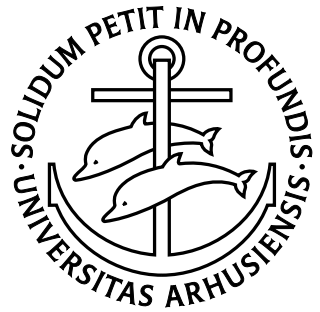
Secure Multi-Player Protocols: Fundamentals, Generality, and Efficiency

Serge Fehr

PhD Dissertation



BRICS

BRICS
Department of Computer Science
University of Århus
Denmark

June 2, 2003

Supervisors: Ronald Cramer
Ivan Damgård

Secure Multi-Player Protocols: Fundamentals, Generality, and Efficiency

A Dissertation
Presented to the Faculty of Science
of the University of Århus
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Serge Fehr
June 2, 2003

Abstract

While classically cryptography is concerned with the problem of private communication among two entities, say *players*, in modern cryptography *multi*-player protocols play an important role. And among these, it is probably fair to say that *secret sharing*, and its stronger version *verifiable secret sharing (VSS)*, as well as *multi-party computation* (MPC) belong to the most appealing and/or useful ones. The former two are basic tools to achieve better robustness of cryptographic schemes against malfunction or misuse by "decentralizing" the security from one single to a whole group of individuals (captured by the term *threshold* cryptography). The latter allows—at least in principle—to execute *any* collaboration among a group of players in a *secure* way that guarantees the correctness of the outcome but simultaneously respects the privacy of the participants.

In this work, we study three aspects of secret sharing, VSS and MPC, which we denote by *fundamentals*, *generality*, and *efficiency*. By fundamentals we mean the quest for understanding *why* a protocol works and is secure in abstract (and hopefully simple) mathematical terms. By generality we mean generality with respect to the underlying mathematical structure, in other words, minimizing the mathematical *axioms* required to do some task. And efficiency of course deals with the improvement of protocols with respect to some meaningful complexity measure.

We briefly summarize our main results. (1) We give a complete characterization of *black-box* secret sharing in terms of simple algebraic conditions on the integer sharing coefficients, and we propose a black-box secret sharing scheme with minimal expansion factor. Note that, in contrast to the classical field-based secret sharing schemes, a black-box secret sharing scheme allows to share a secret sampled from an arbitrary Abelian group and requires only black-box access to the group operations and to random group elements. Such a scheme may be very useful in the construction of threshold cryptosystems based on groups with secret order (like RSA). (2) We show that without loss of efficiency, MPC can be based on arbitrary finite rings. This is in sharp contrast to the literature where essentially all MPC protocols require a much stronger mathematical structure, namely a field. Apart from its theoretical value, this can lead to efficiency improvements since it allows a greater freedom in the (mathematical) representation of the task that needs to be securely executed. (3) We propose a unified treatment of perfectly secure linear VSS and distributed commitments (a weaker version of the former), and we show that the security of such a scheme can be reduced to a linear algebra condition. The security of all known schemes follows as corollaries whose proofs are pure linear algebra arguments, in contrast

to some hybrid arguments used in the literature. (4) We construct a new unconditionally secure VSS scheme with minimized reconstruction complexity in the setting of a dishonest minority. This improves on the reconstruction complexity of the previously best known scheme without affecting the (asymptotic) complexity of the sharing phase. In the context of MPC with *pre-processing*, our scheme allows to improve the computation phase of earlier protocols without increasing the complexity of the pre-processing. (5) Finally, we consider *commitment multiplication proofs*, which allow to prove a multiplicative relation among three commitments, and which play a crucial role in computationally secure MPC. We study a *non-interactive* solution which works in a *distributed-verifier* setting and essentially consists of a few executions of Pedersen's VSS. We simplify and improve the protocol, and we point out a (previously overlooked) property which helps to construct non-interactive proofs of *partial knowledge* in this setting. This allows for instance to prove the knowledge of ℓ out of m given secrets, without revealing which ones. We also construct a non-interactive distributed-verifier proof of *circuit satisfiability*, which—in principle—allows to prove *anything* that can be proven without giving away the proof.

Acknowledgements

First and foremost I'm deeply indebted to Ronald Cramer for having been my unofficial supervisor throughout my doctoral study. He supported me extensively, he shared his knowledge and insights with me, and he encouraged me whenever necessary. Without him, this work would not exist.

Special thanks go to Ivan Damgård for having been my (official) supervisor here in Århus, and to Ueli Maurer who introduced me to this fascinating world named "cryptography".

I'm very thankful to Yuval Ishai for many helpful comments and suggestions that helped to improve the presentation of this thesis.

I would also like to thank all my other co-authors, team mates and colleagues, Masayuki Abe, Jesus Almansa, Kasper Dupont, Stefan Dziembowski, Matthias Fitzi, Jens Groth, Martin Hirt, Clemens Holenstein, Mads Jurik, Olaf Keller, Reto Kohlas, Maciej Koprowski, Thomas Kühne, Eyal Kushilevitz, Julien Marcil, Lennart Meier, Kirill Morozov, Jesper Buus Nielsen, Bartosz Przydatek, Renato Renner, Louis Salvail, and Stefan Wolf, for many interesting discussions and amusing chats. Especially with Clemens and Louis I always had heaps of fun.

Last but definitely not least, a very special thank you goes to my family: to my mother, Heide-Marie, for putting me "onto the right track", and to my wife, Rebecca, for her love and support and for going through thick and thin with me.

Serge Fehr,
Århus, June 2, 2003.

Contents

Chapter 1

Introduction

> *Expecting the world to treat you fairly because you are a good person is a little like expecting the bull not to attack you because you are a vegetarian.*
>
> — Dennis Wholey

Classically, reaching back to the Egyptians some 4000 years ago, cryptography deals with the problem of "hidden writing", as the origin "kryptos graphia" ($\kappa\rho\upsilon\pi\tau\sigma\sigma\ \gamma\rho\alpha\phi\iota\alpha$) of the term cryptography translates: A party, Alice, wants to accord another party, Bob, with a message m such that a possible eavesdropper, Eve, who intercepts the communication between Alice and Bob, does not get to see m. The means of achieving this have of course changed drastically. The old Romans are said to have tattooed a secret message onto the shaved scalp of a slave, who then was sent to the recipient after his hair had grown again. More sophisticated, in order to privately communicate with his generals in the field, Julius Caesar encrypted his orders by substituting each letter of the message with the letter that is third-next in the alphabet (where "A" follows "Z"). In the first half of the 20th century, mechanical encryption devices where common, the most famous one being Enigma, used by the German army in World War II and broken by the Allies. Since the advent of the computer area in the second half of the 20th century, encryption has been done by complex algorithms like the Data Encryption Standard DES, and possibly based on involved mathematical techniques like the famous public-key encryption scheme RSA due to Rivest, Shamir and Adleman.

A milestone (if not to say *the* milestone) in the history of cryptography was set in 1976 by Diffie and Hellman with their ground breaking paper *New Directions in Cryptography.* They not only revolutionized the classical problem of cryptography by introducing the idea of public-key encryption, but they also extended the domain of cryptology beyond that of private communication by introducing the concept of digital signatures, the electronic equivalent of ordinary handwritten signatures. Starting with this work, researchers began to realize that there are many other interesting problems related to information security than "hidden writing". In particular, they began to also study problems that involve more than just two parties Alice and Bob (or three, counting Eve as well), but a whole group of parties, typically called *players.* We distinguish between the following two motivations.

On one hand, it was realized that certain problems simply *do* involve a group of players by their nature. A popular example for that is *secret-ballot voting*, where a group of players, voters, want to accept or reject, say, a new statute by a majority decision, in such a way that each individual vote remains private, but that nevertheless the correctness of the result of the vote can be verified by any voter or external observer. Another example is that of *sealed-bid auctions*, where some good should be sold to the highest bidder, but where the losing bidders don't want the auctioneer or the other bidders to learn their bid. Both examples can be viewed as instantiations of a very general problem, called *multi-party computation* (MPC). In MPC, a group of players want to compute an *arbitrary* specified function on their private inputs. The computation must guarantee the correctness of the result while preserving the privacy of the players' inputs, even if some of the players are malicious and behave in an arbitrary unpredictable way.

On the other hand, it was realized that one can achieve robustness of cryptographic schemes against malfunction or misuse by "spreading" security among a group of players rather than concentrating it in one single player. For example a company might want its staff to be able to sign documents on its behalf. However, giving the signing key to all (or some) staff members is very vulnerable with respect to misuse by individual members. The idea now is to *share* the signing key among the stuff members in such a way that a certain number of members have to collaborate in order to issue a signature. This concept is called *threshold cryptography*. Another, though macabre, example is the access control for the American nuclear arsenal. It is set up in such a way that, on one hand, no single individual has access authorization, and, on the other hand, individuals which should turn out to have "ethical concerns" cannot block the access. The cryptographic primitive that achieves this functionality is called *secret sharing*. It allows to share a *secret* among a set of players such that the players of any large enough subset can jointly reconstruct the secret (without the help of the remaining players) while the players of any smaller subset have no joint information about it. A stronger version of secret sharing, called *verifiable secret sharing* (VSS), incorporates robustness against malicious behavior of the participants. Even though the motivations, as given here, for (verifiable) secret sharing and for MPC are rather distinct, it turned out that the solutions are heavily related, in that (verifiable) secret sharing is a major building block for the construction of MPC protocols, and, on the other hand, VSS may in fact be viewed as a special case of MPC.

Nowadays, multi-player protocols play an important role in practical and theoretical cryptography, and not only because of the above mentioned applications it is probably fair to say that among them (verifiable) secret sharing and MPC belong to the most appealing and/or useful ones, and indeed an immense range of literature can be found treating various issues.

We proceed by giving more precise descriptions as well as some background information on secret sharing, VSS and MPC. We also point to some open problems that we address in this work. In Section 1.4, we summarize our contributions.

1.1 Secret Sharing

The concept of *secret sharing* was introduced in 1979 independently by Shamir [Sha79] and Blakley [Bla79] as a means to protect a secret simultaneously from exposure and from being lost. It allows a so called *dealer* to share the secret among a set of entities, called *players*, in such a way that certain specified subsets of the players are able to to reconstruct the secret without the help of the remaining players (*correctness*), while smaller subsets have no information about it (*privacy*).

The solution presented by Shamir is very elegant: To share a secret s among n players, where s is an element of a finite field K of cardinality $|K| > n$, the dealer chooses a random polynomial $f(X) = a_0 + a_1 X + \cdots + a_t X^t \in K[X]$ with constant coefficient $a_0 = s$ and degree at most t, where t is a public parameter in the range $0 \leq t \leq n-1$. Then, he hands privately to each player P_i the *share* $s_i = f(\omega_i) \in K$, where $\omega_1, \ldots, \omega_n$ are publicly-known, pairwise-distinct, non-zero elements of K. It now follows from Lagrange polynomial interpolation that $t+1$ (or more) players can reconstruct the polynomial $f(X)$ and hence the secret s, while up to t of the players have no information about s, in the sense that the collection of their shares is statistically independent of s.

The *access structure* of Shamir's secret sharing scheme, meaning the subgroups of players than can reconstruct the secret, is characterized by the *threshold* t. Secret sharing schemes for general (non-threshold) access structures were first proposed in [ISN87, BL88, Bri89].

Shamir's secret sharing scheme as well as most other schemes proposed in the literature belong to a particularly nice class of secret sharing schemes, where every share s_i is computed as a fixed linear combination of the secret s and randomly chosen values $a_1, \ldots, a_t$ (the coefficients being $1, \omega_i, \ldots, \omega_i^t$ in case of Shamir's scheme). This is obviously sufficient (and, as is well known, also necessary) for the scheme to be *linear* (or *homomorphic*) in the sense that if $s_1, \ldots, s_n$ and $s'_1, \ldots, s'_n$ are sharings of two secrets $s, s' \in K$, and if $\lambda \in K$, then $s_1 + s'_1, \ldots, s_n + s'_n$ is a sharing of $s + s'$ and $\lambda s_1, \ldots, \lambda s_n$ is a sharing of λs. This property is very useful in particular in multi-party computation (see Section 1.3), as it immediately allows to compute (linear functions) on shared values. On the other hand, it has been shown in [BI01] that non-linear schemes allow more efficient solutions for certain access structures. Nevertheless, in this work we only consider linear schemes.

Most linear secret sharing schemes proposed in the literature (see e.g. [Bla79, Sha79, Bri89, BI93, BL88, KW93, Gál95, Bei96, CDM00]) require and rely on the fact that the secret is sampled from a finite *field* K. Shamir's scheme for instance heavily relies on Lagrange interpolation which does not hold in general over, say, a ring. On the other hand, as pointed out by Desmedt and Frankel in the context of threshold cryptography [DF89], it is desirable to have secret sharing schemes that allow to share secrets that are sampled from a finite Abelian *group* G.[1] In particular, it is desirable to have a *black-box* secret sharing scheme whose security is regardless of the group G from which the secret is chosen,

[1] The obvious approach of sharing a group element by identifying its bit representation with a field element of some large enough field (of characteristic 2) is unsatisfactory, both from a theoretical and a practical point of view, as it depends on the encoding of the secret and the resulting scheme fails to be linear (with respect to the group operation).

and which requires only *black-box* access to the group operations and to random group elements, i.e., where the computation of the shares (and the reconstruction of the secret) is by taking $\mathbb{Z}$-linear combinations of the secret and of random group elements (respectively of the shares).

Desmedt and Frankel proposed in 1994 a black-box threshold secret sharing scheme [DF94]. Their solution is related to Shamir's secret sharing scheme; however, in order to circumvent the problem that polynomial interpolation over the integers does not work as over fields, they pass to an integral extension ring of $\mathbb{Z}$ where polynomial interpolation works "good enough". More precisely, they pass to an integral extension ring of $\mathbb{Z}$ which contains a list of $n+1$ elements $\omega_0, \ldots, \omega_n$ such that all pairwise differences $\omega_i - \omega_j$ $(i \neq j)$ are invertible. The maximal number of such elements is called the *Lenstra constant* of the ring. It can be shown that polynomial interpolation works if $\omega_0, \ldots, \omega_n$ are taken as interpolation points. The scheme is then constructed using the fact that sufficiently many copies of G can be seen as a module over the extension ring.

The concrete choice of the extension ring made in [DF94] is the ring of p-th cyclotomic integers. For any prime p, the Lenstra constant for the ring of p-th cyclotomic integers is p. Hence, a black-box secret sharing scheme for threshold parameters t and n is constructed by choosing p as the smallest prime greater than n. The *expansion factor* of that scheme, i.e., (average) number of group elements per player as part of his share, lies between n and $2n$.

This is the best previous general approach to the problem. Further progress on the black-box secret sharing problem via the approach of [DF94] depends on the problem of finding for each n an integral extension of $\mathbb{Z}$ whose degree is *substantially* smaller than n and whose Lenstra constant is greater than n. To the best of our knowledge, this is an open problem of algebraic number theory (see also [DF94] and the references therein).

And indeed, except for some quite special cases, namely when t is constant or when t (respectively $n - t$) is small compared to n [DDB94, BBDW96], or the constant factor gain from [DKKK98], no substantial improvement on the general black-box secret sharing problem has been reported since.

