = 1$ implies that $|\{j \mid \Gamma_i^j\}| \geq n$, in particular that $\Gamma_i = 1$, and thus that all invocations of Protocol 6.1 achieved broadcast.
- $t_v > 0$: $g_i = 1$, i.e., $|\{j \mid G_i^j = 1\}| \geq n - t_v - t_c > t_c$, implies that at least one correct player p_k sent $G_k = 1$ and thus, that all invocations of Protocol 6.4 achieved broadcast.

Hence, the β-protocols in step 4 all achieve broadcast and all correct players p_j compute the same set of values $\Gamma_j^1, \ldots, \Gamma_j^n$. Furthermore, since $g_i = 1$, for every correct player p_ℓ it holds that $|\{j \mid \Gamma_\ell^j\}| \geq n - t_v$ and thus that $|\{j \mid G_\ell^j = 1\}| \geq n - t_v - t_c$, and every player p_ℓ computes $g_\ell = 1$.
Validity: Suppose that $f \leq t_v$ players are corrupted. Hence, according to the definition of ExtValBC$^+$, all invocations of Protocol 6.4 (or 6.1, respectively) achieve broadcast (when neglecting the grade outputs) and that every correct player p_i computes $g_i = 1$. Thus, the players share a PKI, all correct players p_i compute $G_i = 1$, all β-protocols achieve broadcast, and, in steps 3 and 4, the players p_i compute values G_i^j and Γ_i^j such that $|\{j \mid G_i^j = 1\}| \geq n - t_v > n - t_v - t_c$ and $|\{j \mid \Gamma_i^j = 1\}| \geq n - t_v$. Finally, every correct player p_i computes $g_i = 1$.
Independence: Independence directly follows from the assumed independence property of precomputation Protocol Ψ. □

6.5.2 Zero validity

6.5.2.1 Computational security

We now tailor generic Protocol 6.5 to computational security with respect to $t_v = 0$ and $t_c < n$ in model $\mathcal{M}_{\text{aut}}$. Note that step 3 of generic Proto-

col 6.5 (the additional echo round before broadcast) becomes obsolete for this special case.

Protocol 6.6 `DetPrecomp`(P)

1. Generate a secret-key / public-key pair $(\mathrm{SK}_i, \mathrm{PK}_i)$ according to the key generation algorithm of a digital signature scheme. For every player p_j ExtValBC$^+$ Protocol 6.1 with respect to thresholds $t'_v < n$ and $t'_c = 0$ is invoked where p_j inputs his public key PK_j as a sender. Every player p_i stores all received public keys $\mathrm{PK}_i^1, \ldots, \mathrm{PK}_i^n$ and grades $g_i^1, \ldots, g_i^n$.
2. Compute $\Gamma_i := \bigwedge_{k=1}^n g_i^k$.
4. For every player $p_j \in P$ an instance of broadcast Protocol 4.10 (Dolev-Strong) is invoked where p_j inputs Γ_j as a sender. Store the received values Γ_i^j ($j \in \{1, \ldots, n\} \setminus \{i\}$) and $\Gamma_i^i := \Gamma_i$.
5. Compute $g_i := 1$ if $\bigwedge_{j=1}^n \Gamma_i^j = 1$ and $g_i := 0$, otherwise.

Note that the bit complexity of the whole protocol can be reduced by replacing the n parallel invocations of the Dolev-Strong protocol during step 4 by a consensus-like protocol with default value 1. For this, the n broadcast protocols (with respect to n different senders) are run in parallel in a slightly modified way. In the first round, a sender p_s who accepts simply sends bit $G_s = 1$ without a signature (or no message at all), and only if he rejects sends bit $G_s = 0$ together with a signature on it. As soon as, during some round $r = 1, \ldots, t+1$, a player accepts the value $G_s = 0$ from one or more senders p_s because he received valid signatures from r different players (including p_s) on value $G_s = 0$ with respect to the protocol instance with sender p_s then, for exactly one arbitrary such sender p_s, he adds his own signature for 0 with respect to this sender's corresponding protocol instance and, during the next round, relays all $r+1$ signatures to every other player, decides on 0, and terminates. If a player never accepts value $G_s = 0$ from any sender p_s then he decides on 1 after round $t+1$. If all players p_s are correct and send value $G_s = 1$ then, clearly, all players decide on 1 at the end. On the other hand, if any correct player decides 0 then all correct players do so. Overall, during this protocol, at most $O(n)$ more signatures are exchanged than during a single invocation of the Dolev-Strong protocol.

Theorem 6.8 (Detectable precomputation [FGH$^+$02]). *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 6.6 achieves detectable precomputation among n players computationally secure with respect to thresholds $t_c < n$ and $t_v = 0$. The correct players all terminate the protocol during the same communication round. The round and bit complexities are $\mathcal{R} = t_c + 3$ and $\mathcal{B} = O(n^2 k_{\mathrm{key}} + n^3|\sigma|)$. where k_{key} is the*

maximal size of a public key and $|\sigma|$ is the maximal size of a signature.

Proof. Validity and consistency follow from Lemma 6.7, and independence is evident.

Termination and complexities: That all correct players terminate during the same communication round can be easily seen by code inspection. As ExtValBC$^+$ Protocol 6.1 requires $\mathcal{R}_{\text{ev}} = 2$ rounds and a bit complexity of $\mathcal{B}_{\text{ev}} = n^2 k_{\text{key}}$, and the n merged parallel invocations of Dolev-Strong Protocol 4.10 as described above require $\mathcal{R}_{\text{bc}} = t_c + 1$ rounds of communication and a bit complexity of $\mathcal{B}_{\text{bc}} = O(n^3|\sigma|)$ bits, Protocol 6.6 requires $\mathcal{R} = t_c + 3$ rounds and a bit complexity of $\mathcal{B} = O(n^2 k_{\text{key}} + n^3|\sigma|)$. □

6.5.2.2 Unconditional security

We now tailor generic Protocol 6.5 to unconditional security with respect to $t_v = 0$ and $t_c < n$ in model $\mathcal{M}_{\text{sec}}$.

A direct approach would be to instantiate Protocol Ψ in Protocol 6.5 with Pfitzmann-Waidner Protocol 4.13. However, most of that protocol consists of fault localization, i.e., sub-protocols that allow to identify players that have been misbehaving. These steps are not required if $t_v = 0$ since, for this special case, we are only interested in detecting whether faults occurred or not. As soon as a fault is detected the players simply reject the outcome of the whole protocol. We now sketch a reduced version of the Pfitzmann-Waidner protocol where the fault localization steps are stripped off.

Protocol 6.7 $\texttt{Precomp}(P)$

- For $(j = s, A)$, and (j, A), (j, B) $(j \in \{1, \ldots, n\} \setminus \{s\})$ in parallel:
  - For $k = 1 \ldots m$ in parallel:
    - If $i \neq j$: select random authentication key κ_i;
    - For $\ell = 1 \ldots 2n$ in parallel:
      - $\forall p_h$ $(h \neq i)$ agree on a pairwise key $K_{hi}^{(\ell)}$;
      - Broadcast $\kappa_i^{2\ell-1} + \sum_{h \neq i} K_{hi}^{(\ell)}$;
      - If $i = j$: send to all players "accept" or "reject";
- Decide to accept ($h_i := 1$) if and only if all signers p_j sent message "accept" with respect to (j, A) and (j, B); otherwise reject ($h_i := 0$);

Note that, in contrast to the original Pfitzmann-Waidner protocol, it is sufficient to have the signers p_j distribute their acceptance / rejection at the end by normal bilateral sending instead of broadcast since a final broadcast round according to step 4 of generic Protocol 6.7 will follow anyway.

Protocol 6.8 `DetPrecomp`(P)

1. Execute precomputation Protocol 6.7 for $b+n$ future broadcasts wherein each invocation of broadcast is replaced by an invocation of ExtValBC$^+$ Protocol 6.1 with respect to thresholds $t_v' < n$ and $t_c' = 0$. Of these instances, b are computed with respect to the intended future senders $s \in \{1,\ldots,n\}$ of the future broadcasts. Of the other n instances, one is computed with respect to each player $p_i \in P$.
2. Compute $\Gamma_i = \bigwedge_{j=1}^{\ell} g_i^j \quad \wedge \quad \bigwedge_{m=1}^{b+n} h_i^m$ where the g_i^j are the grades received during all ℓ invocations of ExtValBC$^+$ during step 1 and h_i^m is the bit indicating whether p_i accepted the m-th execution of precomputation Protocol 6.7.
4. For every player p_j an instance of Protocol 4.10 (Dolev-Strong — with pseudo-signatures) is invoked where p_j inputs Γ_j as a sender. Store the received values Γ_i^j ($j \in \{1,\ldots,n\} \setminus \{i\}$) and $\Gamma_i^i := \Gamma_i$.
5. Compute $g_i := 1$ if $\bigwedge_{j=1}^{n} \Gamma_i^j = 1$ and $g_i := 0$, otherwise.