1.2 Verifiable Secret Sharing

Secret sharing "only" guarantees correctness against players that "refuse to hand in their shares". Its stronger version *verifiable secret sharing*, VSS for short, introduced in [CGMA85], is secure against *dishonest* players and a (possibly) dishonest dealer that are *corrupted* by a central malicious entity, called the *adversary*. A corrupted player hands all information to the adversary and, following the adversary's instructions, may deviate arbitrarily from the protocol that he is supposed to run. The number of corrupted players is typically restricted by a threshold t. More general, the corrupted players must form a set that is contained in a prescribed list of possible corrupted player sets, the so called *adversary structure* [HM97]. Security of a VSS scheme requires the following.

- *Privacy:* If the dealer remains uncorrupted, then the adversary gains no information about the shared secret s as a result of the sharing procedure.
- *Correctness:* After the secret is shared, there exists a unique value s' that can be reconstructed by the players (no matter how the corrupted players behave), and, in case the dealer remains uncorrupted during the sharing procedure, s' is equal to the shared secret s.

Depending on the computational power of the adversary and whether one tolerates a small failure probability or not, there are different flavors of the security of VSS:

- *Computational security:* The adversary is restricted to be a probabilistic *poly-time* Turing machine and some computational problem is assumed to be hard.
- *Unconditional security:* The adversary is computationally unbounded, but one tolerates a negligibly small failure probability.
- *Perfect security:* The adversary is computationally unbounded, and no non-zero failure probability is tolerated.

Furthermore, there are different variations of VSS. Relevant for this work are the following two. In the first, the dealer is guaranteed to remain uncorrupted and to share his secret as prescribed. This we call *honest-dealer* VSS (and is also known as *robust* secret sharing). In the second variation, the *efficient* reconstruction of the secret requires the cooperation of the (possibly corrupted) dealer. As a consequence, by refusing to take part, the dealer can prevent the (efficient) reconstruction of the secret (but he cannot achieve that a different secret is reconstructed). This is called a *distributed commitment*, or DC for short, as it has the same functionality of a cryptographic commitment.

While threshold secret sharing schemes can (and do) exist for arbitrary threshold parameters t and n (with $0 \leq t \leq n-1$), the privacy and correctness conditions of a VSS obviously contradict each other if the adversary can corrupt $t \geq n/2$ players: privacy requires that t shares (the shares of the corrupted players) give no information about the secret, while correctness requires in particular that $n-t$ shares (the shares of the uncorrupted players) allow to recover the secret. Furthermore, if t is in the range $n/3 \leq t < n/2$ then the existence of a broadcast channel has to be assumed, and still a non-zero failure probability is unavoidable, as has been stated informally in [CCD88] (and will also be shown in this work). On the other hand these bounds are tight in that there exist VSS schemes for $t < n/2$ (e.g. [Fel87, Ped91] for computational and [RB89, CDD+99] for unconditional security), respectively for $t < n/3$ in case of perfect security (e.g. [BGW88]). The same bounds also apply to honest-dealer VSS and DC's.

Similarly to the case of (ordinary) secret sharing schemes, one important measure for the efficiency of a VSS scheme is the (average) size of the information held by the players as their share, and related to that is the communication complexity of the reconstruction. For computationally secure VSS, or for a threshold $t < n/3$, standard methods immediately give optimal solutions, where the size of a share equals the size of the secret (plus the security parameter in case of computational security). Somewhat surprisingly, little work

seems to have been done on the case of $n/3 \leq t < n/2$ with unconditional security. The first scheme that achieves this level of security was proposed in 1989 by Rabin and Ben-Or [RB89]. In this scheme, the size of a share is proportional to n^2 and the security parameter k, and as a consequence the reconstruction complexity is (kn^3). Ten years later, a scheme was proposed in [CDD+99] which considerably improved the communication complexity of the sharing procedure; however, it failed to improve the size of the shares or the reconstruction complexity. To the best of our knowledge, no other improvements have been reported since.

1.3 Multi-Party Computation

The goal of *multi-party computation*, MPC for short, as introduced in 1982 by Yao [Yao82], is to enable a set of players to evaluate an arbitrary function F on their private inputs. Informally, the computation must guarantee the correctness of the result while preserving the privacy of the players' inputs, even if some of the players are corrupted by an adversary and misbehave in an arbitrary way. A formal definition of secure MPC will be given in Section 3.6. As in the VSS setting, the set of corrupted players is restricted in that its size is upper bounded by a threshold t, or, more generally, in that it is contained in a given adversary structure, and security can be computational, unconditional or perfect. Furthermore, the same tight bounds on t hold as for secure VSS.

Since the initial plausibility results in this area [Yao86, GMW87, BGW88, CCD88, RB89], much effort has been put into enhancing these results, and nowadays there is a wide range of literature treating issues like improving the communication complexity (e.g., [FY92, GRR98, CDN01, HM01]) or the round complexity (e.g., [BB89, BMR90, Bea00, IK00]), and coping with more powerful (e.g., [OY91, CFGN96, DN03]) or more general (e.g., [HM97, FHM98, CDM00]) adversaries. For an up-to-date overview over unconditionally and perfectly secure MPC we refer to [Hir01].

A common restriction on these results is that the function F is always assumed to be represented by an arithmetic circuit (or formula or branching program) over a finite *field*, and hence all computations "take place" in this field. Thus, it is natural to ask whether MPC can also be efficiently implemented over a richer class of structures, such as arbitrary finite *rings*. This question makes sense from a theoretical point as well as from a practical point of view. Unfortunately, general rings do not enjoy some of the useful properties of fields on which standard MPC protocols rely: non-zero ring elements may not have inverses (in fact, a ring may even not contain 1, in which case *no* element is invertible), there might exist zero-divisors, and the multiplication may not be commutative.

The known MPC protocols that are not restricted to fields [BW98, Mau03] are based on "additive secret sharing with replication" and either do not work or are not efficient for the important case of a threshold adversary structure. Also the obvious approach of simulating each ring operation by an arithmetic or boolean circuit computing it is unsatisfactory, both from a theoretical point of view (as its complexity grows super-linearly in the length of a

ring element) and from a practical point of view.[2]

A step in a satisfactory direction was taken by the study of secret sharing schemes for arbitrary Abelian groups (see Section 1.1). However, this step falls short of providing a complete solution to our problem, and so the question of MPC over rings remains unanswered, even for "nice" rings (e.g. commutative and with 1).

Another open question is that of the efficiency of unconditionally secure MPC for a threshold t in the range $n/3 \leq t < n/2$ (with broadcast). There exist rather efficient protocols for computationally secure MPC [CDN01] as well as for $t < n/3$ with perfect or unconditional security [HMP00, HM01]. The currently most efficient protocol covering unconditional security and a threshold $n/3 \leq t < n/2$ is the protocol from [CDD+99], whose communication complexity is proportional to n^4 per multiplication gate in (the circuit representation of) F.[3] Even if pre-processing is allowed, the communication complexity of the actual computation is still proportional to n^3 per multiplication gate.

1.4 Contributions of this Thesis

In this work, which is based on [CDF01, CF02, FM02, ACF02, CFIK03], we address and contribute to three aspects of the theory of secure multi-player protocols:

FUNDAMENTALS GENERALITY EFFICIENCY

The first, fundamentals, is concerned with the quest for understanding *why* a protocol works and is secure. What are the fundamental principles behind a protocol and its security? Due to the complex environment in which multi-player protocols take part, it is important to identify underlying principles and express or characterize them in (hopefully simple) mathematical terms, such that they can be rigorously analyzed. By generality we especially mean generality with respect to the underlying mathematical structure. This can also be viewed as the quest for analyzed the mathematical *axioms* required to do some task. This is in particular motivated by the fact that essentially all multi-player protocols "take part" over a *field*. And efficiency of course deals with the improvement of protocols with respect to various kinds of complexity measures.

In Chapter 3, we first attack the problem of (black-box) secret sharing for finite Abelian groups. We generalize the notion of span programs over a finite field, as introduced by Karchmer and Wigderson in [KW93], to span programs over a (possibly infinite) commutative ring Λ with 1. This has to be done in a careful way, as a straightforward generalization would not result in a convenient notion. We show how this notion gives rise in a natural way to secret sharing schemes for finite Abelian groups. In particular we completely characterize *black-box* secret sharing for finite Abelian groups in terms of *integer* span

[2] Due to the lack of provable lower bounds in complexity-theory, one cannot tell for sure whether this is an inherent phenomenon or just a limitation of currently known techniques.

[3] The communication complexity as stated in [CDD+99] is proportional to n^5. However, as we pointed out (see e.g. [Dzi00]), one factor n can be saved.

programs. This is the first characterization of *black-box* secret sharing in terms of simple linear-algebra conditions on the integer sharing coefficients. This can be seen as an analogue to the well known equivalence between linear secret sharing and span programs over fields.

Then, we consider the problem of doing MPC over rings instead of over fields. We propose a framework that allows to do MPC over an *arbitrary* finite ring. In particular, we present an MPC protocol that requires only *black-box* access to the ring operations and to random ring elements. It is perfectly secure with respect to an adversary corrupting up to $t < n/3$ players, and its complexity essentially coincides with the classical field-based solutions. This is not only the first efficient MPC protocol for rings, but it is also the first efficient MPC protocols that is "universal" in the sense that it can be applied to any finite ring (and in particular any finite field). Maybe somewhat surprisingly, our framework is based on the same building blocks as standard (field-based) MPC protocols. As a consequence, it is compatible with various improvements and generalizations that apply to the standard field-based protocols, such that, loosely speaking, essentially everything that can be done over a field in the context of MPC can also be done over an arbitrary ring. We stress that we pose no restriction on the ring besides being finite.

We conclude Chapter 3 with an example that shows the potential usefulness of the possibility of embedding useful computational tasks into the richer structure of rings. Specifically, we show how to efficiently compute the maximum of n integers with better round complexity than using alternative approaches.

In Chapter 4, we show that for arbitrary threshold parameters t and n there exists a black-box secret sharing scheme with expansion factor $O(\log n)$. Recall that all previous schemes (for a general t) had an expansion factor of order n. Using a result of Karchmer and Wigderson [KW93], we show that the expansion factor of our scheme is minimal up to a small *additive* constant.

As an application of our black-box secret sharing scheme, and in order to show its usefulness in the context of threshold cryptography, we propose an RSA threshold signature scheme. In contrast to Shoup's famous scheme [Sho00], our solution has a considerably more expensive signature generation phase, however it allows a small public exponent like $e = 3$ and is therefore superior in a setting where the complexity of signature verification has to be minimized.

The goal of Chapter 5 is a unified treatment of perfectly secure (linear) VSS and distributed commitment schemes. We present a very natural and general sharing protocol which converts an arbitrary given linear secret sharing scheme into a DC (or VSS) scheme, provided of course that this is possible at all, by enforcing *pairwise* consistency among the shares of the (honest) players, i.e., by enforcing that pairwise linear dependences among the shares that should hold *do* hold. This is done by private pairwise checking and public clarification in case of inconsistencies, as in the classical BGW VSS scheme [BGW88]. This seems to be not only a very natural but the only possible approach for the construction of perfectly secure DC and VSS schemes, and indeed, all known schemes can be seen as concrete instances of this general approach.

We then provide an understanding of the security of the scheme in terms of pure linear algebra. As a consequence, the security of all known schemes (and possibly even all future ones) follow as corollaries whose proofs are linear-algebra arguments, in contrast to some hybrid arguments used in the literature. This is demonstrated for two schemes, for the CDM DC scheme of [CDM00] and for the classical BGW VSS scheme of [BGW88].

Our approach, establishing the minimal conditions for security, may very well lead to the design of ad-hoc VSS or DC schemes for general adversary structures which are more efficient than the schemes resulting from generic constructions (e.g. [CDM00]).

In Chapter 6 we investigate the share size and related to that the reconstruction complexity of threshold VSS. The only interesting case is $n/3 \leq t < n/2$ with unconditional security. We first consider the weaker version of VSS where the dealer is assumed to remain honest, and we show a lower bound on the reconstruction complexity of $\Omega(kn^2/(n-2t))$ bits, under the assumption that the protocol is designed in such a way that (1) the reconstruction is completed in one round, and (2) that every player either reconstructs the correct secret or outputs `failure`, where the latter happens with probability at most $2^{-\Omega(k)}$ (and is unavoidable). This may be seen as an answer to the question "what does it cost to get the best possible security (with respect to an honest dealer) in a minimal number of rounds?". This lower bound meets the upper bound resulting from the honest-dealer VSS scheme from [RB89] for *and only for* maximal t, i.e., $t = \lfloor (n-1)/2 \rfloor$. We propose a protocol whose complexity meets the bound *for all* t in the considered range.

Next we incorporate security against a possibly dishonest dealer. The honest-dealer VSS establishing the upper bound can—at least in principle—be turned into a VSS by replacing the dealer by a multi-party computation using generic methods (e.g. [RB89, CDD+99]). The resulting scheme would improve on the reconstruction complexity of all known schemes; however, the sharing phase would become extremely inefficient in comparison. We close this gap and present a new VSS scheme where the complexity of the sharing phase matches that of the previously best known VSS for our scenario [CDD+99], but where the reconstruction meets our lower bound (for honest-dealer VSS). This beats previous VSS schemes by a factor n for maximal t and by a factor n^2 for $t \leq q \cdot n$ with $1/3 < q < 1/2$.

The idea of *pre-processing* in MPC, introduced by Beaver in [Bea91], is to do as much work as possible in a *pre-processing phase*, before the inputs to the function F to be computed (and possibly even F itself) are known. This is to reduce the work in the *computation phase* when the inputs (and the function) have become available. More specifically, the work of sharing the inputs and securely multiplying shared values is done in the pre-processing phase by doing it on random values, while the computation phase degenerates to reconstructing the differences of the random values used in the pre-processing and the actual values. It is therefore not surprising that MPC in the pre-processing model benefits from our VSS scheme with optimized reconstruction. Indeed, assuming broadcast, we propose an unconditionally secure MPC protocol with pre-processing for a threshold $n/3 \leq t < n/2$ with a communication complexity of the computation phase $O(kn^2/(n-2t))$ per multiplication gate in (the circuit representation of) F. This improves the complexity of the computation phase of earlier similar protocols by a factor of at least n without

increasing the asymptotic complexity of the pre-processing.

In the final Chapter 7. We consider the well know Pedersen threshold VSS scheme [Ped91]. It is based on Pedersen's homomorphic commitment scheme and is computationally secure for $t < n/2$. In [Abe99], Abe used Pedersen's VSS scheme in an elegant way to construct a *commitment multiplication proof*, which allows to prove to a group of n players, *verifiers*, a multiplicative relation among three committed values without revealing anything beyond, even if up to t of the verifiers are corrupted. This is an important building block for computationally secure MPC protocols, but was also used in [Abe99] to construct a provably secure threshold encryption scheme. In comparison with the standard approach of adopting a single verifier interactive zero-knowledge proof of a multiplicative relation among committed values [CD98] to the distributed-verifier setting by jointly generating the random challenge, the advantage of Abe's solution is on one hand its simpleness, it only consist of a few (parallel) executions of the Pedersen VSS sharing phase, and that it is *non-interactive* (in some sense).

While investigating Abe's protocol, we make two observations: (1) a building block used in its construction already constitutes a secure commitment multiplication proof, and (2) it inherits an extra property that has previously been overlooked. The former not only leads to a simplified exposition but also saves on the required number of invocations of Pedersen's VSS, while the latter allows to construct non-interactive proofs of *partial knowledge* in the considered distributed-verifier setting, as we show. This can be used for instance to prove the knowledge of ℓ out of m given secrets, without revealing which ones. We also show how to construct efficient non-interactive zero-knowledge proofs for circuit satisfiability in the distributed-verifier setting.

Chapter 2

Preliminaries

2.1 Access Structures and Adversary Structures

Let n be a positive integer, and consider the set $\{1, \ldots, n\}$.