Again, step 3 of generic detectable precomputation Protocol 6.5 (the additional echo round before broadcast) becomes obsolete for this special case.

Theorem 6.9 (Detectable precomputation [FGH$^+$02]). *In model $\mathcal{M}_{\mathrm{sec}}$, for any integer $b > 0$ and security parameter $\kappa > 0$, Protocol 6.8 achieves detectable precomputation for b broadcasts among n players unconditionally secure with respect to thresholds $t_c < n$ and $t_v = 0$ with the following properties:*

The error probability of any future broadcast is $\varepsilon < 2^{-\kappa}$. Protocol 6.8 has round complexity $\mathcal{R} = t_c + 5$ and bit complexity $\mathcal{B} = O((n+b)n^7 \log\log|\mathcal{D}|(\kappa + \log n + \log b + \log\log\log|\mathcal{D}|)^2 + n^2 \log|\mathcal{D}|)$ where $\mathcal{D}$ is the domain of future messages to be broadcast (including possible padding for session IDs, etc.). Protocol 4.10 for future broadcast then has round complexity $\mathcal{R}_{\mathrm{bc}} = t_c + 1$ and bit complexity $\mathcal{B}_{\mathrm{bc}} = O(n^2 \log|\mathcal{D}| + n^6(\kappa + \log n)^2)$. All correct players terminate the protocol during the same communication round.

Proof. Validity and consistency follow from Lemma 6.7. Independence follows form the independence property of Protocol 6.7.

Executing Protocol 6.7 with security parameter κ guarantees each single of the $b+n$ broadcasts to have an error probability of $\varepsilon < 2^{-\kappa}$ as follows from Proposition 4.25. The error probability of each of the b "net" broadcasts is given by the probability that one of the n broadcasts during step 4 fails and the probability that the one broadcast fails given that those n broadcasts reliably worked, which is bounded by $(n+1)$ times the error probability of one single broadcast being precomputed for. Exe-

cuting Protocol 6.7 with security parameter $\kappa_{\mathrm{pc}} \geq \kappa + \lceil \log_2(n+b) \rceil$ hence bounds the error probability of any single "net" broadcast to $\varepsilon < 2^{-\kappa}$.

Protocol 6.7 where broadcast is substituted by ExtValBC$^+$ Protocol 6.1 requires $\mathcal{R}_{\mathrm{pc}} = 4$ communication rounds and, per future broadcast, a bit complexity of $\mathcal{B}_{\mathrm{pc}} = O(n^7 \log\log|\mathcal{D}|(\kappa_{\mathrm{pc}} + \log n + \log\log\log|\mathcal{D}|)^2)$ where $\mathcal{D}$ is domain of future messages to be broadcast. Protocol 4.10 with the respective pseudo-signatures requires $\mathcal{R}_{\mathrm{bc}} = t_c + 1$ rounds of communication and bit complexity $\mathcal{B}_{\mathrm{bc}} = O(n^2 \log|\mathcal{D}| + n^6(\kappa + \log n)^2)$. Protocol 6.7 is invoked for $b+n$ broadcasts (step 1) and Protocol 4.10 is invoked n times in parallel (step 4) hence yielding round complexity $\mathcal{R} = 4 + (t_c + 1) = t_c + 5$. Including a consensus-like step 4 as mentioned for the computational case, Protocol 6.8 has bit complexity $\mathcal{B} = (b+n)\mathcal{B}_{\mathrm{pc}} + \mathcal{B}_{\mathrm{bc}} = O((b+n)n^7 \log\log|\mathcal{D}|(\kappa + \log n + \log b + \log\log\log|\mathcal{D}|)^2 + n^2 \log|\mathcal{D}|)$. That all correct players terminate during the same communication round can be easily seen by code inspection. □

Furthermore, as follows from Proposition 4.26, using the regeneration techniques in [PW96], the bit complexity of Protocol 6.7 can be reduced to polylogarithmic in the number b of later broadcasts to be precomputed for, i.e., to polynomial in n, $\log|\mathcal{D}|$, κ, and $\log b$.

6.5.3 Non-zero validity

6.5.3.1 Computational security

For completeness, we again state the full protocol for detectable precomputation for model $\mathcal{M}_{\mathrm{aut}}$ with respect to any thresholds t_c and t_v such that $t_v + 2t_c < n$.

Protocol 6.9 `DetPrecomp`(P)

1. Generate a secret-key / public-key pair $(\mathrm{SK}_i, \mathrm{PK}_i)$ according to the key generation algorithm of a digital signature scheme. For every player p_j ExtValBC$^+$ Protocol 6.4 with respect to thresholds $t'_v = t_c$ and $t'_c = t_v$ is invoked where p_j inputs his public key PK_j as a sender. Store all received public keys $\mathrm{PK}_i^1, \ldots, \mathrm{PK}_i^n$ and grades $g_i^1, \ldots, g_i^n$.
2. Compute $\Gamma_i := G_i := \bigwedge_{k=1}^n g_i^k$.
3. Send value G_i to each other player; receive the values $G_i^1, \ldots, G_i^n$.
4. For each player $p_j \in P$, an instance of broadcast Protocol 4.10 (Dolev-Strong) is invoked where p_j inputs Γ_j as a sender. Store the received values Γ_i^j ($j \in \{1, \ldots, n\}$).
5. Compute $g_i = 1$ if $|\{j \mid G_i^j = 1\}| > t_c \wedge |\{j \mid \Gamma_i^j = 1\}| \geq n - t_v$ and $g_i = 0$, otherwise.

Theorem 6.10 (Detectable precomputation [FHHW03]). *In model $\mathcal{M}_{\mathrm{aut}}$, Protocol 6.9 achieves computationally secure detectable precomputation among n players with respect to any thresholds t_c and t_v satisfying $t_v + 2t_c < n$. The correct players all terminate the protocol during the same communication round. The round and bit complexities are $\mathcal{R} = t_c + 3t_v + 4 \leq 4(t_c + 1)$ and $\mathcal{B} = O(n^4 k_{\mathrm{key}} + n^3|\sigma|)$ where k_{key} is the maximal size of a public key and $|\sigma|$ is the maximal size of a signature.*

Proof. Validity and consistency follow from Lemma 6.7 (note that $n - t_v - t_c > t_c$), and independence is evident. That all correct players terminate during the same communication round can be easily seen by code inspection.

Protocol 6.4 requires $\mathcal{R}_{\mathrm{ev}} = 3(t'_c + 1) = 3(t_v + 1)$ rounds and a bit complexity of $\mathcal{B}_{\mathrm{ev}} = O(n^3 k_{\mathrm{key}})$. Protocol 4.10 requires $\mathcal{R}_{\mathrm{bc}} = t_c + 1$ rounds of communication and a bit complexity of $\mathcal{B}_{\mathrm{bc}} = O(n^3|\sigma|)$. Hence, Protocol 6.6 requires $\mathcal{R} = t_c + 3t_v + 4$ rounds and a bit complexity of $\mathcal{B} = O(n^4(k_{\mathrm{key}} + |\sigma|))$. □

6.5.3.2 Unconditional security

For completeness, we again state the full protocol for detectable precomputation for model $\mathcal{M}_{\mathrm{sec}}$ with respect to any bounds t_c and t_v such that $t_v + 2t_c < n$.

Protocol 6.10 `DetPrecomp`(P)

1. Execute precomputation Protocol 4.13 for $b + n$ future broadcasts wherein each invocation of broadcast is replaced by an invocation of ExtValBC$^+$ Protocol 6.4 with respect to thresholds $t'_v = t_c$ and $t'_c = t_v$. Of these instances, b are computed with respect to the intended senders $s \in \{1, \ldots, n\}$ of the future broadcasts. Of the other n instances, one is computed with respect to each player $p_i \in P$.
2. Every player p_i computes $\Gamma_i := G_i = \bigwedge_{k=1}^{\ell} g_i^k$ where the g_i^k are the grades received during all ℓ invocations of ExtValBC$^+$ during step 1. Synchronize: Wait and start executing the next step at round $\lfloor \frac{n^2(9t_v+10)}{2} \rfloor + 1$.
3. Send value G_i to each other player; receive the values $G_i^1, \ldots, G_i^n$.
4. For every player p_j an instance of broadcast Protocol 4.10 (Dolev-Strong — with pseudo-signatures) is invoked where p_j inputs Γ_j as a sender; receive the values $\Gamma_i^1, \ldots, \Gamma_i^n$.
5. Compute $g_i = 1$ if $|\{j \mid G_i^j = 1\}| > t_c \wedge |\{j \mid \Gamma_i^j = 1\}| \geq n - t_v$ and $g_i = 0$, otherwise.