Definition 2.1 *A non-empty collection Γ of subsets $A \subseteq \{1, \ldots, n\}$ is called an* access structure *on $\{1, \ldots, n\}$ if $\emptyset \notin \Gamma$ and if Γ is closed under taking supersets: for all $A \in \Gamma$ and all $A' \subseteq \{1, \ldots, n\}$ with $A \subseteq A'$ it holds that $A' \in \Gamma$.*

A collection $\mathcal{A}$ of subsets $A \subseteq \{1, \ldots, n\}$ is called an adversary structure *on $\{1, \ldots, n\}$ if $\emptyset \in \Gamma$ and if $\mathcal{A}$ is closed under taking subsets: for all $A \in \mathcal{A}$ and all $A' \subseteq \{1, \ldots, n\}$ with $A' \subseteq A$ it holds that $A' \in \mathcal{A}$.*

Remark 2.1 There is a natural one-to-one correspondence between all access structures and all adversary structures on $\{1, \ldots, n\}$: the complement $\bar{\Gamma} = \{A \subseteq \{1, \ldots, n\} \mid A \notin \Gamma\}$ of an access structure Γ is an adversary structure, and vice versa the complement $\bar{\mathcal{A}}$ of an adversary structure $\mathcal{A}$ is an access structure, and $\bar{\bar{\Gamma}} = \Gamma$ respectively $\bar{\bar{\mathcal{A}}} = \mathcal{A}$. However, in the context of secret sharing it will be convenient to consider an access structure, the "collection of player sets that are allowed to reconstruct the secret", while in the context of VSS and MPC it will be more convenient to consider an adversary structure, the "collection of player sets that the adversary may corrupt".

Definition 2.2 *Let t be an integer in the range $0 \leq t \leq n-1$. The* threshold access structure *$\Gamma_{t,n}$ is the collection of all subsets $A \subseteq \{1, \ldots, n\}$ with $|A| > t$. The* threshold adversary structure *$\mathcal{A}_{t,n}$ is the collection of all subsets $A \subseteq \{1, \ldots, n\}$ with $|A| \leq t$, i.e., $\mathcal{A}_{t,n} = \bar{\Gamma}_{t,n}$.*

Remark 2.2 Some authors define $\Gamma_{t,n}$ as consisting of all sets of size *at least* t. Our definition adheres to a convention in the multi-party computation literature.

Definition 2.3 *An adversary structure $\mathcal{A}$ on $\{1, \ldots\}$ is called to be Q^2 if no two sets of $\mathcal{A}$ cover $\{1, \ldots, n\}$: $A \cup A' \neq \{1, \ldots, n\}$ for all $A, A' \in \mathcal{A}$. Similarly, $\mathcal{A}$ is called to be Q^3 if no three sets of $\mathcal{A}$ cover $\{1, \ldots, n\}$: $A \cup A' \cup A'' \neq \{1, \ldots, n\}$ for all $A, A', A'' \in \mathcal{A}$.*

Remark 2.3 The Q^2 (respectively Q^3) condition was introduced in [HM97] and generalizes the threshold condition $t < n/2$ (respectively $t < n/3$) for threshold adversary structures to arbitrary adversary structures.

2.2 Communication Model

A secret sharing scheme is described simply by a probabilistic poly-time sharing algorithm (polynomial in the parameter n and in a possible security parameter), which describes how to compute the shares given a secret, and a (typically deterministic) poly-time reconstruction algorithm, which describes how to reconstruct the secret given enough shares. For VSS (including its variations mentioned in Section 1.2) and MPC, the matter is more complex. VSS and MPC are described by *interactive protocols*, executed among a set of n entities, denoted by $P_1, \ldots, P_n$ and called *players*, (and a so called *dealer* $D = P_0$ in case of VSS) for which we have to specify their means of communication. This is done below.

We assume that every pair of entities is connected by a secure (meaning authentic and private) channel. For the protocols in Chapter 6 it is also necessary to assume the existence of a broadcast channel.[1] Formally, the players (and the dealer) are probabilistic poly-time interactive Turing machines. Each entity has a read-only input tape, a write-only output tape, and a read-write working tape. The secure communication from an entity P_i to another entity P_j is modeled by a communication tape that is write-only for P_i and read-only for P_j (while all other entities have neither read nor write access). A possible broadcast channel is modeled by another communication tape for each entity P_i, for which P_i has write-only and all other entities have read-only access.

Furthermore, we assume the entities to be fully *synchronous*, meaning that the protocol execution is divided up into globally clocked *rounds*. In each round, every entity reads all messages that have been sent to him (or broadcasted) in the previous round, i.e., he reads all communication tapes (for which he has read access), and, based on these messages and some local computations, he sends off new messages, i.e., writes new messages to all communication tapes (for which he has write access). Exceptions are the first and the last round. In the first round, every entity reads his input tape instead of the communication tapes, and in the last round, every entity writes to his output tape instead of to the communication tapes.

2.3 Adversary Model

As has been informally discussed in the introduction, security of a VSS or MPC protocol involves an additional entity: the *adversary*. The adversary can corrupt entities and take over their control, such that he can read their memory, and act on their behalf in an arbitrarily malicious way.

[1] All other protocols are in a setting where the broadcast channel can be implemented from the bilateral channels. Nevertheless, for simplicity, we often treat it as a given primitive.

Formally, the adversary is modeled by another probabilistic Turing machine. Whether the adversary is poly-time or not depends on the flavor of security considered. If he is, then we say the adversary is *computationally bounded*, and if not, then we say that he is *computationally unbounded*. Per se, the adversary has only (read and write) access to his own working tape and read access to the communication tapes that model the broadcast channel (if existing). This models the assumption that the channels are secure. However, the adversary may *corrupt* some entities. Corrupting an entity P_i means that the adversary gets to see P_i's internal state and inherits all of P_i's read and write accesses, and P_i's program is halted. Furthermore, modelling a system where participants are not trusted to effectively erase local data, the adversary learns *for any prior time* P_i's internal state and the content of all tapes to which P_i has read access. This information may for instance be stored on some memory tape associated with P_i. An entity that is (currently) uncorrupted is also called *honest*.

At some points we also consider a *passive* adversary. Upon corrupting P_i, such an adversary can monitor P_i's internal states, working tape and incoming messages, but P_i continues to correctly execute his program. We also say that the adversary *eavesdrops* P_i.

The adversary is restricted in the players that he can corrupt. Either by a threshold t in that the number of corrupted players must not be greater than t, or, more generally, by an adversary structure $\mathcal{A}$ on $\{1, \ldots, n\}$ in that the set $\{P_{i_1}, \ldots, P_{i_m}\}$ of corrupted players must correspond to a set in $\mathcal{A}$ in the sense that $\{i_1, \ldots, i_m\} \in \mathcal{A}$.[2] In the former case, we call the adversary a *threshold* adversary (and otherwise a *general* adversary). If $\mathcal{A}$ is Q^2 (respectively Q^3), then we call the adversary a Q^2 (respectively Q^3) adversary.

In case of VSS, where besides the players there is also a dealer, we take it as understood that the adversary may corrupt the dealer *in addition* to the players he is allowed to corrupt. However, if one of the players executes the dealer's part (besides his own), this e.g. holds in the context of MPC, corrupting the dealer is *included* in the player corruptions.

Furthermore, the adversary may be *static* or *adaptive*, the former meaning that he has to corrupt the players *before* the execution of the protocol and the latter that he can corrupt players at his will *during* the protocol execution, depending on what he has seen so far. We consider per default an adaptive adversary, except in Chapter 7 where we have to restrict to a static one.

Finally, we assume the adversary to be *rushing*. This means that he can read the messages (sent by the uncorrupted to the corrupted players) already in the *same* round as they have been sent, rather than in the next following round, and thus before having to decide on the messages for the corrupted players in this round.

2.4 Groups, Rings and Fields

We assume the reader to be familiar with basic concepts of group, ring and field theory. We fix some notation and terminology.

[2] We often fail to distinguish between a set $\{P_{i_1}, \ldots, P_{i_m}\}$ of players and the set $\{i_1, \ldots, i_m\}$ of indices.

In this work, we mainly consider groups that are *finite* and *Abelian*. We typically use *additive* notation and denote the neutral element by 0. The *exponent* of a finite group G is the smallest integer $m > 0$ such that $\underbrace{a + \cdots + a}_{m \text{ times}} = 0$ for all $a \in G$.

Let R be an arbitrary ring (not necessarily with 1). The set of *units* (i.e. invertible elements) of R is denoted by R^*. If R is finite then the *exponent* of R is the exponent of the additive group of R.

Consider now a (possibly infinite) *commutative* ring *with* 1 (which may be the trivial ring $\{0\}$ where $1 = 0$). Typically, we denote such a ring by Λ. For $\lambda_1, \ldots, \lambda_l \in \Lambda$, we write $(\lambda_1, \ldots, \lambda_l)_\Lambda$ for the ideal in Λ generated by $\lambda_1, \ldots, \lambda_l$. If Λ is clear from the context, we simply write $(\lambda_1, \ldots, \lambda_l)$. Let I be an ideal in Λ, and let $\Lambda \to \Lambda/I$, $\lambda \mapsto \overline{\lambda}$ be the canonical projection from Λ to the ring of residue classed modulo I. Then a *lift* of $h \in \Lambda/I$ is an element $\lambda \in \Lambda$ such that $\overline{\lambda} = h$. Finally, two ideals I and J in Λ are called *co-prime* if $I + J = \Lambda$, and two elements λ and μ of Λ are called *co-prime* if the ideals (λ) and (μ) are, i.e., if $(\lambda, \mu) = \Lambda$. Note that λ and μ are co-prime if and only if the residue class of λ modulo (the ideal generated by) μ is a unit: $\overline{\lambda} \in (\Lambda/(\mu))^*$.

Let q be a prime power. The unique field K with q elements is denoted by $\mathbb{F}_q$. Elements of a field K or of a commutative ring Λ with 1 are also called *numbers*.

2.5 Modules and Algebras

Let Λ be a commutative ring with 1.

Definition 2.4 *An (additive) Abelian group G is called a* module *over Λ, or simply a* Λ-module, *if a* number-multiplication $\Lambda \times G \to G$, $(\lambda, a) \mapsto \lambda \cdot a$, *is given such that*

$$\begin{aligned} 1 \cdot a &= a \\ \lambda \cdot (a + b) &= (\lambda \cdot a) + (\lambda \cdot b) \\ (\lambda + \mu) \cdot a &= (\lambda \cdot a) + (\mu \cdot a) \\ (\lambda \cdot \mu) \cdot a &= \lambda \cdot (\mu \cdot a) \end{aligned}$$

for all $\lambda, \mu \in \Lambda$ and $a, b \in G$.

Hence, loosely speaking, a module is a vector space over a ring (instead of over a field).

Remark 2.4 We have two kinds of addition (addition in Λ and in G) and two kinds of multiplication (multiplication in Λ, and in G). It should always be clear from the context, which addition or multiplication is meant.

Definition 2.5 *An arbitrary ring R is called an* algebra *over Λ, or simply a* Λ-algebra *if (the additive group of) R is a Λ-module and*

$$(\lambda \cdot a) \cdot b = \lambda \cdot (a \cdot b) = a \cdot (\lambda \cdot b)$$

holds for all $\lambda \in \Lambda$ and $a, b \in R$.

If G is a Λ-module (respectively R a Λ-algebra), then we also write λa or $a\lambda$ instead of $a \cdot \lambda$ for $\lambda \in \Lambda$ and $a \in G$ (respectively $a \in R$).

Example 2.1 Every Abelian group G is a $\mathbb{Z}$-module and every ring R is a $\mathbb{Z}$-algebra; the number-multiplication is given by $0 \cdot a = 0$, $\lambda \cdot a = a + \ldots + a$ (λ times) for $\lambda > 0$ and $\lambda \cdot a = -((-\lambda) \cdot a)$ for $\lambda < 0$.

Example 2.2 Let G be an Abelian group (respectively R a ring) with exponent m. Then G (respectively R) is a $\mathbb{Z}_m$-module (algebra). The number multiplication $\lambda \cdot a$ for $\lambda \in \mathbb{Z}_m$ is (well) defined by $\lambda \cdot a = \ell \cdot a = a + \ldots + a$ (ℓ times), where $0 < \ell \in \mathbb{Z}$ is a lift of λ.

Below, we give an informal definition of the tensor product of two modules or algebras. For a formal definition we refer to [Lan97].

Definition 2.6 (informal) *Let G and G' be two Λ-modules. The* tensor product *of G and G', denoted as $G \otimes G'$, is a Λ-module whose elements are finite sums of the form $\sum_i a_i \otimes a'_i$ with $a_i \in G$ and $a'_i \in G'$ such that*

$$\begin{aligned} a \otimes a' + a \otimes b' &= a \otimes (a' + b') \\ a \otimes a' + b \otimes a' &= (a + b) \otimes a' \\ \lambda \cdot (a \otimes a') &= (\lambda \cdot a) \otimes a' = a \otimes (\lambda \cdot a') \end{aligned}$$

for all $a, b \in G$, $a', b' \in G'$ and $\lambda \in \Lambda$.

For two Λ-algebras R and R', the tensor product *$R \otimes R'$ of R and R' is defined in a similar way, except that additionally*

$$(a \otimes a') \cdot (b \otimes b') = ab \otimes a'b'$$

is required to hold for all $a, b \in G$, $a', b' \in G'$.

2.6 Asymptotic Complexity Notation

We briefly introduce the standard asymptotic complexity notation which we use in this work. Consider two non-negative real-valued functions $f(k)$ and $g(k)$ over the positive integers $k \in \mathbb{N}$.

Definition 2.7 *We write $g(k) \leq O(f(k))$ if there exists $c > 0$ and $k_0 \in \mathbb{N}$ such that*

$$g(k) \leq cf(k) \quad \text{for all} \quad k \geq k_0 .$$

We write $g(k) \geq \Omega(f(k))$ if there exists $c > 0$ and $k_0 \in \mathbb{N}$ such that

$$\frac{1}{c} f(k) \leq g(k) \quad \text{for all} \quad k \geq k_0 .$$

And we write $g(k) = \Theta(f(k))$ if there exists $c > 0$ and $k_0 \in \mathbb{N}$ such that

$$\frac{1}{c} f(k) \leq g(k) \leq cf(k) \quad \text{for all} \quad k \geq k_0 .$$

This can be generalized to multi-variable functions $f(\mathbf{k})$ and $g(\mathbf{k})$ over $\mathbf{k} \in \mathbb{N} \times \cdots \times \mathbb{N}$ by defining a partial ordering "$\leq$" on $\mathbb{N} \times \cdots \times \mathbb{N}$ in the obvious way.

Definition 2.8 *We write $g(k) \leq negl(k)$ if for every $c > 0$ there exists $k_0 \in \mathbb{N}$ such that*

$$g(k) \;\leq\; \frac{1}{k^c} \quad \textit{for all} \quad k \geq k_0\,.$$

Convention. By slight abuse of notation, we also write $O(f(k))$ to represent some function $g(k)$ with $g(k) \leq O(f(k))$, and accordingly for $\Omega(f(k))$, $\Theta(f(k))$ and $negl(k)$.

Chapter 3

Secret Sharing over Groups and MPC over Rings

3.1 Introduction

> *There are two kinds of people, those who do the work and those who take the credit. Try to be in the first group; there is less competition there.*
>
> — Indira Gandhi

We are motivated by the following restriction on "standard" linear secret sharing schemes and multi-party computation protocols found in the literature. Most of the proposed schemes and protocols "take place" over a finite *field*, and rely on the nice mathematical structure that distinguishes a field from other mathematical objects like groups or rings. This means in the context of secret sharing that the secret has to be an element of a finite field, and in the context of MPC that the function F to be computed has to be represented as an arithmetic circuit over a finite field. Indeed, already over a relatively "harmless" ring like $\mathbb{Z}_m$, Shamir's secret sharing scheme for instance (which also serves as standard building block for MPC) is not secure a-priori. For instance, if m is even and if $f(X)$ is a polynomial over $\mathbb{Z}_m$ hiding a secret s as its free coefficient, then the share $s_2 = f(2)$ is odd if and only if the secret s is odd.

It is therefore natural to ask whether linear secret sharing and MPC can be based upon weaker algebraic objects like groups or rings. This question makes sense from a theoretical point of view, where it may be viewed as a quest for minimizing the *axioms* for linear secret sharing and MPC, but also from a practical point of view, as a positive answer would for instance in the case of MPC allow greater freedom in the representation of F, which in turn can lead to efficiency improvements. Note that it is obviously always possible to share the bit representation of an arbitrary secret by identifying it with a field element of some large enough field (of characteristic 2). However, this approach is unsatisfactory from a theoretical point of view, as it depends on the encoding of the secret, and from a practical point of view, as the resulting scheme is not linear. A similar objection applies to basing MPC on, say, rings by implementing the ring operations using field operations.

The idea of sharing a secret from a group rather than a field was considered by Desmedt and Frankel in the context of threshold cryptography [DF89]. The reason being that for

certain cryptosystems, like for instance RSA, the secret key does not come from a field but from a (typically Abelian) group. In particular, they pointed out the need for a *black-box* secret sharing scheme that allows to share a secret from an arbitrary Abelian group G, requiring only black-box access to the group operations and to random group elements. In such a scheme, the shares are computed as $\mathbb{Z}$-linear combinations of the secret and of random group elements, and, similarly, the secret is reconstructed by a $\mathbb{Z}$-linear combination of the shares, where the integer sharing and reconstruction coefficients are designed *independently* of the group G from which the secret is sampled, and privacy and correctness hold *regardless* of G.

Desmedt and Frankel then showed in [DF94] how to construct for every finite Abelian G with publicly known exponent a linear threshold secret sharing scheme that allows to share secrets from G. Furthermore, they proposed a (less efficient) black-box threshold secret sharing scheme for arbitrary Abelian groups. However, they did not address non-threshold access structures, neither did they give *necessary* conditions for (black-box) secret sharing, which seem to be important in proving lower bound results. Besides the work of [Kin00, Kin01], these issues remained unconsidered. And even though [Kin00] does give some characterization of black-box threshold secret sharing, this characterization is still in terms of secret sharing and thus still quantifies over all finite Abelian groups, rather than providing simple algebraic conditions on the integer sharing and reconstruction coefficients.

On the MPC front, to the best of our knowledge, the only result dealing with (unconditionally secure) MPC over non-fields is due to [Mau03], where a MPC protocol was proposed which works for non-field rings. However, in the important case of a threshold adversary structure $\mathcal{A}_{t,n}$, its complexity is exponential in t.

In this chapter, we propose a framework for (black-box) secret sharing for finite Abelian groups, as well as for unconditionally secure MPC over *arbitrary* finite rings. In particular, on one hand, we give a characterization of black-box secret sharing in terms of our notion of *integer span programs* (Section 3.4), which generalizes the well-known correspondence between span programs over fields [KW93] and linear secret sharing schemes over fields. On the other hand, we present an MPC protocol over an arbitrary finite ring which requires only *black-box* access to the ring operations (addition, subtraction and multiplication) and the ability to sample random ring elements (Section 3.5). It is perfectly secure with respect to an adversary corrupting up to $t < n/3$ players, and its complexity is comparable to the classical field-based solutions. This is a two-fold improvement over the classical field-based MPC results. It shows that MPC can be efficiently implemented over a much richer class of structures, namely arbitrary finite rings, and it shows that there exists in fact one "universal" protocol that works for any finite ring. We stress that besides being finite, we pose no restriction on the ring: there may exist zero-divisors, the ring may not contain 1 (and thus no units at all), and the multiplication may not be commutative. Furthermore, the framework is compatible with various improvements and generalizations that apply to the classical field-based protocols, except that it is not compatible with the standard methods to achieve *constant-round* MPC.[1] This gap is closed in [CFIK03], where

[1] The reason being that these methods typically require a randomly chosen element to have a good chance of being invertible, which does not hold for general rings, which might contain no units at all.

a constant-round black-box MPC protocol is proposed.

We conclude the chapter with an example for the potential usefulness of secure MPC over non-field rings. Specifically, we show how to efficiently compute the maximum of n integers with better round complexity than using alternative approaches.

The material in this chapter is based on [CF02, CFIK03].

3.2 Secret Sharing

Throughout this section, let Λ be a (not necessarily finite) commutative ring with 1, and let n be a positive integer.

Definition 3.1 *Let M be a matrix over Λ with, say, d rows, and let $\psi : \{1,\ldots,d\} \to \{1,\ldots,n\}$ be a surjective function. The quadruple $\mathcal{M} = (\Lambda, M, \psi, n)$ is called a* labeled matrix *(over Λ, on $\{1,\ldots,n\}$). Its* size *is defined as* $\mathrm{size}(\mathcal{M}) = d$*, the number of rows of M, and its* expansion factor *is defined as* $\mathrm{size}(\mathcal{M})/n$.

Definition 3.2 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix over Λ. We write $d(\mathcal{M})$ and $e(\mathcal{M})$ for the number of rows and columns, respectively, of M. For $A \subseteq \{1,\ldots,n\}$, we write $d_A(\mathcal{M})$ for the size of the set $\psi^{-1}(A) \subseteq \{1,\ldots,d(\mathcal{M})\}$.*

Convention. If the labeled matrix $\mathcal{M}$ is clear from the context, then we simply write d, d_A and e instead of $d(\mathcal{M})$, $d_A(\mathcal{M})$ and $e(\mathcal{M})$, respectively. Sometimes, we also write e or d even if $\mathcal{M}$ is not *yet* clear, but will become later on.

Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix. ψ is called *labeling function*, and we say that the j-th row $\mathbf{m}_j$ ($j = 1 \ldots d$) of M is *labeled* by $\psi(j)$. For $\emptyset \neq A \subseteq \{1,\ldots,n\}$, $M_A \in \Lambda^{d_A \times e}$ denotes the restriction of M to the rows $\mathbf{m}_j$ with $\psi(j) \in A$. Similarly, for an arbitrary d-vector $\mathbf{x} = (x_1,\ldots,x_d)$ (with entries not necessarily in Λ), $\mathbf{x}_A$ denotes the restriction of $\mathbf{x}$ to the coordinates x_j with $\psi(j) \in A$. If $A = \{i\}$ for some $i \in \{1,\ldots,n\}$, we write M_i and $\mathbf{x}_i$ instead of M_A and $\mathbf{x}_A$, respectively.

Definition 3.3 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix and Γ an access structure on $\{1,\ldots,n\}$. $\mathcal{M}$ is called a* linear secret sharing scheme *(over Λ) for the access structure Γ, respectively for the adversary structure $\mathcal{A} = \bar{\Gamma}$, if the following holds. Let G be an arbitrary finite Λ-module, and let $A \subseteq \{1,\ldots,n\}$ be a non-empty set. For arbitrarily distributed $s \in G$, let $\mathbf{b} = (b_1,\ldots,b_e)^T \in G^e$ be drawn uniformly at random, subject to $b_1 = s$, and define $\mathbf{s} = M\mathbf{b}$.*

- *Correctness: If $A \in \Gamma$, then there exists a* reconstruction vector $\boldsymbol{\lambda} \in \Lambda^{d_A}$ *(for A), not depending on G, such that $\boldsymbol{\lambda}^T \cdot \mathbf{s} = s$.*
- *Privacy: If $A \notin \Gamma$, then $\mathbf{s}_A$ contains no Shannon information on s.*

We introduce some terminology: For s, $\mathbf{b}$ and $\mathbf{s}$ as in the above definition, we call s the *secret*, $\mathbf{s}$ a *sharing* and $\mathbf{s}_1, \ldots, \mathbf{s}_n$ *shares* (of s), and $\mathbf{b}$ the *sharing vector*. In a setting where shares are not guaranteed to be correctly computed, we call $\mathbf{s}$ and $\mathbf{s}_1, \ldots, \mathbf{s}_n$ as above a *correct* sharing, respectively *correct* shares. Finally, the entries of a reconstruction vector are called *reconstruction coefficients*.

Note that these schemes are indeed linear in the sense that the sum $\mathbf{s} + \mathbf{s}'$ of two sharings $\mathbf{s}$ and $\mathbf{s}'$, is a sharing for the sum $s + s'$ of their corresponding secrets s and s'. We also note that only requiring reconstruction to be linear, as some authors do, results in an equivalent definition. This is an easily proved consequence of Lemma 4.1.

Since every Abelian group can be seen as a $\mathbb{Z}$-module, a secret sharing scheme over the integers allows to share a secret s from an arbitrary finite Abelian group G, requiring only black-box access to the group operations and to random group elements, and where correctness and privacy hold regardless of G. Note that the number-multiplication can be done by standard double-and-add.

Definition 3.4 *A linear secret sharing scheme $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ over the integers is called a* black-box secret sharing scheme.

While linearity is an additional property that a general secret sharing scheme may have (and the schemes considered here *do* have), it is an inherent property of a secret sharing scheme that allows to share a secret from an arbitrary Abelian group, having only black-box access the the group operations and to random elements: there is no other possibility to compute the shares as by taking $\mathbb{Z}$-linear combinations.

Example 3.1 Let $n = 2$. The labeled matrix $\mathcal{M} = (\mathbb{Z}, M, \psi, 2)$, given by the matrix

$$M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{matrix} \to & 1 \\ \to & 2 \end{matrix}$$

with the indicated labeling ψ, is a black-box secret sharing scheme for the threshold access structure $\Gamma_{1,2}$. Indeed, for any finite Abelian group G, the shares of a secret $s \in G$ are $s_1 = s + g_2$ and $s_2 = g_2$ for a random $g_2 \in G$, and thus both are independent of s while on the other hand $s_1 - s_2 = s$. The expansion factor is 1.

Example 3.2 Let $n = 3$. The labeled matrix $\mathcal{M} = (\mathbb{Z}, M, \psi, 3)$, given by the matrix

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} \to & 1 \\ \to & 1 \\ \to & 2 \\ \to & 2 \\ \to & 3 \end{matrix}$$

with the indicated labeling ψ, is a black-box secret sharing scheme for the threshold access structure $\Gamma_{1,3}$. The expansion factor is $5/3$.

Example 3.3 Let $p > n$ be a prime. Consider the ring $\Lambda = \mathbb{Z}_p$ (which is in fact the field $\mathbb{F}_p$) and let $\omega_1, \ldots, \omega_n \in \mathbb{Z}_p$ be pairwise distinct non-zero elements. For $0 \leq t \leq n-1$, define the labeled matrix $\mathcal{M} = (\mathbb{Z}_p, M, \psi, n)$ as follows. M consists of n rows, where the i-th row equals $(1, \omega_i, \ldots, \omega_i^t)$ and is labeled by i. It is not hard to verify that $\mathcal{M}$ is a linear secret sharing scheme over $\mathbb{Z}_p$ for $\Gamma_{t,n}$. When applied to $G = \mathbb{Z}_p$ (which trivially is a $\mathbb{Z}_p$-module), it coincides with Shamir's secret sharing scheme. The expansion factor is 1.

Note that a group G is a $\mathbb{Z}_p$-module if and only if its exponent is p. Threshold secret sharing schemes for groups with non-prime exponent can be constructed using Chinese Remainder Theorem, as in [DF94].

3.3 Span Programs over Rings

3.3.1 Definitions

In this section, we introduce the notion of *span programs over rings*. This is a certain variation of span programs over finite fields, introduced by Karchmer and Wigderson [KW93]. These are well-known to have a natural one-to-one correspondence with linear secret sharing schemes over *finite fields* (see e.g. [Gál95, Bei96]). Span programs over rings will turn out to have a similar correspondence with linear secret sharing schemes over rings, as introduced in the previous section. In particular, *integer* span programs (ISP's) will turn out to be equivalent to *black-box* secret sharing.

Let Λ be a commutative ring with 1.

Definition 3.5 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix over Λ. The (column) vector $(1, 0, \ldots, 0)^T \in \Lambda^{e(\mathcal{M})}$ is called the* target vector *for $\mathcal{M}$, and is denoted by $\boldsymbol{\varepsilon}(\mathcal{M})$.*

Convention. If the labeled matrix $\mathcal{M}$ is clear from the context, then we simply write $\boldsymbol{\varepsilon}$ instead of $\boldsymbol{\varepsilon}(\mathcal{M})$.

Definition 3.6 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix over Λ, and let $A \subseteq \{1, \ldots, n\}$ be non-empty. We say that*

- $\mathcal{M}$ accepts A *if* $\boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$ *(and otherwise* $\mathcal{M}$ rejects A*), and*
- $\mathcal{M}$ strongly rejects A *if there exists* $\boldsymbol{\kappa} \in \ker M_A$ *with* $\boldsymbol{\varepsilon}^T \boldsymbol{\kappa} = 1$.

Let in addition Γ be an access structure on $\{1, \ldots, n\}$. Then $\mathcal{M}$ is a span program *(over Λ) for the access structure Γ, respectively for the adversary structure $\mathcal{A} = \bar{\Gamma}$, if it accepts the sets $A \in \Gamma$ and rejects the sets $A \notin \Gamma$. In that case, we also say that $\mathcal{M}$* computes *Γ, respectively $\mathcal{A}$.*

Convention. If the linear secret sharing scheme $\mathcal{M} = (\Lambda, M, \psi, n)$ is clear from the context, and if $\mathcal{M}$ accepts A, then $\boldsymbol{\lambda}(A)$ stands for some arbitrary (but fixed) vector $\boldsymbol{\lambda} \in \Lambda^{d_A}$ with

$M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$. If $\mathcal{M}$ strongly rejects A, then $\boldsymbol{\kappa}(A)$ denotes some arbitrary (but fixed) vector $\boldsymbol{\kappa} \in \ker M_A$ with $\boldsymbol{\varepsilon}^T \boldsymbol{\kappa} = 1$.

If Λ is a *field*, then rejection and strong rejection are equivalent, and our definition is identical to the computational model of (monotone) span programs over fields [KW93]. Indeed, this model is characterized by the condition that $A \in \Gamma$ if and only if $\boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$. The equivalence follows from the remark below.

Remark 3.1 By basic linear algebra, in case Λ is a *field*, $\boldsymbol{\varepsilon} \notin \mathrm{im} M_A^T$ if and only if there exists $\boldsymbol{\kappa} \in \ker M_A$ with $\boldsymbol{\varepsilon}^T \boldsymbol{\kappa} \neq 0$, which holds if and only if there exists $\boldsymbol{\kappa} \in \ker M_A$ with $\boldsymbol{\varepsilon}^T \boldsymbol{\kappa} = 1$. In case Λ is *not a field*, the implications from the right to the left do not necessarily hold: Consider for example the integer matrix $M = (2\ 0)$ or $M = (1\ 2)$, respectively. As a consequence, as opposed to the case of fields, a labeled matrix $\mathcal{M} = (\Lambda, M, \psi, n)$ with $\boldsymbol{\varepsilon} \in \mathrm{im} M^T$ need not compute any access structure. The implications from the left to the right trivially hold regardless of Λ.