Theorem 6.11 (Detectable precomputation [FHHW03]). *In model $\mathcal{M}_{\text{sec}}$, for any integer $b > 0$ and security parameter $\kappa > 0$, Protocol 6.10 achieves unconditionally secure detectable precomputation for b later broadcasts among n players with respect to thresholds t_c and t_v satisfying $t_v + 2t_c < n$ with the following properties:*

The error probability of any future broadcast is $\varepsilon < 2^{-\kappa}$. Protocol 6.10 has round complexity $\mathcal{R} = \lfloor \frac{n^2(9t_v+10)}{2} \rfloor + t_c + 1$ and bit complexity $\mathcal{B} = O(n^{11} \log |\mathcal{D}| \log\log |\mathcal{D}| (\kappa + \log n + \log\log\log |\mathcal{D}|)^2 + n^2 \log |\mathcal{D}|)$ where $\mathcal{D}$ is the domain of future messages to be broadcast (including possible padding for session IDs, etc.). Protocol 4.10 for future broadcast then has round complexity $\mathcal{R}_{\text{bc}} = t + 1$ and bit complexity $\mathcal{B}_{\text{bc}} = O(n^2 \log |\mathcal{D}| + n^6(\kappa + \log n)^2)$.

The correct players all terminate the protocol during the same communication round.

Proof. Validity and consistency follow from Lemma 6.7. Independence follows form the independece property of Protocol 4.13. Executing Protocol 4.13 with security parameter κ guarantees each single of the $b + n$ broadcasts to have an error probability of $\varepsilon < 2^{-\kappa}$ as is stated in Proposition 4.25. The error probability of each of the b "net" broadcasts is given by the probability that one of the n broadcasts during step 4 fails and the probability that the one broadcast fails given that those n broadcasts reliably worked, which is bounded by $(n + 1)$ times the error probability of one single precomputed broadcast. Executing Protocol 4.13 with security parameter $\kappa_{\text{pc}} \geq \kappa + \lceil \log_2(n + b) \rceil$ hence bounds the error probability of any single "net" broadcast to $\varepsilon < 2^{-\kappa}$.

Protocol 6.4 requires $3(t'_c + 1) = 3(t_v + 1)$ rounds, and hence Protocol 4.13 where broadcast is replaced by Protocol 6.4 requires $\mathcal{R}_{\text{pc}} \leq \frac{n^2}{2} + \frac{3}{2}n^2 \cdot 3(t_v + 1) = \frac{n^2}{2}(9t_v + 10)$ communication rounds — note that hence all correct players get synchronized at the end of step 2 and thus will terminate during the same communication round — and, per future broadcast, a bit complexity of $\mathcal{B}_0 = O(n^8 \log\log |\mathcal{D}| (\kappa + \log n + \log\log\log |\mathcal{D}|)^2 \cdot n^3 \log |\mathcal{D}|)$ where $\mathcal{D}$ is the domain of future messages to be broadcast. Protocol 4.10 with the respective pseudo-signatures requires $\mathcal{R}_{\text{bc}} = t_c + 1$ rounds and bit complexity $\mathcal{B}_{\text{bc}} = O(n^2 \log |\mathcal{D}| + n^6(\kappa + \log n)^2)$.

Protocol 4.13 is invoked for $b + n$ later broadcasts (step 1) and Protocol 4.10 is invoked n times in parallel (step 4) hence yielding round complexity $\mathcal{R} = \lfloor \frac{n^2(9t_v+10)}{2} \rfloor + t_c + 1$. Thus, Protocol 6.10 has bit complexity $\mathcal{B} = (b + n)\mathcal{B}_0 + n\mathcal{B}_{\text{bc}} = O((n + b)n^{11} \log |\mathcal{D}| \log\log |\mathcal{D}| (\kappa + \log n + \log\log\log |\mathcal{D}|)^2 + n^3 \log |\mathcal{D}|)$. □

Furthermore, using the regeneration technique sketched in Section 4.3.1, the protocol can be modified such that the number b of later broadcasts only contributes to the message complexity in polylogarithmic order.

6.6 Impossibility result

It remains to show that the given bounds for ExtValBC$^+$ and ExtConsBC$^+$ (, and detectable precomputation) are optimal. It fact, these bounds are optimal even with respect to the weaker definitions of ExtValBC and ExtConsBC. Since ExtValBC and ExtConsBC are special cases of TTBC, we directly show that two-threshold broadcast (TTBC) is impossible if $t_c > 0$, $t_v > 0$, and either $t_c + 2t_v \geq n$ or $t_v + 2t_c \geq n$.

Theorem 6.12 (Impossibility [FHHW03]). *In standard models $\mathcal{M}_{\text{aut}}$ and $\mathcal{M}_{\text{sec}}$, two-threshold broadcast among a set of n players $P = \{p_0, \ldots, p_{n-1}\}$ is impossible if $t_c > 0$, $t_v > 0$, and either $t_c + 2t_v \geq n$ or $t_v + 2t_c \geq n$. For every protocol there exists a value $x_0 \in \{0,1\}$ such that, when the sender holds input x_0, the adversary can make the protocol fail*

- *with a probability of at least $\frac{1}{6}$ if she is computationally bounded, and*
- *with a probability of at least $\frac{1}{3}$ if she is computationally unbounded.*

Proof. Assume Ψ to be a protocol for two-threshold broadcast among n players $p_0, \ldots, p_{n-1}$ with sender p_0 that tolerates $t_c > 0$, $t_v > 0$, and either $t_c + 2t_v \geq n$ or $t_v + 2t_c \geq n$. Let $t_+ = \max(t_c, t_v)$ and $t_- = \min(t_c, t_v)$.

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs. As follows from the impossibility of standard broadcast it must hold that $t_- < n/3$, and thus, that $t_+ \geq n/3$. Hence, the processors can be partitioned into three sets, $\Pi_0 \dot\cup \Pi_1 \dot\cup \Pi_2 = \Pi$, such that $1 \leq |\Pi_0| \leq t_c$, $1 \leq |\Pi_1| \leq t_v$, and hence $1 \leq |\Pi_2| \leq t_+$. Note that, hence, $|\Pi_0 \cup \Pi_1| \geq n - t_+$, $|\Pi_1 \cup \Pi_2| \geq n - t_c$, and $|\Pi_2 \cup \Pi_0| \geq n - t_v$.

Furthermore, for each $i \in \{0, \ldots, n-1\}$, let π_{i+n} be an identical copy of processor π_i. For every π_i ($0 \leq i \leq 2n-1$) let the *type* of processor π_i be defined as the number $i \bmod n$. Finally, for each $k \in \{0,1,2\}$, let $\Pi_{k+3} = \{\pi_{i+n} \mid \pi_i \in \Pi_k\}$ form identical copies of the sets Π_k.

Instead of connecting the original processors as required for broadcast, we build a network involving all $2n$ processors (i.e., the original ones together with their copies) by arranging the six processor sets Π_k in

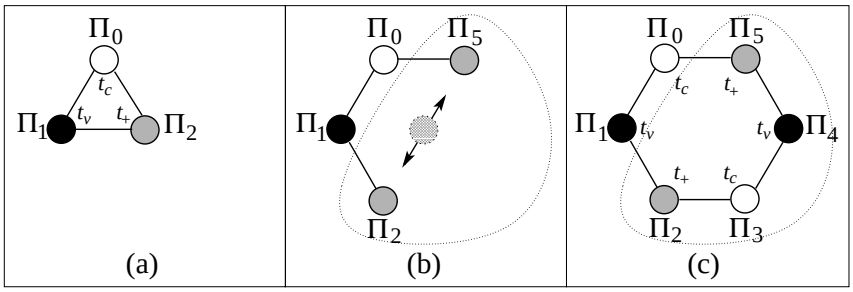

Figure 6.2: *Rearrangement of processors in the proof of Theorem 6.12.*

a circle. In particular, for all sets Π_k ($0 \leq k \leq 5$), every processor $\pi_i \in \Pi_k$ is connected (exactly) by one channel with all processors in $\Pi_k \setminus \{\pi_i\}$, $\Pi_{(k-1) \bmod 6}$, and $\Pi_{(k+1) \bmod 6}$. Hence, each processor π_i in the new system is symmetrically connected with exactly one processor of each type (different from his own one) as in the original system. We say that Π_k and Π_ℓ are *adjacent processor sets* if and only if $\ell \equiv k \pm 1 \pmod 6$.

Now, along the lines of [FLM86], for every set $\Pi_k \cup \Pi_{(k+1) \bmod 6}$ ($0 \leq k \leq 5$) in the new system and without the presence of an adversary, their common view is indistinguishable from their view as the set of processors $\Pi_{k \bmod 3} \cup \Pi_{(k+1) \bmod 3}$ in the original system with respect to an adversary who corrupts all (up to either t_- or t_+) processors of the remaining processor set $\Pi_{(k+2) \bmod 3}$ in an admissible way.

Let now π_0 and π_n be initialized with different inputs. We now argue that, for each run of the new system, there are at least two pairs $\Pi_k \cup \Pi_{(k+1) \bmod 6}$ ($0 \leq k \leq 5$) such that the conditions of two-threshold broadcast are not satisfied for them:

By the validity property of two-threshold broadcast, the at least $n - t_v$ players $p_i \in \Pi_5 \cup \Pi_0$ must compute $y_i = x_0$ whereas the players $p_i \in \Pi_2 \cup \Pi_3$ must compute $y_i = x_n = 1 - x_0$. By the consistency property of two-threshold broadcast, the at least $n - t_c$ players $p_i \in \Pi_1 \cup \Pi_2$ must compute $y_i = 1 - x_0$ whereas the players $p_i \in \Pi_4 \cup \Pi_5$ must compute $y_i = x_0$. By either the validity or the consistency property of two-threshold broadcast, the at least $n - t_+$ players $p_i \in \Pi_0 \cup \Pi_1$ must compute $y_i = y_0 = x_0$ whereas the players $p_i \in \Pi_3 \cup \Pi_4$ must compute $y_i = y_n = 1 - x_0$.

Hence, for any possible run of the new system on inputs x_0 and $x_n = 1 - x_0$ it holds that, chosen a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ of adjacent processor sets uniformly at random, the probability that the conditions for two-threshold broadcast are violated for this pair is at least $\frac{1}{3}$.

In particular, there is a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ in the new system such that, over all possible runs on inputs $x_0 = 0$ and $x_n = 1$ the probability that the conditions for two-threshold broadcast are violated for $(\Pi_k, \Pi_{(k+1) \bmod 6})$ is at least $\frac{1}{3}$.

If the adversary is unbounded, given any protocol Ψ, she can compute such a pair $(\Pi_k, \Pi_{(k+1) \bmod 6})$ and act accordingly by corrupting the processors in $\Pi_{(k+2) \bmod 3}$ in the original system, hence forcing the protocol to fail on input

$$x_0 = \begin{cases} 0 & \text{, if } 0 \in \{k, k+1\} \text{ , and} \\ 1 & \text{, else ,} \end{cases}$$

with a probability of at least $\frac{1}{3}$.

If the adversary is computationally bounded then she can still make the protocol fail with a probability of at least $\frac{1}{6}$. □

Finally, including the consensus variant of Byzantine agreement, we get the following

Theorem 6.13 (Refinement of Theorem 6.1). *In model $\mathcal{M}_{\text{aut}}$, perfectly secure broadcast with extended validity and consistency detection (ExtValBC$^+$) is efficiently achievable if either $t_c = 0$ or $t_c + 2t_v < n$ (with $t_v \geq t_c$). If these bounds are not satisfied then not even computationally secure broadcast with extended validity (ExtValBC) is achievable in model $\mathcal{M}_{\text{sec}}$.*

In model $\mathcal{M}_{\text{sec}}$, unconditionally secure broadcast with extended consistency and validity detection (ExtConsBC$^+$) or, more generally, detectable precomputation is efficiently achievable if either $t_v = 0$ or $t_v + 2t_c < n$ (with $t_c \geq t_v$). If these bounds are not satisfied then not even computationally secure broadcast with extended consistency (ExtConsBC) is achievable. If $t_c \geq n/3$ then there is no deterministic protocol for ExtConsBC.

With respect to the corresponding consensus variants, the same bounds hold except that $t_v < n/2$ is additionally necessary.

Proof. The theorem follows from Theorems 6.2 and 6.6 (achievability of ExtValBC$^+$), Theorems 6.9 and 6.11 (achievability of ExtConsBC$^+$), Theorem 6.12 (impossibility of TTBC), and [Lam83] (impossibility of deterministic ExtConsBC for $t_c \geq n/3$). □

6.7 Observations and applications

The given protocols for ExtConsBC$^+$ or detectable precomputation with respect to thresholds t_c and t_v were based on protocols for ExtValBC$^+$

with respect to thresholds $t'_v = t_c$ and $t'_c = t_v$. We now show that also the converse is possible in a very simple way: to obtain a protocol for ExtValBC$^+$ from any implementation of ExtConsBC$^+$.

Lemma 6.14. *In standard model $\mathcal{M}_{\mathrm{sec}}$, the achievability of ExtValBC$^+$ for thresholds t'_v and $t'_c \leq t'_v$ satisfying $t'_c + 2t'_v < n$ or $t'_c = 0$ implies the achievability of ExtConsBC$^+$ for thresholds $t_c = t'_v$ and $t_v = t'_c$. In standard model $\mathcal{M}_{\mathrm{aut}}$, the achievability of ExtConsBC$^+$ for thresholds t'_c and $t'_v \leq t'_c$ implies the achievability of ExtValBC$^+$ for thresholds $t_v = t'_c$ and $t_c = t'_v$;*

Proof.
$ExtValBC^+ \Rightarrow ExtConsBC^+$: This direction follows form the constructions given in the previous section.

$ExtConsBC^+ \Rightarrow ExtValBC^+$: Suppose that ExtConsBC$^+$ is achievable with respect to thresholds t'_c and t'_v. Then ExtValBC$^+$ with sender p_s and thresholds $t_v = t'_c$ and $t_c = t'_v$ can be achieved in the following way:

- The sender sends his input value x_s to all players; player p_i receives value y_i^0.
- The players participate in an instance of ExtConsBC$^+$ with sender p_s with respect to thresholds $t'_c = t_v$ and $t'_v = t_c$; player p_i receives output y_i^1 and grade g_i.
- Player p_i decides $y_i := y_i^{g_i}$.

Consistency: If $f \leq t_c = t'_v$ players are corrupted then, by the validity and consistency properties of ExtConsBC$^+$, there is a value y such that every correct player p_i computes $y_i^1 = y$ and $g_i = 1$; and if the sender is correct then $y = x_s$. Finally, $y_i = y_i^1 = y$ is computed.

Validity: If $f \leq t_v = t'_c$ players are corrupted then, by the consistency property of ExtConsBC$^+$, all correct players p_i receive the same output $y_i^1 = y$ and the same grade $g_i = g \in \{0, 1\}$.

If $g = 1$ then every correct player p_i decides on $y_i := y_i^1 = y$ and, by the validity-detection property of ExtConsBC$^+$, it follows that $y_i^1 = y = x_s$ whenever the sender is correct. If $g = 0$ then every correct player p_i decides on $y_i := y_i^0$, and validity is trivially satisfied.

Consistency Detection: If $f \leq t_v = t'_c$ and any correct player p_i computes $g_i = 1$ then, by the consistency property of ExtConsBC$^+$, every correct player p_j computes $y_j = y_j^1 = y_i^1$. □

6.7.1 Detectable multi-party computation

Detectable precomputation immediately allows to turn any protocol Ψ (e.g., a protocol for multi-party computation) in model $\mathcal{M}_{\text{aut}}^{\text{bc}}$ (or $\mathcal{M}_{\text{sec}}^{\text{bc}}$) into a "detectable version" for standard model $\mathcal{M}_{\text{aut}}$ (or $\mathcal{M}_{\text{sec}}$) without broadcast channels. For the case that $f \leq t_v$ players are corrupted this transformation preserves any security properties of Ψ except for zero-error. For the case that $f \leq t_c$ players are corrupted the transformation still preserves any security properties of Ψ except for zero-error and robustness. Robustness is lost since detectable precomputation cannot guarantee validity for t_c (at least for the interesting cases where $t_c \geq n/3$). Zero-error is lost since there is no deterministic protocol for detectable precomputation as follows from Lamport's result [Lam83].

Theorem 6.15 ([FHHW03]). *Given any protocol Ψ in model $\mathcal{M}_{\text{sec}}^{\text{bc}}$ secure against t corrupted players (for any t), for any $t_c \leq t$ and $t_v \leq t_c$ such that either $t_v = 0$ or $t_v + 2t_c < n$, there is a protocol Ψ' in model $\mathcal{M}_{\text{sec}}$ with respect to thresholds t_c and t_v such that all security properties of Ψ except for zero-error are guaranteed when up to $f \leq t_v$ players are corrupted, and all security properties of Ψ except for robustness and zero-error are guaranteed when up to $f \leq t_c$ players are corrupted.*

Proof. Such a protocol Ψ' can be obtained by first running a protocol for detectable precomputation for broadcast which is possible exactly with the stated bounds for t_c and t_v. If it is successful, then Ψ is run, replacing every call to the broadcast channel with an execution of Dolev-Strong Protocol 4.10 with help of the precomputed PKI. □

In particular, it is possible to define the "detectable" version of multi-party computation.