Remark 3.2 If $\mathcal{M}'$ is a span program for an access structure Γ, but with respect to a non-standard target vector $\boldsymbol{\varepsilon}' \neq (1, 0, \ldots, 0)^T \in \Lambda^{e(\mathcal{M}')}$, then it can be transformed into a span program $\mathcal{M}$ with $d(\mathcal{M}) = d(\mathcal{M}')$ and $e(\mathcal{M}) = e(\mathcal{M}')$, computing the same access structure Γ, but with respect to the standard target vector $\boldsymbol{\varepsilon}(\mathcal{M}) = (1, 0, \ldots, 0)^T \in \Lambda^{e(\mathcal{M})}$, *if* (and in general only if) the vector $\boldsymbol{\varepsilon}'$ can be extended to a basis of the Λ-module $\Lambda^{e(\mathcal{M}')}$.

Remark 3.3 Using (generally less inefficient) representations of access structures as monotone Boolean formulas and using induction in a similar style as in e.g. [BL88], it is straightforward to verify that for every access structure Γ and for every commutative ring Λ with 1, there is a span program over Λ that computes Γ.

Definition 3.7 *A span program $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ over the integers is called an* integer span program*, ISP for short.*

3.3.2 Span Programs and Linear Secret Sharing

Let Λ be a commutative ring with 1.

Proposition 3.1 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a span program for an access structure Γ on $\{1, \ldots, n\}$. Then, $\mathcal{M}$ is a linear secret sharing scheme over Λ for Γ.*

Remark 3.4 The reciprocal of the claim of Proposition 3.1 does not hold in (very) general, i.e., for certain rings Λ, a linear secret sharing scheme over Λ need not necessarily be a span program over Λ. A (pathological) example is $\Lambda = \mathbb{Q}$. It is easy to see that the only finite group that is a module over $\mathbb{Q}$ is the trivial group $G = \{0\}$. Therefore, *any* labeled matrix over $\mathbb{Q}$ is a linear secret sharing scheme over $\mathbb{Q}$ for *any* access structure, while of course a labeled matrix over $\mathbb{Q}$ is a span program for at most one (and in fact,

since $\mathbb{Q}$ is a field, exactly one) access structure. Unfortunately, we are not aware of any example ring Λ which allows a non-trivial group G as module and still separates linear secret sharing schemes from span programs, neither were we able to prove that the only separating examples are of the above type (see also Remark 3.6).

Remark 3.5 In Theorem 3.1 we will show that for the important case $\Lambda = \mathbb{Z}$, the reciprocal of the claim of Proposition 3.1 does hold. This is also true in case of a *finite* ring Λ, as can easily be verified by applying the linear secret sharing scheme to the (additive) group $G = \Lambda$.

The proof of Proposition 3.1 is along the lines of the corresponding claim in the case of span programs and linear secret sharing schemes over finite fields [KW93]:

Proof. Consider an arbitrary set $A \subseteq \{1, \ldots, n\}$ and an arbitrary finite Λ-module G. Define $\mathbf{s} = M\mathbf{b}$ for arbitrary $\mathbf{b} = (s, b_2, \ldots, b_e)^T \in G^e$. Suppose $A \in \Gamma$. It follows that

$$\mathbf{s}_A^T \boldsymbol{\lambda}(A) \;=\; (M_A \mathbf{b})^T \boldsymbol{\lambda}(A) \;=\; \mathbf{b}^T M_A^T \boldsymbol{\lambda}(A) \;=\; \mathbf{b}^T \boldsymbol{\varepsilon} \;=\; s\,.$$

Thus the correctness condition from Definition 3.3 is satisfied. Suppose now that $A \notin \Gamma$. For arbitrary $s' \in G$, define

$$\mathbf{b}' \;=\; \mathbf{b} + (s' - s) \cdot \boldsymbol{\kappa}(A)$$

and $\mathbf{s}' = M\mathbf{b}'$. The secret defined by $\mathbf{s}'$ equals s' (i.e. the first coordinate of $\mathbf{b}'$ is s'), while on the other hand $\mathbf{s}'_A = \mathbf{s}_A$. This implies perfect privacy: the assignment $\mathbf{b}' = \mathbf{b} + (s' - s) \cdot \boldsymbol{\kappa}(A)$ provides a bijection between the set of possible vectors of "random coins" (group elements) consistent with $\mathbf{s}_A$ and s, and the set of those consistent with $\mathbf{s}_A$ and s'. Therefore, the privacy condition from Definition 3.3 is also satisfied. □

Remark 3.6 This proof allows to read off the main potential gap between linear secret sharing schemes and span programs over rings (as indicated in Remark 3.4 above): The privacy condition of a linear secret sharing scheme requires that if $A \notin \Gamma$, then for every finite Abelian group G and all $s, s' \in G$ there exists a vector $\mathbf{k} \in G^e$ with $s' - s$ as first entry and such that $M_A\mathbf{k} = \mathbf{0}$. On the other hand, a span program promises the existence of $\boldsymbol{\kappa} \in \Lambda^e$ such that for every finite Abelian group G and all $s, s' \in G$ the vector $\mathbf{k} = (s' - s) \cdot \boldsymbol{\kappa}$ is as required. So the (potential) difference lies essentially in the order of the quantifiers.

3.3.3 A Geometric View of Secret Sharing and Span Programs

Sometimes it is convenient to have a more geometric though equivalent view of linear secret sharing and span programs as defined above.

Let Λ be a commutative ring with 1. For $\mathbf{x}, \mathbf{y} \in \Lambda^e$, $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the standard in-product of $\mathbf{x}$ and $\mathbf{y}$. By identifying $\mathbf{x} \in \Lambda^e$ with $F_{\mathbf{x}} : \Lambda^e \to \Lambda$ such that $F_{\mathbf{x}}(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$, there is a

one-to-one correspondence between the vectors in Λ^e and the linear functionals $\Lambda^e \to \Lambda$. If G is a Λ-module, then $\langle \mathbf{x}, \mathbf{y} \rangle \in G$ is well-defined for $\mathbf{x} \in G^e$ and $\mathbf{y} \in \Lambda^e$. Furthermore, similar to above, identifying $\mathbf{x} \in G^e$ with $F_{\mathbf{x}} : \Lambda^e \to G$ such that $F_{\mathbf{x}}(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ provides a one-to-one correspondence between the vectors in G^e and the linear functions $\Lambda^e \to G$, which we also call linear *functionals* to emphasize the similarity to the case where $\mathbf{x} \in \Lambda^e$.

It is obvious that we may change the definition of secret sharing (Definition 3.3) and span programs (Definition 3.6) as follows, without changing their meaning. In the definition of linear secret sharing, we may consider $\mathbf{s}$ not as being computed by applying the integer matrix M to the vector $\mathbf{b}$, but by applying the corresponding functional $F_{\mathbf{b}} : \Lambda^e \to G$ to the rows of M. And, using V_A as a short hand for $\mathrm{im} M_A^T$, the "space owned by A", we may replace in Definition 3.6 the definition of acceptance and the strong rejection by

- $\mathcal{M}$ accepts A if $\boldsymbol{\varepsilon} \in V_A$, i.e. if the target vector lies in the space owned by A, and
- $\mathcal{M}$ strongly rejects A if there exists $\boldsymbol{\kappa} \in \Lambda^e$ with $F_{\boldsymbol{\kappa}}(V_A) = \{0\}$ and $F_{\boldsymbol{\kappa}}(\boldsymbol{\varepsilon}) = 1$, i.e. if there exists a functional that maps the target vector to 1 and simultaneously the space owned by A to 0.

This view allows for instance a more intuitive and less technical proof of Proposition 3.1 above: For the correctness condition, note that by linearity $\mathbf{s}_A$ uniquely defines how $F_{\mathbf{b}}$ acts on V_A, and hence, if $\boldsymbol{\varepsilon} \in V_A$, uniquely defines the shared secret $s = F_{\mathbf{b}}(\boldsymbol{\varepsilon})$. For the privacy condition, note that the functional $F_{\boldsymbol{\kappa}}$ allows to change the secret arbitrarily without changing the shares belonging to A.

3.4 Black-Box Secret Sharing and ISP's

In this section, we give a full characterization of black-box secret sharing in terms if integer span programs. We note that [Kin00] also gives a characterization of black-box secret sharing. However, the conditions stated there are still in terms of the black-box secret sharing scheme and quantify over all finite Abelian groups, rather than by providing simple algebraic conditions on the matrix M as we do. Therefore, we feel that our approach based on integer span programs is perhaps more useful and insightful, especially since span programs over finite fields have since long been known to be equivalent to linear secret sharing schemes over finite fields.

We first introduce some terminology and basic facts concerning the theory of $\mathbb{Z}$-modules (see e.g. [Lan97]), and we prove a Lemma that puts some light on the strong rejection property of ISP's.

Consider the $\mathbb{Z}$-module $\mathbb{Z}^e$, and let V be an arbitrary submodule. Then there exists a basis for V. The size of such a basis is called the *rank*, denoted by $\mathrm{rank}(V)$. We say that V is *closed* if any basis of V can be completed to a basis of $\mathbb{Z}^e$. If this is the case, then any basis of V can in fact be completed to *any* module lying in-between V and $\mathbb{Z}^e$. The *closure* of V is the smallest closed submodule $W \subseteq \mathbb{Z}^e$ that contains V. It holds that if W is the closure of V then $cW \subset V$ for some non-zero integer c (and W is maximal with

that property). In particular, $\mathrm{rank}(W) = \mathrm{rank}(V)$. Finally, for an arbitrary vector $\mathbf{x} \in \mathbb{Z}^e$ and an arbitrary non-zero integer m, we write $\mathbf{x} \in V \pmod m$ if there exists $\mathbf{v} \in V$ such that $\mathbf{x} \equiv \mathbf{v} \pmod m$, and we write $\mathbf{x} \notin V \pmod m$ if not.

Lemma 3.1 *Let V be a submodule of $\mathbb{Z}^e$ and W its closure, and let $\boldsymbol{\varepsilon} = (1, 0, \ldots, 0)^T \in \mathbb{Z}^e$. Then the following statements are equivalent.*

1. *There exists $\boldsymbol{\kappa} \in \mathbb{Z}^e$ with $F_{\boldsymbol{\kappa}}(V) = \{0\}$ and $F_{\boldsymbol{\kappa}}(\boldsymbol{\varepsilon}) = 1$.*
2. *$\boldsymbol{\varepsilon} \notin W \pmod p$ for all primes p.*
3. *$\boldsymbol{\varepsilon} \notin W$, and $W + \mathbb{Z}\boldsymbol{\varepsilon}$ is closed.*

Proof. For the implication from 1. to 2., note that $F_{\boldsymbol{\kappa}}(V) = \{0\}$ implies that $F_{\boldsymbol{\kappa}}(W) = \{0\}$. The implication is now immediate by reducing modulo an arbitrary prime p.

For the implication from 2. to 3., let $\{\mathbf{b}_1, \ldots, \mathbf{b}_l\}$ be an arbitrary basis of W and let $\mathbf{b}_{l+1}$ complete this basis to a basis of the closure of $W + \mathbb{Z}\boldsymbol{\varepsilon}$. Write $(\varepsilon_1, \ldots, \varepsilon_{l+1})$ for the coordinates of $\boldsymbol{\varepsilon}$ with respect to this basis. By 2. it holds that $\varepsilon_{l+1} = \pm 1$. This implies that also $\{\mathbf{b}_1, \ldots, \mathbf{b}_l, \boldsymbol{\varepsilon}\}$ is a basis for the closure of $W + \mathbb{Z}\boldsymbol{\varepsilon}$, and hence that $W + \mathbb{Z}\boldsymbol{\varepsilon}$ is closed.

Finally, the implication from 3. to 1. is obvious, using that by 3., a basis for W together with $\boldsymbol{\varepsilon}$ can be completed to a basis of $\mathbb{Z}^e$. □

We are now ready to prove

Theorem 3.1 *Let $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ be a labeled matrix over $\mathbb{Z}$, and let Γ be an access structure on $\{1, \ldots, n\}$. Then $\mathcal{M}$ is a black-box secret sharing scheme for Γ if and only if it is an ISP for Γ.*

Proof. It remains to show the "only if" direction. Let $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ be a black-box secret sharing scheme for Γ according to Definition 3.4. Consider an arbitrary set $A \subset \{1, \ldots, n\}$. Suppose $A \in \Gamma$. For an arbitrary prime p, set $G = \mathbb{Z}_p$. By the correctness condition from Definition 3.3, it follows that $(1, 0, \ldots, 0) \equiv \boldsymbol{\lambda}(A)^T M_A I = \boldsymbol{\lambda}(A)^T M_A \pmod p$, where $I \in \mathbb{Z}_p^{e \times e}$ is the identity matrix. This holds for all primes p. Hence, $M_A^T \boldsymbol{\lambda}(A) = (1, 0, \ldots, 0)^T = \boldsymbol{\varepsilon}$. Therefore, $\mathcal{M}$ accepts the sets $A \in \Gamma$.
To conclude the proof we show that the privacy condition from Definition 3.3 implies that $\mathcal{M}$ strongly rejects the sets $A \in \Gamma$. Suppose $A \notin \Gamma$. Assume that $\mathcal{M}$ does *not* strongly reject A. We show that this contradicts the privacy condition. Write $V_A = \mathrm{im} M_A^T$ and W_A for the closure of V_A, and let $c > 0$ be such that $cW_A \subseteq V_A$. By Lemma 3.1, there exists a prime p such that $\boldsymbol{\varepsilon} \in W_A \pmod p$, i.e., such that $\boldsymbol{\varepsilon} + p\mathbf{k} \in W_A$ for some $\mathbf{k} \in \mathbb{Z}^e$. It follows that $c\boldsymbol{\varepsilon} + cp\mathbf{k} \in cW_A \subseteq V_A$, and hence that $c\boldsymbol{\varepsilon} \in V_A \pmod{p^{m+1}}$, where m is maximal such that p^m divides c. Finally, since c/p^m is invertible modulo p^m, it holds that $p^m \boldsymbol{\varepsilon} \in V_A \pmod{p^{m+1}}$. However, similar to the the proof of Proposition 3.1, this implies that when the black-box secret sharing scheme $\mathcal{M}$ is applied to the group $G = \mathbb{Z}_{p^{m+1}}$,

then the players in A can compute a p^m-multiple of the shared secret, which contradicts the privacy condition. □

For a span program $\mathcal{M}$ over a field it is known that one can assume without loss of generality that $e \leq d$. The following lemma shows that the same holds for ISP's.