Definition 6.9 (Detectable precomputation for MPC). *Let Ψ be an MPC protocol among P in a model assuming broadcast, model $\mathcal{M}_*^{\text{bc}} \in \{\mathcal{M}_{\text{aut}}^{\text{bc}}, \mathcal{M}_{\text{sec}}^{\text{bc}}\}$, and let $\mathcal{M}_* \in \{\mathcal{M}_{\text{aut}}, \mathcal{M}_{\text{sec}}\}$ be the same model as $\mathcal{M}_*^{\text{bc}}$ but without broadcast. A protocol among P where every player $p_i \in P$ computes some private data Δ_i and finally decides on a decision bit $g_i \in \{0,1\}$ achieves* detectable precomputation for MPC with Ψ *with respect to thresholds t_c and t_v ($t_c \geq t_v$), and t, if it satisfies the following conditions:*

- **Robustness:** *If at most $f \leq t_v$ players are corrupted then the correct players accept ($g_i = 1$).*
- **Correctness:** *If $f \leq t_c$ then all correct players commonly accept ($g_i = 1$) or commonly reject ($g_i = 0$) the protocol; moreover, if the private data*

Δ_i held by all correct players is inconsistent in the sense that it does not guarantee for MPC secure against t corrupted players in model $\mathcal{M}_*$ then the correct players reject ($g_i = 0$).

Independence: *At the time of the precomputation, a correct player does not yet need to know his input values for the later multi-party computations.*

$\diamond$

Regarding the results in [Bea89, RB89, CDD+99], detectable precomputation for broadcast trivially allows for detectable MPC such that only robustness is lost since non-zero error is necessary for multi-party computation secure against $t \geq n/3$ corrupted players [DDWY93]. Thus, we get the following corollary of Theorem 6.15.

Theorem 6.16 ([FHHW03]). *Let Ψ be the MPC protocol in [CDD+99] for model $\mathcal{M}_{\text{sec}}^{\text{bc}}$ unconditionally secure against a faulty minority of corrupted players. In model $\mathcal{M}_{\text{sec}}$, detectable precomputation for unconditionally secure MPC with Ψ among n players with respect to thresholds t_c and t_v ($t_v \leq t_c$), and t, is efficiently achievable if $t_v + 2t_c < n \ \vee \ t_v = 0$ and $t < n/2$.*

For the case that $t_v > 0$ and $t_v + 2t_c \geq n$, and $t \geq n/3$, detectable precomputation for MPC is not even achievable with respect to computational security.

Proof. Achievability follows from [CDD+99] and Theorems 6.9 and 6.11. Impossibility follows from Theorem 6.12 (impossibility of TTBC for $t_v > 0$ and $t_v + 2t_c \geq n$) and from the impossibility of broadcast for $t \geq n/3$. $\square$

Alternatively, for example the protocol in [Gol01b] for MPC without robustness nor fairness in model $\mathcal{M}_{\text{sec}}^{\text{bc}}$ computationally secure against $t < n$ corrupted players can be detectably precomputed for with help of detectable precomputation Protocol 6.8 or Protocol 6.6.

Corollary 6.17 ([FHHW03]). *Let Ψ be the non-robust MPC protocol without fairness in [Gol01b] for model $\mathcal{M}_{\text{sec}}^{\text{bc}}$ computationally secure against $t < n$ statically corrupted players. In model $\mathcal{M}_{\text{sec}}$, detectable precomputation for computationally secure MPC with Ψ among n players is efficiently achievable if $t_v = 0$ or $t_v + 2t_c < n$.*

Proof. Achievability follows from [Gol01b] and Theorem 6.8. $\square$

In [GL02], for the special case of computational security and $t_v = 0$, Goldwasser and Lindell improved over the efficiency of the given reductions by relaxing the security properties to not demanding agreement on the outcome, and not demanding fairness for the case where $t < n/2$.

6.7.2 Involving quantum channels

Unconditionally secure detectable precomputation for broadcast or MPC with respect to threshold $t_v = 0$ can even be achieved in a slightly weaker model than with secure channels, namely in a model with classical authenticated channels and insecure quantum channels. Since every player has the possibility to initiate rejection of the precomputation, we can apply quantum key agreement [BB84] before the protocol for detectable precomputation or MPC. Whenever any quantum channel between two correct players were eavesdropped they would detect it and just initiate rejection of the whole precomputation.

Theorem 6.18 ([FGMR02, FGH+02]). *In model $\mathcal{M}_{\mathrm{aut}}$ with additional insecure pairwise quantum channels among the players, efficient and unconditionally secure detectable precomputation for broadcast among n players is achievable with respect to $t_v = 0$ and $t_c < n$. In the same setting, efficient and unconditionally secure detectable precomputation for MPC is achievable with respect to $t_v = 0$ and $t_c < n/2$.*

Proof. The theorem follows from Theorem 6.9, the protocol in [BB84], and the remarks above. □

6.7.3 Ad-hoc computations and PKI setup

For all previous "detectable protocols" it was implicitly assumed that all players start the protocol in the same communication round. This requires agreement among the players on which protocol is to be run and on a point of time when the protocol is to be started (cf. Sections 2.3.4 and 3.1.3).

We now show that this assumption is not necessary, i.e., such a protocol exists for the synchronous model without a global clock and not even agreement is required on the player set among which the protocol will be run. This problem can be seen as a very strong version of the firing squad problem — but additionally allowing the correct players to abort in unison.

We describe a protocol wherein a player p_I who shares authenticated (or secure) channels with all members of a player set P' to (unexpectedly) initiate a protocol among the players in $P = P' \cup \{p_I\}$ for a detectable precomputation. Let such a player p_I be called the *initiator* and the players in P' be called the *initiees* of the protocol. The following protocol description is split into the initiator's part and the initiees' part.

Protocol 6.11 `InitAdHocComp`(P) [Initiator p_I]

1. Send to P' an initiation message containing a unique random session identifier id, the specification of the player set P' and of a multi-party computation protocol Ψ among player set $P = P' \cup \{p_I\}$.
2. Perform detectable precomputation Protocol 6.8 among P' to precompute for all broadcast invocations required by protocol Ψ. Thereby, combine each message with session identifier id. In the final "broadcast round", instead of broadcasting the value G_I to indicate whether all conditional gradecast protocols achieved broadcast, the value $G_I \wedge S_I$ is broadcast where S_I indicates whether all players in P' synchronously entered the precomputation protocol and always used session identifier id.
3. Accept and execute protocol Ψ (with identifier id) if and only if $G_I \wedge S_I$ and all players in P' broadcasted their acceptance at the end.

Protocol 6.12 `AdHocComp`(P) [Initiee p_i]

1. Upon receipt of an initiation message by an initiator p_I:
   • decide whether you are interested to execute an instance of Ψ among player set P.
   • check that the specified id is not being used in any concurrent invocation.
   • check whether $p_i \in P'$.
   • check whether there are authenticated (or secure) channels between p_i and all other players in $P' \cup \{p_I\}$ as required by protocol Ψ.
2. If all checks in step 1 were positive then perform detectable precomputation Protocol 6.8 to precompute for all broadcast invocations required by protocol Ψ. Thereby, combine each message with session identifier id and do not initiate any ad-hoc computation with the same identifier id. In the final "broadcast round", instead of broadcasting the value G_i to indicate whether all conditional gradecast protocols achieved broadcast, the value $G_i \wedge S_i$ is broadcast, where S_i indicates whether all players in P' synchronously entered the precomputation protocol and always used session identifier id.
3. Accept and execute protocol Ψ (with identifier id) if and only if $G_i \wedge S_i$ and all players in P' broadcasted their acceptance at the end.

Note that the first check in step 1 of the initiees' Protocol 6.12 implicitly prevents the adversary from "spamming" players with initiations in order to overwhelm a player with work load. A player can simply ignore initiations without affecting consistency among the correct players.

Theorem 6.19 ([FGH+02]). *Suppose there is a player set P' and a player p_I such that p_I shares authenticated (or secure) channels with every player in P'*

(whereas no additional channels are assumed between the players in P'). Then p_I can initiate a protocol among player set $P = P' \cup \{p_I\}$ that achieves the following properties for $t < n$:

1. *All correct players in P either commonly accept or reject the protocol (instead of rejecting it is also possible that a player ignores the protocol, which is an implicit rejection). If they accept, then broadcast or multi-party computation will be achievable among P (with everybody knowing this fact).*

2. *If all players in P are correct and all players are connected by pairwise authenticated (or secure) channels then all players accept.*

Proof.

1. First, suppose that all correct players accept the precomputation. Then all correct players share pairwise authenticated (or secure) channels and synchronously performed a detectable precomputation with respect to the same protocol Ψ — using the same session identifier id and accepting the outcome of the precomputation. Thus, all correct players synchronously execute protocol Ψ (using the same identifier) which has been reliably precomputed for.

   Second, suppose that correct player p_j rejects the precomputation or ignores it because he did not get an invitation message. If he ignores it then all non-ignoring correct players notice this and reject. If he rejects then he sent or received a rejection message during the final "broadcast round" of the precomputation. Thus, every other non-ignoring player has either already rejected before this round or reliably receives the same rejection message during the final round, and thus also rejects.

2. If all players in P are correct and are connected by pairwise authenticated (or secure) channels then they all accept the precomputation unless two different precomputations with the same session identifier id have been initiated during the same round. Having id be randomly chosen from a large enough domain, such a collision happens with negligible probability.

□

Note that protocol Ψ can be omitted from this protocol, resulting in an "optimistic" protocol for the ad-hoc setup of a PKI without a trusted party nor broadcast channels.

Chapter 7

Concluding Remarks

In this thesis, we have generalized some fundamental standard models for Byzantine agreement and multi-party computation for synchronous networks in several ways.

7.1 Standard communication

A first direction addressed the resilience of protocols with respect to standard pairwise communication. It could be shown that the generalization of standard security definitions often allows for a level of resilience that is strictly better than previously achievable. Thereby, two different models were considered, the model where a PKI is shared among the players, and the model where the players do not share a PKI.

Shared PKI. For the model where a PKI is shared among the players, the following main result was obtained (Section 4.2.2). PKI-based security of Byzantine agreement can be augmented with unconditional security up to a certain non-trivial level: Given two thresholds t_σ and t_u ($t_\sigma \geq t_u$), the aim is to have a protocol for Byzantine agreement that is as secure as the underlying PKI if up to t_σ players are corrupted and, independently, unconditionally secure if up to t_u players are corrupted.

It could be shown that, with respect to thresholds t_σ and t_u, broadcast is achievable if and only if $2t_u + t_\sigma < n$ and consensus is achievable if and only if $2t_u + t_\sigma < n$ and $2t_\sigma < n$. Efficient protocols are given for every case where $2t_\sigma < n$. As an implication, multi-party computation com-

putationally secure against $t < n/2$ corrupted players (which is optimal) can additionally tolerate $t \leq n/4$ corrupted players unconditionally.

The following problem remains open:

- For the case that $2t_\sigma \geq n$ it is not yet known whether efficient broadcast is achievable.

No shared PKI. By separating the resilience of Byzantine agreement with respect to the two orthogonal conditions "consistency" (threshold t_c) and "validity" (threshold t_v), protocols with interesting new properties can be constructed for the model where no consistent PKI is given among the players. The following main result was obtained (Chapter 6). TTBC with respect to thresholds t_v and t_c is achievable if and only if

$$t_v = 0 \vee t_c = 0 \vee (2t_v + t_c < n \wedge 2t_c + t_v < n) \; .$$

TTBC can be strengthened in a way that allows the players to additionally detect whether both validity and consistency, have been achieved. Some implications of this result are listed below.

- As a special case, broadcast among three players is possible such that both conditions validity and consistency, are satisfied if no player is corrupted, but that consistency is additionally satisfied independently of the number of corrupted players; furthermore, the players detect whether validity has been achieved.
- In a scenario where the adversary does not have permanent control over the corrupted players, broadcast secure against any number of corrupted players is achievable from scratch. Furthermore, a first phase wherein the adversary does not corrupt any player can be reliably detected and exploited such that broadcast with arbitrary resilience will be possible in the future — this is achieved by so-called "detectable precomputation".
- Detectable precomputation allows for multi-party computation secure against faulty minorities without broadcast channels after the first phase without corruption.
- Any protocol for a model with secure pairwise communication and broadcast channels can be generically transformed into a protocol for the same model but without broadcast channels satisfying all security conditions of the original protocol except for robustness and zero-error.

For ExtValBC$^+$, relatively efficient, perfectly secure protocols could be given ($\mathcal{B} = O(n^3)$). For ExtConsBC$^+$ (and detectable precomputation), relatively efficient, computationally secure protocols could be given ($\mathcal{B} = O(n^4)$); however, the given protocols with unconditional security are quite costly ($\mathcal{B} = \Omega(n^7)$ and $\mathcal{B} = \Omega(n^{11})$).

The following questions remain open:

- Do protocols for unconditionally secure ExtConsBC$^+$ and detectable precomputation exist that have low communication complexities?
- Are there other applications for detectable precomputation than given here?

7.2 Modular reductions and extended communication

A second direction addressed the reducibility among standard Byzantine agreement and some of its variations such as partial broadcast or weak broadcast. It was shown that broadcast among each subset of $b \geq 2$ players (BC$_b$) allows for global broadcast secure against $t < \frac{b-1}{b+1}n$ corrupted players; and that this bound is tight. Moreover, the well-known primitives multi-send, weak broadcast, and graded broadcast were captured by a single, parameterized definition, b-set-neighboring among n players (N_n^b), that naturally extends over these special cases where N_n^2 is multi-send, N_n^3 is weak broadcast, and N_n^4 is graded broadcast. It was shown that N_n^b and BC$_b$ are equivalent. Some implications of these results are stated below.

- In order to demonstrate the achievability of broadcast for $t < n/2$ with respect to a certain model it is sufficient to construct a weak-broadcast protocol for $t \geq \lceil n/3 \rceil$.
- In order to demonstrate the achievability of broadcast for $t < n/2$ with respect to a certain model it is sufficient to show that weak broadcast is achievable among every triplet of players.
- Instead of global broadcast, broadcast among each triple of players is sufficient for multi-party computation unconditionally secure against faulty minorities.

Whereas the given constructions and reductions with respect to $b \leq 3$ are all efficient, those with respect to $b \geq 4$ are only efficient if $n - b = O(1)$.

The following questions remain open:

- Given BC_b, is it possible to construct efficient protocols for broadcast among all n players with respect to $t < \frac{b-1}{b+1}n$? If not, for which bounds on t are there efficient protocols?
- Is full connectivity of partial broadcast, i.e., that every subset of b players shares broadcast channels, required in order to achieve optimal resilience? If so, what resilience can be achieved with lower connectivity? Note that the necessity of full connectivity would imply that efficient global broadcast is impossible for most cases, e.g., for all cases where $b = \Omega(n)$ and $n - b = \Omega(n)$.

7.3 Future directions

Although the main focus of this thesis was on *generalizing* existing standard models in Byzantine agreement, two major *restrictions* have been made throughout: assuming synchronous instead of asynchronous networks, and assuming a threshold adversary instead of a general non-threshold adversary. Asynchronous networks are more realistic to assume (e.g., the Internet is not synchronous) and yet, often allow for the same resilience as synchronous networks — although typically at the cost of less efficient protocols. The notion of a general adversary strictly subsumes the notion of a threshold adversary. Thus, the following tasks remain to be solved:

- to transform the given results to the asynchronous model,
- to extend the given results to the general adversary model, and
- to combine the given model generalizations, e.g., considering detectable precomputation based on partial broadcast.

Bibliography

[AFM99] B. Altmann, M. Fitzi, and U. Maurer. Byzantine agreement secure against general adversaries in the dual failure model. In *13th International Symposium on Distributed Computing (DISC '99)*, volume 1693 of *Lecture Notes in Computer Science*, pp. 123–137. Springer-Verlag, 1999.

[AW98] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. McGraw-Hill Publishing Company, 1998.

[BB84] C. H. Bennett and G. Brassard. An update on quantum cryptography. In *Advances in Cryptology: CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pp. 475–480. Springer-Verlag, 1984.

[BB90] J. Bos and B. Boer. Detection of disrupters in the DC protocol. In *Advances in Cryptology: EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pp. 320–327. Springer-Verlag, 1990.

[BDDS92] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. *Information and Computation*, 97(2):205–233, Apr. 1992.

[BDP97] P. Berman, K. Diks, and A. Pelc. Reliable broadcasting in logarithmic time with Byzantine link failures. *Journal of Algorithms*, 22(2):199–211, Feb. 1997.

[BE03] M. Ben-Or and R. El-Yaniv. Optimally-resilient interactive consistency in constant time. *Distributed Computing*, 16(2), May 2003.

[Bea89] D. Beaver. Multiparty protocols tolerating half faulty processors. In *Advances in Cryptology: CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pp. 560–572. Springer-Verlag, 1989.

[Bea91] D. Beaver. Foundations of secure interactive computation. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pp. 377–391. Springer-Verlag, 1991.

[Ben83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *Proceedings of the 2nd ACM Symposium on Principles of Distributed Computing (PODC '83)*, pp. 27–30, 1983.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pp. 103–112. ACM Press, 1988.

[BG89a] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS '89)*, pp. 468–473, 1989.

[BG89b] P. Berman and J. Garay. Asymptotically optimal distributed consensus. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pp. 80–94. Springer-Verlag, 1989.

[BG93] P. Berman and J. A. Garay. Cloture votes: $n/4$-resilient distributed consensus in $t+1$ rounds. *Mathematical Systems Theory*, 26(1):3–19, 1993.