Lemma 3.2 *Let $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ be an ISP for an access structure Γ on $\{1, \ldots, n\}$. Then there exists an ISP $\mathcal{M}' = (\mathbb{Z}, M', \psi, n)$ of equal size and computing the same access structure Γ such that the number of columns of M' is at most the number of rows.*

Proof. Using the basic theory of $\mathbb{Z}$-modules, the straightforward dimension argument in the field case can be closely followed. Write $V = V_{\{1,\ldots,n\}}$ and let $W \subseteq \mathbb{Z}^e$ be its closure. The rank e' of W, seen as a $\mathbb{Z}$-module, is at most d. Since $\{1, \ldots, n\} \in \Gamma$ and hence $\boldsymbol{\varepsilon} \in \mathcal{M}$, and since $\mathbb{Z}\boldsymbol{\varepsilon}$ is obviously closed, $\boldsymbol{\varepsilon}$ can be completed to a basis of W, which itself can be completed to a basis of $\mathbb{Z}^e$. It follows that there exists a $\mathbb{Z}$-linear map $\phi : \mathbb{Z}^e \to \mathbb{Z}^{e'}$ whose restriction to W is bijective and which maps $\boldsymbol{\varepsilon} = (1, 0, \ldots, 0)^T \in \mathbb{Z}^e$ to $\boldsymbol{\varepsilon}' = (1, 0, \ldots, 0)^T \in \mathbb{Z}^{e'}$. Consider the labeled matrix $\mathcal{M}' = (\mathbb{Z}, M', \psi, n)$, where the matrix M' results from M by replacing each row $\mathbf{m}$ by $\phi(\mathbf{m})$. We show that $\mathcal{M}'$ accepts and strongly rejects the same sets $A \subseteq \{1, \ldots, n\}$ than $\mathcal{M}$. Clearly, $\boldsymbol{\varepsilon} \in V_A$ implies that $\boldsymbol{\varepsilon}' \in V'_A$, where V'_A is defined as the span of the rows of M'_A. On the other hand, if $\mathcal{M}$ strongly rejects A, then via the equivalence between 1. and 3. in Lemma 3.1, and observing that since $\phi|_W$ is bijective it preserves closeness, it holds that $\mathcal{M}'$ strongly rejects A. □

3.5 Secure MPC over Rings

3.5.1 Model

We assume a communication network among n players $P_1, \ldots, P_n$ as described in Section 2.2. We consider a computationally unbounded adversary that is adaptive and rushing. The security of the protocols will require the adversary to be Q^3. Our goal is to obtain a protocol for securely computing a function given by an arithmetic circuit over an arbitrary finite ring R. Such a protocol is *black-box* if its description is independent of R and it only makes black-box calls to the ring operations (addition, subtraction and multiplication) and to random ring elements, and if its security holds regardless of the underlying ring R. Formal security definitions can be found in Section 3.6.

3.5.2 Multiplicative Span Programs

The multiplication property for a span program over a field has been introduced in [CDM00]. It essentially requires that the product of two shared secrets can be written as a linear combination of locally computable products of shares. However, in our setting (where, given the span program, it is not clear from what ring the secret and shares are

sampled) we define the multiplication property as a sole property of the span program matrix.

Let Λ be a commutative ring with 1.

Definition 3.8 *Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a span program. $\mathcal{M}$ is called* multiplicative *if there exists a block-diagonal matrix $D \in \Lambda^{d \times d}$ such that $M^T D M = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$, where block-diagonal is to be understood as follows. Let the rows and columns of D be labeled by ψ, then the non-zero entries of D can be collected in blocks $D_1, \ldots, D_n$ such that for every i the rows and columns in D_i are labeled by P_i.*
$\mathcal{M}$ is called strongly multiplicative *if, for every strongly rejected set A, there exists a matrix D as above with $D_i = 0$ (the zero-matrix) for $i \in A$.*

As in the case of span programs over fields (see [CDM00]), for every adversary structure $\mathcal{A}$ there exists a (strongly) multiplicative span program $\mathcal{M}$ over Λ for $\mathcal{A}$ if and only if $\mathcal{A}$ is Q^2 (Q^3). Furthermore, there exists an efficient procedure to transform any span program $\mathcal{M}$ over Λ for a Q^2 adversary structure $\mathcal{A}$ into a *multiplicative* span program $\mathcal{M}'$ (over Λ) for the same adversary structure $\mathcal{A}$, such that the size of $\mathcal{M}'$ is at most twice the size of $\mathcal{M}$. Unfortunately, a similar result concerning the *strong* multiplication property is not known to exist, not even in the field case.

Corresponding to the field case, the multiplication property allows to securely compute a sharing of the product of two shared secrets. Indeed, let R be a finite ring which can be seen as an algebra over Λ, and let $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$ be two sharings of two secrets s and s' of R. Then, the product ss' of the shared secrets can be written as

$$ss' = \mathbf{b}^T \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T \mathbf{b}' = \mathbf{b}^T M^T D M \mathbf{b}' = (M\mathbf{b})^T D M \mathbf{b}' = \mathbf{s}^T D \mathbf{s}' = \sum_{i=1}^{n} \mathbf{s}_i^T D_i \mathbf{s}_i'$$

i.e., by the special form of D, as a linear combination (in fact as the sum) of locally computable values. Hence the multiplication protocol from [GRR98] can be applied: To compute securely a sharing $\mathbf{s}'' = M\mathbf{b}''$ of the product ss', every player P_i shares $p_i = \mathbf{s}_i^T D_i \mathbf{s}_i'$, and then every player P_i adds up his shares of $p_1, \ldots, p_n$, resulting in P_i's share $\mathbf{s}_i''$ of ss'.

3.5.3 Secure MPC over Rings

Given a multiplicative span program $\mathcal{M}$ over Λ for a Q^2 adversary structure $\mathcal{A}$ (where the multiplication property can always be achieved according to a remark above), it follows that if R is a Λ-algebra, then any circuit over R can be computed securely with respect to a *passive* adversary that can (only) eavesdrop the players of an arbitrary set $A \in \mathcal{A}$.

Namely, every player shares his private input(s) using the linear secret sharing scheme $\mathcal{M}$, and then the circuit is securely evaluated gate by gate, the addition gates non-interactively based on the linearity of the secret sharing scheme, and the multiplication gates using the above mentioned multiplication protocol. Finally, the (shared) result of the computation

is reconstructed. We show in the following Section 3.5.4 how to achieve security against an active Q^3 adversary. Note that, as in the field case [CDM00], *perfect* security requires a strongly-multiplicative span program, while an (ordinary) multiplicative span program is sufficient for unconditional security.

Theorem 3.2 *Let Λ be a commutative ring with 1, let $\mathcal{M}$ be a (strongly) multiplicative span program over Λ for a Q^3 adversary structure $\mathcal{A}$, and let $\mathcal{C}$ be an arithmetic circuit. Then there exists a (perfectly) $\mathcal{A}$-secure MPC protocol to evaluate $\mathcal{C}$ over an arbitrary finite ring R which can be seen as a Λ-algebra. Its communication complexity (in terms of the number of ring elements to be communicated) is polynomial in n,* size$(\mathcal{M})$ *and $|\mathcal{C}|$, the number of multiplication gates in $\mathcal{C}$.*

Remark 3.7 So far, we have not yet formally defined what is an *$\mathcal{A}$-secure* MPC protocol. We make up for this in Section 3.6.2, where we also show how to prove Theorem 3.2. Informally, an MPC protocol is called *$\mathcal{A}$-secure* if an adversary that corrupts (at most) the players of a set $A \in \mathcal{A}$ "cannot achieve more" than in an ideal setting where the circuit is evaluated by an uncorruptable entity.

Remark 3.8 For simplicity, we restrict to MPC in the sense of *secure function evaluation*, where the players provide the inputs during the *input phase*, compute securely an agreed-upon function (given as arithmetic circuit) on the inputs in the *computation phase* and reconstruct the result in the *output phase*. However, at least in case of a *strongly* multiplicative span program $\mathcal{M}$, Theorem 3.2 also holds with respect to MPC understood as an on-going computation, where the environment (as formalized in Section 3.6.2) specifies "on the fly" which player provides an input at what time, what is computed on those inputs, and which player gets what output at what time.

A formal proof is given in Section 3.6.2. Combining this theorem with the fact that every ring R can be seen as a module over $\mathbb{Z}$ yields

Corollary 3.1 *Let $\mathcal{M}$ be a (strongly) multiplicative* ISP *for a Q^3 adversary structure $\mathcal{A}$, and let $\mathcal{C}$ be an arithmetic circuit. Then there exists a (perfectly) $\mathcal{A}$-secure* black-box *MPC protocol to evaluate $\mathcal{C}$ over an arbitrary finite ring R. Its communication complexity is polynomial in n,* size$(\mathcal{M})$ *and $|\mathcal{C}|$.*

Note that viewing R as an algebra over $\mathbb{Z}$, the corresponding number-multiplication can efficiently be computed using standard "double and add", requiring only black-box access to the addition in R. In Chapter 4 we show how to construct size-optimal threshold ISP's, leading to polynomial-time threshold black-box MPC protocols:

Corollary 3.2 *Let $\mathcal{C}$ be an arithmetic circuit. For $0 \leq t < n/3$, there exists a perfectly $\mathcal{A}_{t,n}$-secure black-box MPC protocol to evaluate $\mathcal{C}$ over an arbitrary finite ring R. Its communication complexity is polynomial in n and $|\mathcal{C}|$.*

It turns out that using a broadcast protocol with optimal message complexity $O(n^2)$ (but $O(n)$ rounds), e.g. [BGP89], this MPC protocol requires $O(|\mathcal{C}|n^6 \log n)$ elements of R to be communicated. Hence, the communication complexity coincides asymptotically with the classical protocols of [BGW88, Bea91, FY92], up to a possible loss of a factor $\log n$ (which is due to the fact that there exist smaller threshold span programs over *large* fields). Furthermore, our protocol is compatible with improvements to the communication complexity of non-black-box MPC over fields [HMP00, HM01], as well as with methods achieving security against a *mixed* adversary ([FHM98, FHM99] and Section 5.6.1).

3.5.4 Achieving Security Against an Active Adversary

Following the paradigm of [CDM00], security against an *active* adversary can be achieved by means of a *linear distributed commitment* (DC) and three corresponding auxiliary protocols: a *commitment transfer protocol* (CTP), a *commitment sharing protocol* (CSP) and a *commitment multiplication protocol* (CMP). A linear distributed commitment allows a player to commit to a secret, however, in contrast to its cryptographic counterpart, a distributed commitment is perfectly hiding *and* binding. A CTP allows to transfer a commitment for a secret from one to another player, a CSP allows to (verifiably) share a committed secret such that the players will be committed to their shares, and a CMP allows to prove that three committed secrets s, s' and s'' satisfy the relation $s'' = ss'$, if this is indeed the case. These protocols allow to modify the passively secure MPC protocol, sketched in the above section, in such a way that at every stage of the MPC every player is committed to his current intermediary results. This guarantees that dishonest behaviour will be detected, and thus leads to a MPC protocol that achieves security against an *active* adversary.

We extend the field based solutions of [CDM00] to our more general setting of MPC over an arbitrary ring. Note, as in [CDM00], for a *perfectly* secure CMP we require a *strongly* multiplicative span program. Furthermore, these protocols require broadcast. We refer to [FM98] for a perfectly secure broadcast protocol for an arbitrary Q^3 adversary structure. However, for simplicity, we pretend that broadcast channels are given as primitives.

Throughout this section, let Λ be a commutative ring with 1, and let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a span program for a Q^3 adversary structure $\mathcal{A}$. Write Γ for the corresponding access structure $\Gamma = \bar{\mathcal{A}}$. In the perfect CMP part, we additionally assume $\mathcal{M}$ to be strongly multiplicative. Finally, R is an arbitrary finite Λ-algebra. The following protocols are secure with respect to an adversary corrupting the players corresponding to an arbitrary set $A \in \mathcal{A}$.

Linear Distributed Commitment. A commitment C of a secret $s \in R$ consists of a *correct* sharing $\mathbf{s} = M\mathbf{b}$ with respect to the linear secret sharing scheme given by the span program $\mathcal{M}$. To *open* such a commitment C, the committer reveals the corresponding sharing vector $\mathbf{b}$ and every player reveals his share $\mathbf{s}_i$, and the opening is accepted if and only if $M_i\mathbf{b} = \mathbf{s}_i$ for all players P_i, except those corresponding to a set $A \in \mathcal{A}$. The hiding property follows from the privacy of the secret sharing scheme, and the binding property

can be verified using the Q^3 property of $\mathcal{A}$ (see also Lemma 5.1). To *commit* to a secret s, it is of course not enough to just let the committer share this value, as the correctness of the shares has to be guaranteed. This can be achieved as follows. Instead of the vector $\mathbf{b}$, the dealer/committer chooses a random symmetrical matrix B with the secret s in the upper left corner and sends $U_i = M_i B$ to player P_i. By pairwise checking, public complaining and accusing it is then guaranteed that $M_j U_i^T = U_j M_i^T$ for every pair P_i, P_j of honest players. This then implies that there exists a vector $\mathbf{b}$ such that $\mathbf{s}_i = M_i \mathbf{b}$ for every honest player P_i, where $\mathbf{s}_i$ denotes the first column of U_i, meaning that the honest players hold a correct sharing of some secret s. Indeed, if A collects the honest players, and hence $A \in \Gamma$, then $M_A U_A^T = U_A M_A^T$ implies that

$$\mathbf{s}_A \;=\; U_A \boldsymbol{\varepsilon} \;=\; U_A M_A^T \boldsymbol{\lambda}(A) \;=\; M_A U_A^T \boldsymbol{\lambda}(A) \;=\; M_A \mathbf{b}$$

for $\mathbf{b} = U_A^T \boldsymbol{\lambda}(A)$. If, for every set $A \in \Gamma$, the span program not only fulfils $\boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$ but in fact $\mathrm{im} M_A^T = \Lambda^e$ (this e.g. holds for the threshold span programs resulting from Corollary 4.2), then it even follows that $U_A = M_A B$ for some matrix B.

An alternative security proof can be found in Section 5.4, based on the framework for linear VSS and distributed commitments proposed in Chapter 5.

Commitment Transfer Protocol (CTP). The purpose of a CTP is to allow a player P_j to transfer a commitment of a secret s to another player P_k such that P_k is committed to the same secret s and is able to open the commitment (to s). It must be guaranteed that this protocol leaks no information to the adversary if P_j and P_k are honest, but also that the new commitment contains the same value as the old, even if P_j and P_k are both corrupt.
If, for every set $A \in \Gamma$, the span program not only fulfils $\boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$ but in fact $\mathrm{im} M_A^T = \Lambda^e$ (this holds for instance for the threshold schemes resulting from Corollary 4.2), then the CTP works as simple as with cryptographic commitments. Namely, to transfer a commitment C, which in our case is nothing else than a correct sharing $\mathbf{s} = M\mathbf{b}$ of the secret s, distributed among the players, C is opened privately to P_k. This means that P_j sends $\mathbf{b}$ and every P_i sends $\mathbf{s}_i$ privately to P_k, and P_k accepts the opening if (and only if) $\mathbf{s}_i = M_i \mathbf{b}$ holds except for $i \in A$ for some set $A \in \mathcal{A}$. P_i's commitment for s is now the same sharing $\mathbf{s}$, and $\mathbf{b}$ is the corresponding "opening information".
However, in the general case, this would *not* lead to a secure CTP, as the adversary could achieve that an honest player P_k would not be able to open the commitment that has been transfered to him by a corrupted player P_j. This can be overcome as follows (see also [CDM00]). P_k does not adopt P_j's commitment C but he generates a new commitment C' for s by the commit protocol, and he opens the difference $C' - C$ to zero (and if he does not succeed, P_j has to open C' in public).

Commitment Sharing Protocol (CSP). A CSP allows a dealer, who is committed to a secret s, to share s such that every player will be committed to his share, and, on the other hand, it is guaranteed that indeed s is correctly shared. A CSP can generically be

constructed from the linear distributed commitment and the corresponding CTP. Namely, the dealer chooses a random sharing vector $\mathbf{b}$ with s as first entry and commits to the random entries (he is already committed to s), and he transfers the by linearity resulting commitments for the shares $\mathbf{s} = M\mathbf{b}$ to the corresponding players, using the CTP. Clearly, this also works for an (arbitrary) span program matrix that is different than M, on which the distributed commitment is based.