[BGP89] P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS '89)*, pp. 410–415, 1989.

[BGP92a] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. In *Computer Science Research*, pp. 313–322. Plenum Publishing Corporation, 1992.

[BGP92b] P. Berman, J. A. Garay, and K. J. Perry. Optimal early stopping in distributed consensus (extended abstract). In *Distributed Algorithms, 6th International Workshop, WDAG '92*, volume 647 of *Lecture Notes in Computer Science*, pp. 221–237. Springer-Verlag, 1992.

[BGR96] M. Bellare, J. A. Garay, and T. Rabin. Distributed pseudo-random bit generators : A new way to speed-up shared coin tossing. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing (PODC '96)*, pp. 191–200. ACM Press, 1996.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pp. 1–10. Springer-Verlag, 1988.

[BL87] J. E. Burns and N. A. Lynch. The Byzantine firing squad problem. In F. P. Preparata, editor, *Advances in Computing Research, Parallel and Distributed Computing*, volume 4, pp. 147–161. JAI Press, Inc., Greenwich, Conn., 1987.

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

[BMM99] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology: CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pp. 80–97. Springer-Verlag, 1999.

[Bor96] M. Borcherding. Levels of authentication in distributed agreement. In *Distributed Algorithms, 10th International Workshop, WDAG '96*, volume 1151 of *Lecture Notes in Computer Science*, pp. 40–55. Springer-Verlag, 1996.

[BPW91] B. Baum-Waidner, B. Pfitzmann, and M. Waidner. Unconditional Byzantine agreement with good majority. In *8th Annual Symposium on Theoretical Aspects of Computer Science*, volume 480 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, 1993.

[Bra87a] G. Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, Nov. 1987.

[Bra87b] G. Bracha. An $O(\log n)$ expected rounds randomized Byzantine generals protocol. *Journal of the Association for Computing Machinery*, 34(4):910–920, Oct. 1987.

[BS94] D. Beaver and N. So. Global, unpredictable bit generation without broadcast. In *Advances in Cryptology: EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pp. 424–434. Springer-Verlag, 23–27 May 1994.

[BT85] G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824–840, Oct. 1985.

[Can95] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute of Science, 1995.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pp. 11–19. ACM Press, 1988.

[CD89] B. Chor and C. Dwork. Randomization in Byzantine agreement. *ADVCR: Advances in Computing Research*, 5, 1989.

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, 1999.

[CDDS89] B. A. Coan, D. Dolev, C. Dwork, and L. Stockmeyer. The distributed firing squad problem. *SIAM Journal on Computing*, 18(5):990–1012, Oct. 1989.

[CDG87] D. Chaum, I. B. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology: CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pp. 87–119. Springer-Verlag, 1987.

[CDM00] R. Cramer, I. Damgård, and U. Maurer. General secure multiparty computation from any linear secret-sharing scheme. In *Advances in Cryptology: EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pp. 316–334. Springer-Verlag, 2000.

[CDN01] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pp. 279–298. Springer-Verlag, 2001.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pp. 383–395, 1985.

[Cha88] D. Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[Cha89] D. Chaum. The spymasters double-agent problem. In *Advances in Cryptology: CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pp. 591–602. Springer-Verlag, 1989.

[Chv79] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287, 1979.

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC '00)*, pp. 123–132, 2000.

[CLM00] J. Considine, L. A. Levin, and D. Metcalf. Byzantine agreement with bounded broadcast. http://arxiv.org/abs/cs.DC/0012024, 2000.

[CMS89] B. Chor, M. Merritt, and D. B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *Journal of the ACM*, 36(3):591–614, July 1989.

[CR90] D. Chaum and S. Roijakkers. Unconditionally-secure digital signatures. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pp. 206–214. Springer-Verlag, 1990.

[CR93] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience (extended abstract). In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC '93)*, pp. 42–51, 1993.

[Cra99] R. Cramer. Introduction to secure computation. In *Lectures on data security: modern cryptology in theory and practice*, volume 1561 of *Lecture Notes in Computer Science*, pp. 16–62. Springer-Verlag, 1999.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology: CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pp. 13–25. Springer-Verlag, 1998.

[CW92] B. A. Coan and J. L. Welch. Modular construction of a Byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, Mar. 1992.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993.

[DFF+82] D. Dolev, M. J. Fischer, R. Fowler, N. A. Lynch, and H. R. Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 52(3):257–274, Mar. 1982.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.

[DLM82] R. A. DeMillo, N. A. Lynch, and M. J. Merritt. Cryptographic protocols. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC '82)*, pp. 383–400, 1982.

[DM90] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, 1990.

[Dol82] D. Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[DR85] D. Dolev and R. Reischuk. Bounds on information exchange for Byzantine agreement. *Journal of the ACM*, 32(1):191–204, Jan. 1985.

[DR98] J. Daemen and V. Rijmen. AES proposal: Rijndael. NIST AES Proposal, June 1998.

[DRS82] D. Dolev, R. Reischuk, and H. R. Strong. 'Eventual' is earlier than 'immediate'. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pp. 196–203, 1982.

[DRS90] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in Byzantine agreement. *Journal of the ACM*, 37(4):720–741, Oct. 1990.

[DS82] D. Dolev and H. R. Strong. Polynomial algorithms for multiple processor agreement. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC '82)*, pp. 401–407, 1982.

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

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology: CRYPTO '84*, pp. 10–18. Springer-Verlag, 1985.

[Fel89] P. Feldman. Asynchronous Byzantine agreement in constant expected time. Manuscript, 1989.

[FG02] M. Fitzi and J. Garay. Efficient and player-optimal protocols for strong and differential consensus. Manuscript, 2002.

[FGH+02] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith. Detectable Byzantine agreement secure against faulty majorities. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02)*, pp. 118–126, 2002.

[FGM01] M. Fitzi, N. Gisin, and U. Maurer. Quantum solution to the Byzantine agreement problem. *Physical Review Letters*, 87(21):7901–1–7901–4, Nov. 2001.

[FGMO01] M. Fitzi, J. A. Garay, U. Maurer, and R. Ostrovsky. Minimal complete primitives for unconditional multi-party computation. In *Advances in Cryptology: CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pp. 80–100. Springer-Verlag, 2001.

[FGMO02] M. Fitzi, J. A. Garay, U. Maurer, and R. Ostrovsky. Minimal complete primitives for unconditional multi-party computation. Journal version — to appear, 2002.

[FGMR02] M. Fitzi, N. Gisin, U. Maurer, and O. von Rotz. Unconditional Byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology: EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pp. 482–501. Springer-Verlag, 2002.

[FH96] M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, Fall 1996.

[FH02] M. Fitzi and T. Holenstein. Amplified security for secure multi-party computation. Manuscript, 2002.

[FHHW03] M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-threshold broadcast and detectable multi-party computation. Accepted to EUROCRYPT '03, Lecture Notes in Computer Science, Springer Verlag., 2003.

[FHM98] M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *Advances in Cryptology: CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pp. 121–136. Springer-Verlag, 1998.

[FHM99] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *Advances in Cryptology: ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*, pp. 232–246. Springer-Verlag, 1999.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

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

[FLM86] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1:26–39, 1986.

[FLP85] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty processor. *Journal of the ACM*, 32(2):374–382, 1985.

[FM85] P. Feldman and S. Micali. Byzantine agreement in constant expected time (and trusting no one). In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pp. 267–276. IEEE Computer Society Press, 1985.

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

[FM98] M. Fitzi and U. Maurer. Efficient Byzantine agreement secure against general adversaries. In *12th International Symposium on Distributed Computing (DISC '98)*, volume 1499 of *Lecture Notes in Computer Science*, pp. 134–148, 1998.

[FM00a] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pp. 494–503, 2000.

[FM00b] M. Fitzi and U. Maurer. Global broadcast by broadcasts among subsets of players. In *2000 IEEE International Symposium on Information Theory (ISIT '00)*, p. 267, 2000.

[Fra93] M. K. Franklin. *Complexity and Security of Distributed Protocols*. PhD thesis, Columbia University, 1993.

[FW00] M. Franklin and R. N. Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, Winter 2000.

[FY95] M. Franklin and M. Yung. Secure hypergraphs: Privacy from partial broadcast (extended abstract). In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC '95)*, pp. 36–44, 1995.

[GHY87] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Advances in Cryptology: CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pp. 135–155. Springer-Verlag, 1987.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pp. 11–15. Springer-Verlag, 1990.

[GL02] S. Goldwasser and Y. Lindell. Secure computation without agreement. In *16th International Symposium on Distributed Computing (DISC '02)*, volume 2508 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[GLR95] L. Gong, P. Lincoln, and J. Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults, 1995.

[GM98] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM Journal on Computing*, 27(1):247–290, Feb. 1998.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[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)*, pp. 218–229. ACM Press, 1987.