Commitment Multiplication Protocol (CMP). A CMP allows a player P_j, who is committed to secrets s, s' and to the product $s'' = ss'$, to prove that indeed $s'' = ss'$. The first solution is based on a multiplicative span program, but is "only" unconditional secure, while the second is perfectly secure but requires a *strongly* multiplicative span program.

Unconditional CMP: Let C, C' and C'' be the commitments of s, s' and $s'' = ss'$, respectively. In order to prove that $s'' = ss'$, the prover chooses a random $d \in R$, and commits to d and ds'. Let D and E denote the resulting commitments. The players jointly generate a random challenge $c \in \{0, 1\}$ using standard techniques (see e.g. Section 7.3.3). Then, the prover opens the commitment $cC + D$ to $r = cs' + d$ and the commitment $rC' - E - cC''$ to 0. The proof is accepted if both openings are accepted.

It is easy to see that the opened values give no extra information to the adversary, while the proof is rejected if $s'' \neq ss'$ except with probability $1/2$ (which can be made negligible small by repetition).

Perfect CMP: First,we introduce some notation. Let $\mathbf{x} = (x_1, \ldots, x_d)$ and $\mathbf{y} = (y_1, \ldots, y_d)$ be two d-dimensional row (or column) vectors over an arbitrary ring R. Then $\mathbf{x} \otimes \mathbf{y}$ denotes the d^2-dimensional row (or column) vector consisting of all products $x_i y_j$:

$$\mathbf{x} \otimes \mathbf{y} \;=\; (x_1y_1, \ldots, x_1y_d; x_2y_1, \ldots, x_2y_d; \ldots\ldots; x_dy_1, \ldots, x_dy_d)$$

Let now $\psi : \{1, \ldots, d\} \to \{1, \ldots, n\}$ be a labeling function. Then $\mathbf{x} \otimes_\psi \mathbf{y}$ denotes the row (or column) vector consisting of all products $x_i y_j$ with $\psi(i) = \psi(j)$:

$$\mathbf{x} \otimes_\psi \mathbf{y} \;=\; (\mathbf{x}_1 \otimes \mathbf{y}_1, \ldots, \mathbf{x}_n \otimes \mathbf{y}_n)$$

We also say that $\mathbf{x} \otimes_\psi \mathbf{y}$ consists of the *locally computable products* of the entries of $\mathbf{x}$ and $\mathbf{y}$.

Now, the perfectly secure CMP is based on the fact that if $\mathcal{M} = (\Lambda, M, \psi, n)$ is strongly multiplicative, then there exists an span program $\mathcal{M}_\otimes$ such that

1. if $\mathbf{s}$ and $\mathbf{s}'$ are sharings of s and s' with respect to $\mathcal{M}$, then $\mathbf{s}'' = \mathbf{s} \otimes_\psi \mathbf{s}'$ is a sharing of $s'' = ss'$ with respect to $\mathcal{M}_\otimes$, and
2. if a set A is rejected by $\mathcal{M}$, then its complement $\bar{A} = \{1, \ldots, n\} \backslash A$ is accepted by $\mathcal{M}_\otimes$.

For instance, in the threshold case, when s and s' have been shared using two polynomials of degree t, multiplying corresponding shares of s and s' yields a sharing of ss', but with respect to a polynomial of degree $2t$. In general, $\mathcal{M}_\otimes = (\Lambda, M_\otimes, \psi_\otimes, n)$ is constructed as

follows. For every $i \in \{1, \ldots, n\}$, and for every pair of (not necessarily different) rows $\mathbf{m}$ and $\mathbf{m}'$ of M that are labeled by i, let $\mathbf{m} \otimes \mathbf{m}'$ be a row of $M_\otimes$, labeled by i. It is not hard to verify that condition 1. follows by construction and 2. from the strong multiplication property.

The CMP now works as follows. The prover applies the CSP to M to share s and s', resulting in sharings $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$, respectively, and commitments for all shares. Furthermore, he applies the CSP to $M_\otimes$ to share $s'' = ss'$, resulting in a sharing $\mathbf{s}'' = M_\otimes \mathbf{b}''$ and corresponding commitments. However, he chooses $\mathbf{b}''$ in such a way that $\mathbf{s}'' = \mathbf{s} \otimes_\psi \mathbf{s}'$. Every player P_i now verifies whether indeed $\mathbf{s}''_i = \mathbf{s}_i \otimes \mathbf{s}'_i$, and, in case this does not hold, shows the dishonestness of the dealer by opening the commitments for $\mathbf{s}_i$, $\mathbf{s}'_i$ and $\mathbf{s}''_i$. If no such triple is opened, the proof is accepted. Therefore, if the proof is accepted, then $\mathbf{s}''_{\bar{A}} = \mathbf{s}_{\bar{A}} \otimes_\psi \mathbf{s}'_{\bar{A}}$, where $A \in \mathcal{A}$ collects the corrupted players, and since $\bar{A}$ is accepted by $\mathcal{M}_\otimes$, the secret "behind" $\mathbf{s}''_{\bar{A}}$ is uniquely defined and thus equal to ss'.

3.6 Defining and Proving the Security of MPC

3.6.1 Definition of Secure MPC

We define the security of an MPC protocol π in the *universal composability* framework of [Can01], adapted to a synchronous network as in [DN03] and to unconditional security. In essence, the security definition requires that for any *real-life* adversary Adv attacking the *real-life* execution of the protocol π, there exists an *ideal-life* adversary Sim which "achieves exactly the same" as Adv by attacking a so called *ideal-life* execution. In this real-life execution, the players simply send their private inputs to a MPC *functionality* $\mathcal{F}_{MPC}$, which is an uncorruptable entity that computes the specified function F on given inputs, and hands the output to the (or some specified) players. The requirement that Sim "achieves exactly the same" as Adv is formalized by introducing another entity, the *environment* Env, whose goal is to distinguish the real-life execution from the ideal-life execution. The environment provides the inputs to and receives the outputs from the honest players and communicates arbitrarily with the adversary (either Adv or Sim), and it produces a binary output. For the protocol π to be secure, it is required that for every Adv there exists Sim such that for all Env the output of Env is (essentially) the same in both cases, when talking with Adv and the players of a real-life execution and when talking with Sim and the players of an ideal-life execution.

While this formulation seems to capture the "right" security requirement, it will be convenient to use a slightly simpler though equivalent formulation: As pointed out in [Can01], the real-life adversary Adv can be eliminated from the definition of secure MPC by viewing it as part of the environment Env. In this formulation, Env (instead of Adv) corrupts players. And, it is required that there exists an ideal-life adversary Sim (now called *simulator*) that simulates the corrupted players in the real-life execution to the environment Env while in fact running the ideal-life execution, such that for any Env, its output in the ideal-life execution is indistinguishable from its output in the real-life execution. We will distinguish between *perfect* and *statistical* indistinguishability.

Definition 3.9 *If* X *and* Y *are two binary random variables (over possibly non-identical but fixed probability spaces), then we write* $\mathrm{X} \sim \mathrm{Y}$ *if* X *and* Y *have the same distribution. If* X *and* Y *are two families of binary random variables, indexed by a security parameter* $k \in \mathbb{N}$*, then we write* $\mathrm{X} \overset{s}{\sim} \mathrm{Y}$ *if*

$$\max_z \left| P[\,\mathrm{X}_k = 0\,] - P[\,\mathrm{Y}_k = 0\,] \right| \;\leq\; negl(k)$$

In the former case X *and* Y *are said to be* perfectly *and in the latter case* statistically indistinguishable.

In contrast to [Can01] where Env is restricted to be poly-time, we put no computational restriction on Env (though we still require Sim to be poly-time). This allows to define unconditional as well as perfect (rather than computational) security. Let $\mathrm{REAL}_{\pi,\mathsf{Env}}$ denote Env's output bit in the real-life execution, when talking with the real players, and let $\mathrm{IDEAL}_{\mathcal{F}_{MPC},\mathsf{Sim},\mathsf{Env}}$ denote Env's output bit in the ideal-life execution, when talking with the simulated corrupted players.

Definition 3.10 *Let* π *be an* n*-party protocol, and let* $\mathcal{A}$ *be an adversary structure. Furthermore, let* $\mathcal{F}_{MPC}$ *be the functionality for evaluating the function* F*. If there exists a poly-time simulator* Sim *such that for any (possibly computationally unbounded) environment* Env*, which corrupts the players of a set* $A \in \mathcal{A}$*, it holds that*

$$\mathrm{REAL}_{\pi,\mathsf{Env}} \;\sim\; \mathrm{IDEAL}_{\mathcal{F}_{MPC},\mathsf{Sim},\mathsf{Env}}\,,$$

then π *is called a* perfectly $\mathcal{A}$-secure *MPC protocol. If* $\mathrm{REAL}_{\pi,\mathsf{Env}} \overset{s}{\sim} \mathrm{IDEAL}_{\mathcal{F}_{MPC},\mathsf{Sim},\mathsf{Env}}$*, then* π *is called an* (unconditionally) $\mathcal{A}$-secure *MPC protocol.*[2]

In the literature, the function F is typically specified by an arithmetic circuit $\mathcal{C}$ over some *fixed* (finite) field. Theorem 3.2 generalizes this paradigm not only in that it allows $\mathcal{C}$ to be a circuit over a (finite) *ring*, but also in that it achieves some sort of "*universality*", in the sense that the same protocol can be used for a whole class of rings over which $\mathcal{C}$ can be considered. In the extreme, it can be used for *any* finite ring. In order to make this precise, we use the following terminology. We call a protocol π *generic* with respect to all finite rings if "it can be applied to any finite ring" in the following sense: π's input and output are binary encodings of elements of a finite ring R, where R as well as the encoding function are arbitrary and not know to π, and π may consult an oracle in order to compute the elementary ring operations (addition, subtraction and multiplication) on encodings of elements of R and in order to generate encodings of random elements of R.[3] Similarly, for a fixed commutative ring Λ with 1, we can define what it shall mean for a protocol to be generic with respect to all finite Λ-algebras, the difference being that R is restricted to be a Λ-algebra, and that π may consult the oracle also in order to compute the number-multiplication on encodings of elements of R.

[2] This can only be meaningfully applied to a protocol that takes a security parameter k as input.

[3] The difference with the notion of a *generic* algorithm used in [Sho97] is the additional quantification over the considered algebraic objects (here the finite rings) while in [Sho97] only the encoding function is variable.

Definition 3.11 *Let π be an n-party protocol, let $\mathcal{A}$ be an adversary structure on $\{1,\ldots,n\}$, let $\mathcal{C}$ be an arithmetic circuit, and let Λ be a commutative ring with 1. We call π a perfectly (or unconditionally) $\mathcal{A}$-secure* black-box *MPC protocol* to evaluate $\mathcal{C}$ over an arbitrary finite ring R, *respectively a perfectly (or unconditionally) $\mathcal{A}$-secure MPC protocol* to evaluate $\mathcal{C}$ over an arbitrary finite ring R which can be seen as a Λ-algebra, *if π is generic with respect to all finite rings, respectively with respect to all finite Λ-algebras, and if π is perfectly (or unconditionally) $\mathcal{A}$-secure in the sense that there exists a generic simulator* Sim *(generic with respect to all finite rings, respectively all finite Λ-algebras) satisfying the requirement from Definition 3.10, independent of R and the encodings of its elements.*

Note that in contrast to π and Sim, the environment Env is not required to be generic, but may depend on R and the encodings of the elements of R.

3.6.2 Proof of Theorem 3.2

The proof is organized as follows. First, we prove Theorem 3.2 in a *hybrid* model where we assume the existence of an *ideal* commitment scheme, given by a functionality $\mathcal{F}_{Com}$, and then we prove secure an implementation of $\mathcal{F}_{Com}$. Theorem 3.2 then follows by a general composition theorem (see [Can01] and [DN03]). Without loss of generality, we may assume that a broadcast channel is available.[4]

Let R be an arbitrary finite ring which can be seen as a Λ-algebra. It will be clear from the exposition below that the MPC protocol as well as the simulator do not make use of the concrete choice of R or of the encodings of the elements, and therefore are generic with respect to all finite Λ-algebras.

We start by describing the functionality $\mathcal{F}_{Com}$ that shall implement the ideal commitment scheme on which upon we will base the MPC protocol. A player P_i commits to a value $s \in R$ by sending s to $\mathcal{F}_{Com}$ who simply records s together with the identity of P_i, the *owner* of s, and sends a confirmation to all the players (and the adversary). And it releases a recorded value s by sending it to all the players as soon as he receives a corresponding open-command from the owner of s. Obviously, this implements an ideal commitment scheme. However, we require some additional properties of $\mathcal{F}_{Com}$. Let s, s', s'' be three committed values with owner P_i. On P_i's request, $\mathcal{F}_{Com}$ records $(\lambda s + \lambda' s', P_i)$ for given parameters λ and λ'. We say that "P_i computes a commitment for $\lambda s + \lambda' s'$". Also, on P_i's request, $\mathcal{F}_{Com}$ sends an affirmation to all players that $s'' = s \cdot s'$ if this indeed the case. We say that "P_i proves that $s'' = s \cdot s'$". And finally, on P_i's request, $\mathcal{F}_{Com}$ records (s, P_j) for a different player, owner, P_j, and sends s to P_j. We say that "P_i transfers the commitment for s to P_j".

To simplify notation, we take the following as implicitly understood: first, for any executed action, $\mathcal{F}_{Com}$ sends a confirmation to all the players (and the adversary), and, second,

[4] Existing broadcast protocols like [BGP89], or [FM98] for a general adversary, are universally composable, and therefore implementing a broadcast channel by such a protocol preserves security by the composition theorem.

in order to identify committed values, every committed s is associated with a unique identifier.

We now describe the input, the computation and the output phase of the hybrid-model MPC protocol that has access to the ideal commitment scheme given by $\mathcal{F}_{Com}$:

Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a multiplicative span program for $\mathcal{A}$. To simplify the notation, we assume that size$(\mathcal{M}) = n$ and $\psi(i) = i$ for $i = 1, \ldots, n$. Furthermore, let D be the diagonal matrix with $M^T D M = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$, guaranteed by the multiplication property, and let $r_1, \ldots, r_d \in \Lambda$ be its diagonal entries.

INPUT PHASE

For every player P_i, and for every input s of P_i, the following is done: First, P_i chooses $\mathbf{b} = (b_1, \ldots, b_e)^T \in R^e$ at random subject to $b_1 = s$ and commits to all entries of $\mathbf{b}$. Then, P_i computes commitments for the entries of $\mathbf{s} = M\mathbf{b}$ and, for every P_j, transfers the commitment for s_j to P_j.

If some (corrupted) player refuses to commit to his input s and to compute commitments for shares of s as instructed, then a default input value is taken and the player's part of the protocol is executed in public with default randomness.

COMPUTATION PHASE

$\mathcal{C}$ is securely evaluated gate by gate as follows on the shared inputs:

Addition: To securely add two shared values s and s', every player P_i simply adds his respective shares s_i and s'_i and computes a commitment for $s_i + s'_i$.

Multiplication with constant: To securely multiply a constant c with a shared value s, every player P_i simply multiplies c with his share s_i and computes a commitment for $c \cdot s_i$.

Multiplication: First, for every player P_i, the following is done.

- P_i computes and commits to $p_i = s_i s'_i$, and proves that $p_i = s_i s'_i$.
- P_i chooses $\mathbf{b}_i = (b_{i1}, \ldots, b_{ie})^T \in R^e$ at random subject to $b_{i1} = p_i$ and commits to all entries of $\mathbf{b}_i$ but the first.
- P_i computes commitments for the entries of $\mathbf{p}_i = M\mathbf{b}_i$.
- For j from 1 to n, P_i transfers the commitment for p_{ij} to P_j.