[GMY95] Z. Galil, A. Mayer, and M. Yung. Resolving message complexity of Byzantine agreement and beyond. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS '95)*, pp. 724–733, 1995.

[Gol01a] O. Goldreich. *Foundations of Cryptography, Volume 1*. Cambridge University Press, 2001.

[Gol01b] O. Goldreich. Secure multi-party computation, working draft, version 1.3, June 2001.

[GP90] O. Goldreich and E. Petrank. The best of both worlds: Guaranteeing termination in fast Byzantine agreement protocols. *Information Processing Letters*, 36:45–49, Oct. 1990.

[GP92] J. A. Garay and K. J. Perry. A continuum of failure models for distributed computing. In *Distributed Algorithms, 6th International Workshop, WDAG '92*, volume 647 of *Lecture Notes in Computer Science*, pp. 153–165. Springer-Verlag, 1992.

[GRR98] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC '98)*, pp. 101–111, 1998.

[GY89] R. L. Graham and A. C. Yao. On the improbability of reaching Byzantine agreements. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pp. 467–478. ACM Association for Computing Machinery, 1989.

[Had83] V. Hadzilacos. Byzantine agreement under restricted types of failures (not telling the truth is different from telling lies). Technical Report TR.CRCT TR-1, Harvard University, 1983.

[HH93a] V. Hadzilacos and J. Y. Halpern. The failure discovery problem. *Mathematical Systems Theory*, 26(1):103–129, 1993.

[HH93b] V. Hadzilacos and J. Y. Halpern. Message-optimal protocols for Byzantine agreement. *Mathematical Systems Theory*, 26(1):41–102, 1993.

[Hir01] M. Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[HM90] J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[HM00] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, Winter 2000.

[HM01] M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. In *Advances in Cryptology: CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pp. 101–118. Springer-Verlag, 2001.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.

[Hol01] T. Holenstein. Hybrid broadcast protocols. Master's thesis, ETH Zürich, October 2001. Supervised by Matthias Fitzi.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS '00)*, pp. 294–304, 2000.

[ISN87] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proceedings IEEE Globecom '87*, pp. 99–102, 1987.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pp. 20–31, 1988.

[Kil91] J. Kilian. A general completeness theorem for two-party games. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC '91)*, pp. 553–560, 1991.

[Kil00] J. Kilian. More general completeness theorems for secure two-party computation. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pp. 316–324, 2000.

[Kob94] N. Koblitz. *A Course in Number Theory and Cryptography*. Graduate texts in mathematics. Springer-Verlag, second edition, 1994.

[KY84] A. Karlin and A. C. Yao. Manuscript, 1984.

[Lam83] L. Lamport. The weak Byzantine generals problem. *Journal of the ACM*, 30(3):668–676, July 1983.

[LF82] L. Lamport and M. J. Fischer. Byzantine generals and transaction commit protocols. Technical Report Opus 62, SRI International (Menlo Park CA), TR, 1982.

[LLR02] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pp. 514–523. ACM Press, 2002.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Lyn96] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, 1996.

[Mau93] U. Maurer. Protocols for secret key agreement by public discussion based on common information. In *Advances in Cryptology: CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pp. 461–470. Springer-Verlag, 1993.

[MOV97] A. J. Menezes, P. Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

[MP91] F. J. Meyer and D. K. Pradhan. Consensus with dual failure modes. *IEEE Transactions on Parallel and Distributed Systems*, 2(2):214–222, Apr. 1991.

[MR91] S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pp. 253–267. Springer-Verlag, 1991.

[MW88] Y. Moses and O. Waarts. Coordinated traversal: (t+1)-round Byzantine agreement in polynomial time. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (FOCS '88)*, pp. 246–255, 1988.

[MW94] Y. Moses and O. Waarts. Coordinated traversal: $(t+1)$-round Byzantine agreement in polynomial time. *Journal of Algorithms*, 17(1):110–156, July 1994.

[Nei94] G. Neiger. Distributed consensus revisited. *Information Processing Letters*, 49(4):195–201, February 1994.

[Nie02] J. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology: CRYPTO '02*, Lecture Notes in Computer Science. Springer-Verlag, 2002.

[Pel92] A. Pelc. Reliable communication in networks with Byzantine link failures. *Networks: An International Journal*, 22:441–459, 1992.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.

[PW92] B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS '92)*, volume 577 of *Lecture Notes in Computer Science*, pp. 339–350. Springer-Verlag, 1992.

[PW96] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t >= n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.

[Rab83a] M. O. Rabin. Randomized Byzantine generals. In *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science (FOCS '83)*, pp. 403–409, 1983.

[Rab83b] M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, Oct. 1983.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pp. 73–85, 1989.

[Rot00] O. von Rotz. Reduktion von informationstheoretisch sicheren Konsistenzprimitiven. Master's thesis, ETH Zürich, 2000. Supervised by Matthias Fitzi.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM*, 21(2):120–126, Feb. 1978.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[Sim92] G. J. Simmons. *Contemporary Cryptology*. IEEE, 1992.

[Sti02] D. R. Stinson. *Cryptography: Theory and practice*. CRC Press, 2nd edition edition, 2002.

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

[Tou84] S. Toueg. Randomized Byzantine agreements. In *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing (PODC '84)*, pp. 163–178, 1984.

[TPS87] S. Toueg, K. J. Perry, and T. K. Srikanth. Fast distributed agreement. *SIAM Journal on Computing*, 16(3):445–457, June 1987.

[Ver26] G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *J. Am. Inst. Elec. Eng.*, 55:109–115, 1926.

[VP93] N. H. Vaidya and D. K. Pradhan. Degradable agreement in the presence of byzantine faults. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pp. 237–245. IEEE Computer Society Press, 1993.

[Wai92] M. Waidner. *Byzantinische Verteilung ohne kryptographische Annahmen trotz beliebig vieler Fehler*. PhD thesis, Universität Karlsruhe, 1992.

[WC81] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

[WD01] Y. Wang and Y. Desmedt. Secure communication in multicast channels: The answer to Franklin and Wright's question. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(2):121–135, 2001.

[WP89] M. Waidner and B. Pfitzmann. Unconditional sender and recipient untraceability in spite of active attacks — some remarks. Technical Report 5/89, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1989.

[Yao82] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pp. 160–164, 1982.

[Zam96] A. Zamsky. A randomized Byzantine agreement protocol with constant expected time and guaranteed termination in optimal (deterministic) time. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing (PODC '96)*, pp. 201–208, 1996.

Bibliography Index

[RSA78], 23, 24, 83
[Rab83a], 42, 44, 46, 71
[Rab83b], 100
[Rot00], 114
[Sch91], 24
[Sha79], 59
[Sim92], 23, 24
[Sti02], 23, 24
[TC84], 50
[TPS87], 41
[Tou84], 42, 44, 46, 85
[VP93], 127, 128
[Ver26], 23
[WC81], 24, 95
[WD01], 99
[WP89], 86
[Wai92], 94, 96
[Yao82], 55
[Zam96], 51

Glossary

$[c]$	Computational security
$[u]$	Unconditional security
$\perp$	Invalid value or symbol to represent rejection
κ	Security parameter
λ	Transferability of a pseudo-signature scheme; fraction factor (Section 5.4)
Π	Processor set
π	Processor
σ	Signature or pseudo-signature; index set (Section 5.4)
Ψ	Protocol
$\mathcal{B}$	Bit complexity
BA	Byzantine agreement
BC_b	Broadcast among b players
BC	Broadcast
C	Consensus
$\mathcal{D}$	Domain
EBA	Eventual Byzantine agreement
EIG	Exponential information gathering
ExtConsBC	Broadcast with extended consistency
$\mathrm{ExtConsBC}^+$	Broadcast with extended consistency and validity detection

ExtValBC	Broadcast with extended validity
ExtValBC$^+$	Broadcast with extended validity and consistency detection
f	Number of corrupted players that are present
g	Grade value
GBC	Graded broadcast
GC	Graded consensus
IG	Information gathering
KC	King consensus
$\log$	Logarithm to a base of size $O(1)$
$\mathcal{M}$	Model
m	Message — but often just a free variable
MPC	Multi-party computation
n	Number of players
N_n^b	b-set-neighboring among n players
P	Player set
p	Player
p_i	Local player
PK	Public key
PKI	Public-key infrastructure
Poly	Of polynomial order
$\mathrm{Polylog}$	Of polylogarithmic order
p_s	Sender (sending player) in a broadcast protocol
$\mathcal{R}$	Range or round complexity
SBA	Simultaneous Byzantine agreement
SK	Secret key
t	Corruption threshold
TTBC	Two-threshold broadcast
TTC	Two-threshold consensus
TTGC	Two-threshold graded consensus
V	Verification algorithm of a (pseudo-)signature scheme
WBC	Weak broadcast
WC	Weak consensus

Index