Then, every player P_j computes $s''_j = \sum_i r_i p_{ij}$ and a commitment for it.

Note that by the use of the commitment scheme, it is guaranteed that dishonest behavior will be detected by all honest players. In such a case, the players can simply restart the

computation phase, but with the detected corrupted player P_i excluded. P_i's input is reconstructed, and his part of the protocol is executed in public with default randomness. If $\mathcal{M}$ is *strongly* multiplicative, then it is not hard to see that the computation can be continued without player P_i and it is not necessary to restart the computation phase. In that case, the MPC protocol is not restricted to do secure function evaluation, but its building blocks allow to do MPC in the sense of an on-going computation as discussed in Remark 3.8.

OUTPUT PHASE

Every player P_i, holding a share s_i of the result of the secure evaluation of the circuit, does the following: It opens the commitment for s_i, and then, receiving s_j for players P_j from a set $A \in \Gamma$, it computes the reconstruction vector $\boldsymbol{\lambda}(A)$ and output $s = \boldsymbol{\lambda}(A)^T \mathbf{s}_A$.

Proposition 3.2 *The above MPC protocol is $\mathcal{A}$-secure in the hybrid model with $\mathcal{F}_{Com}$.*

Proof. First note that using the linearity of the secret sharing scheme, the observations in Section 3.5.2, as well as the properties of the ideal commitment scheme, it is straightforward to verify that the MPC protocol satisfies *correctness* in the sense that it has the same input-output behavior as $\mathcal{F}_{MPC}$, i.e., that the output of the (honest) players coincides with the evaluation of the circuit $\mathcal{C}$ on the players' inputs. This value is well defined as the players have to commit to their inputs, i.e. send them to $\mathcal{F}_{Com}$, and hence the inputs not only of honest but also of corrupted players are well defined.

In order to prove security according to Definition 3.10, we need to construct a simulator Sim that runs an ideal-life execution and simulates a hybrid-model execution, such that for any environment Env it looks as if the hybrid-model execution was running. In order to do that, Sim generates virtual hybrid-model players $\tilde{P}_1, \ldots, \tilde{P}_n$ as well as a copy of the ideal functionality $\mathcal{F}_{Com}$, and it runs as follows the MPC protocol among those players.

First, the input phase. Every player $\tilde{P}_i$ follows the protocol with a default input, say $s = 0 \in R$, until the input phase is done or Env asks to corrupt P_i, and Sim stores the corresponding sharing vector $\mathbf{b}$. If at some point Env does ask to corrupt P_i, Sim proceeds as follows. First, it corrupts the ideal-life player P_i and gets P_i's input s, and it updates $\tilde{P}_i$'s sharing vector $\mathbf{b}$ to a sharing vector for s (rather than 0) by adding $s \cdot \boldsymbol{\kappa}(A)$, where $A \in \mathcal{A}$ contains the currently corrupted players. Accordingly, it updates the corresponding shares of the players not in A. Then, it gives Env control over $\tilde{P}_i$ with its updated memory, as specified by a corruption. By construction, $\tilde{P}_i$'s sharing vector has the same distribution as in a hybrid-model execution (with the same environment). If the corruption takes place *before* $\tilde{P}_i$ could commit to its (default) input, the simulator awaits the value s' that the now adversary-controlled player $\tilde{P}_i$ commits to as input (note that the simulator controls $\mathcal{F}_{Com}$) and instructs the corrupted ideal-life player P_i to submit s' to $\mathcal{F}_{MPC}$ as its input.

In the computation phase, Sim keeps track of all the sharing vectors used by the honest and the corrupted players $\tilde{P}_i$ during the secure multiplications, as well as of the resulting sharing vector of any shared intermediary result. If at some point Env asks to corrupt some player P_i, Sim corrupts the ideal-life player P_i and gets its input s, and it updates as above $\tilde{P}_i$'s sharing vector $\mathbf{b}$ and the corresponding shares without changing the shares of the currently corrupted players. In a similar manner, Sim updates all sharing vectors and all shares that need to be adapted. Note that this concerns only sharing vectors or shares held by honest players. Then, it gives Env control over $\tilde{P}_i$ with its updated memory. Again, the information Env receives has exactly the same distribution as in a hybrid-model execution.

Finally, when it comes to the output phase, Sim holds a sharing vector for the shares of the incorrect result of the computation, incorrect because some players $\tilde{P}_i$ used default inputs. Sim now simply takes the correct result from the corrupted ideal-life players and updates the sharing vector and the shares of the (honest) players $\tilde{P}_i$ accordingly, before letting them execute the output phase.

Since the information Env receives from the corrupted players has exactly the same distribution as in a hybrid-model execution, the output of Env must have the same distribution when running an hybrid-model execution and when running a ideal-life execution with Sim as described above. □

In the following proposition, a $\mathcal{A}$-*secure* implementation of the functionality $\mathcal{F}_{Com}$ is defined in the spirit of Definition 3.10.

Proposition 3.3 *The distributed commitment scheme described in Section 3.5.4 is a $\mathcal{A}$-secure implementation of $\mathcal{F}_{Com}$.*

Depending on the CMP protocol, the security is unconditional or perfect.

Proof sketch. The arguments given in Section 3.5.4 show that the distributed commitment scheme satisfies correctness, in the sense that it has the same input-output behavior as $\mathcal{F}_{Com}$. For the formal proof, we need to construct a simulator Sim that simulates to the environment Env a real-life while running an ideal-life execution, such that Env cannot tell the difference. This can be done similarly to the proof of Proposition 3.2 above. Sim runs the protocol among virtual real-life players $\tilde{P}_1, \ldots, \tilde{P}_n$, using default values for the unknown inputs of the honest players, and updating them and any related information without changing the view of Env whenever Env asks to corrupt some player and Sim learns the inputs by corrupting the corresponding ideal-life player. Note that in order to be able to update a sharing *matrix* B used to commit to a value as described in Section 3.5.4, Sim needs for any set $A \in \mathcal{A}$ a symmetrical matrix $K \in \Lambda^{e \times e}$ with entry 1 in the upper left corner and $M_A K = 0$. It is easy to see that Sim can take the matrix $K = \boldsymbol{\kappa}(A)\boldsymbol{\kappa}(A)^T$. □

3.7 Application: Securely Computing MIN and MAX

Aside from its theoretical value, the study of MPC over non-field rings is motivated by the possibility of embedding useful computation tasks into their richer structure. In this section we demonstrate the potential usefulness of this approach by describing an application to the round-efficient secure computation of the minimum and maximum function.

Suppose there are n players, where each player P_i holds an integer y_i from the set $\{0, 1, \ldots, M\}$. (We consider M to be a feasible quantity.) Our goal is to design protocols for securely evaluating $\min(y_1, \ldots, y_n)$ and $\max(y_1, \ldots, y_n)$ with the following optimization criteria in mind. First, we would like the round complexity to be as small as possible. Second, we want to minimize the communication complexity subject to the latter requirement.

Obviously, a protocol for one function implies a protocol for the other. Therefore, it suffices to concentrate on one of the two: the minimum function as we do here, or the maximum function as it is done in [CFIK03]. Let k be a (statistical) security parameter, and fix a ring $R = \mathbb{Z}_{Q^M}$ where Q is a k-bit prime. We denote the elements of R by $1, 2, \ldots, Q^M = 0$. We utilize the following property of the ring R: R contains (exactly) $M + 1$ ideals, and these ideals are *totally* ordered with respect to containment:

$$(0) \;=\; (Q^M) \;\subset\; (Q^{M-1}) \;\subset\; \cdots \;\subset\; (Q) \;\subset\; (1) \;=\; R\,.$$

We call $j \in \{0, \ldots, M\}$ the *index* of the ideal (Q^j), and, for any $x \in R$, we call the index of the smallest ideal of R that contains x the *index* of x. Consider the degree-2 polynomial

$$p(x_1, \ldots, x_n, r_1, \ldots, r_n) \;=\; \sum_{i=1}^{n} r_i x_i$$

over R, with *input variables* $x_1, \ldots, x_n$ and *randomizing variables* $r_1, \ldots, r_n$, and let $P(x_1, \ldots, x_n)$ denote the distribution of $p(x_1, \ldots, x_n, r_1, \ldots, r_n)$ for uniformly distributed $r_1, \ldots, r_n \in R$. It is not hard to verify that the distribution $P(x_1, \ldots, x_n)$ is uniform over the smallest ideal of R that contains at least one x_i, i.e., over the ideal (Q^j) whose index j coincides with the *smallest* index of the elements $x_1, \ldots, x_n$.

We are now ready to describe the protocol. First, each player i maps its input y_i to a ring element x_i with index y_i, i.e. $x_i = Q^{y_i}$. Next, the players securely sample an element z from the output distribution $P(x_1, \ldots, x_n)$. This task can be reduced to the secure evaluation of the *deterministic* degree-2 polynomial

$$p(x_1, \ldots, x_n, r_{11} + \cdots + r_{1n}, \ldots, r_{n1} + \cdots + r_{nn})$$

over R, where $r_{i1}, \ldots, r_{in} \in R$ are randomly chosen and provided as input by player P_i. Finally, the output of the computation is defined by the minimal ideal of R that contains z; concretely, the output is the largest $j \in \{0, \ldots, M\}$ such that $z \in (Q^j)$, i.e. Q^j divides z. Note that the value of z reveals no information about the inputs y_i except what follows from their maximum, and the protocol produces the correct output except with probability

$1/Q \leq 2^{-(k-1)}$. We stress that a corrupted player P_i cannot pick an "invalid" value for x_i: every $x_i \in R$ is consistent with a possible input $y_i \in \{0, \ldots, M\}$.

We turn to analyze the protocol's efficiency. Recall that our main optimization criterion was the round complexity. The protocol requires the secure evaluation of a single *degree-2* polynomial over R. Using a MPC protocol for rings as developed in Section 3.5, this requires fewer rounds than evaluating degree-3 polynomials or more complex functions.[5] The communication complexity of the protocol is linear in M and polynomial in the number of players. In [CFIK03] several alternative approaches are discussed for securely evaluating the minimum respectively maximum function. All of these alternatives either require more rounds, require a higher communication complexity (quadratic in M), or fail to remain secure against an active adversary.

The polynomial $p(x_1, \ldots, x_n, r_1, \ldots, r_n)$ in the above construction is a so called *randomizing polynomial* in the randomizing-polynomial framework introduced in [IK00]. In this framework, functions are represented by (lists of) low degree polynomials, similar to how the polynomial $p(x_1, \ldots, x_n, r_1, \ldots, r_n)$ represents the minimum function.[6] This is very convenient for the construction of *constant-round* MPC protocols. We refer to [IK00,IK02] and to [CFIK03] for more details.

3.8 Open Problems

The main open question of the chapter is probably that of the (non-)equivalence of linear secret sharing and span programs over arbitrary commutative rings with 1. We know that these notions coincide if Λ is finite (Remark 3.5) and if $\Lambda = \mathbb{Z}$ (Theorem 3.1). On the other hand we have seen that they do not coincide for $\Lambda = \mathbb{Q}$ (Remark 3.4). A complete characterization of the rings Λ for which these two notions coincide would definitely round off this work. For instance it might be the case that the equivalence of linear secret sharing and span programs hold for a commutative ring Λ with 1 if and only if Λ can be finitely generated as a $\mathbb{Z}$-algebra.

Interesting would also be to find further examples of useful computational tasks that allow efficient representations over rings by exploiting the richer structure that rings provide in comparison to fields.

[5] The exact number of rounds being saved depends on the specific setting, e.g. on whether a broadcast channel is available.

[6] To be precise, $p(x_1, \ldots, x_n, r_1, \ldots, r_n)$ represents the minimum-*index* function.

Chapter 4

Optimal Black-Box Threshold Secret Sharing

4.1 Introduction

Light is the task where many share the toil. — Homer

A *black-box* secret sharing scheme as introduced in the previous chapter is one which works over any finite Abelian group G. Briefly, the sharing and the reconstruction procedures consist of taking $\mathbb{Z}$-linear combinations of group elements, and correctness and privacy are guaranteed *regardless* of which group G is chosen. The idea of black-box (threshold) secret sharing was first considered by Desmedt and Frankel [DF89] in the context of distributed cryptosystems based on groups with secret order. There is no immediate adaptation of Shamir's polynomial based secret sharing scheme over finite fields [Sha79] to the setting of black-box secret sharing, due to the fact that polynomial interpolation over the integers does not work as over large enough finite fields.

In [DF94], Desmedt and Frankel proposed a black-box threshold secret sharing scheme for Abelian groups that elegantly circumvents the problem of polynomial interpolation over $\mathbb{Z}$ by passing to an integral extension ring of $\mathbb{Z}$ where polynomial interpolation works "well enough". More precisely, they pass to an integral extension ring of $\mathbb{Z}$ which contains a list of $n+1$ elements $\omega_0, \ldots, \omega_n$ such that all pairwise differences $\omega_i - \omega_j$ $(i \neq j)$ are *invertible*. The maximal number of such elements is called the *Lenstra constant* of the ring. It can be shown that polynomial interpolation works if $\omega_0, \ldots, \omega_n$ are taken as interpolation points. The scheme is then constructed using the fact that sufficient many copies of G can be seen as a module over the extension ring.

The concrete choice of the extension ring made in [DF94] is the ring of p-th cyclotomic integers. For any prime p, the Lenstra constant for the ring of p-th cyclotomic integers is p. Given arbitrary threshold parameters t and n, and choosing p as the smallest prime greater than n, this leads to a black-box secret sharing scheme for arbitrary Abelian groups with expansion factor between n and $2n$.

On the other hand, it was also shown in [DF94] that for every finite Abelian group there exists a (non-black-box) secret sharing scheme, tailored for that particular group, with a worst-case expansion factor $O(\log n)$, leaving a significant a gap between black-box and non-black-box secret sharing.

There are several applications of black-box secret sharing. In the previous chapter, black-box secret sharing schemes are applied in protocols for black-box multi-party computation over arbitrary (finite) rings. The result of [DF94] is exploited in [DDFY94] to obtain an efficient and secure solution for sharing any function out of a certain abstract class of functions, including RSA. The interest in application of black-box secret sharing to practical distributed RSA-based protocols seems to have decreased somewhat due to recent developments, see for instance [Sho00] and the references therein. However, apart from the fact that black-box secret sharing is perhaps interesting in its own right, black-box secret sharing may very well be relevant to new distributed cryptographic schemes.

Except for some quite special cases, namely when t (or $n-t$) is constant or small compared to n [DDB94,BBDW96] or the factor-2 gain from [DKKK98], no substantial improvement on the problem of constructing black-box secret sharing schemes has been reported since the initial results by Desmedt and Frankel. And, further progress via the their approach depends on the problem of finding for each n and integral extension of $\mathbb{Z}$ whose degree is substantially smaller than n and whose Lenstra constant is greater than n. To the best of our knowledge, this is an open problem of algebraic number theory (see also [DF94] and the references therein).

In this chapter, we construct for arbitrary threshold parameters $0 < t < n-1$ a black-box secret sharing scheme with expansion factor $O(\log n)$, closing the gap between non-black-box and black-box secret sharing. Furthermore, we show this to be optimal. Note that the case $t = 0$ is trivial and that the case $t = n-1$ is easily solved by "additive sharing", both resulting in a minimal expansion factor 1. The crucial difference with our approach to the black-box secret sharing problem is that we avoid dependence on Lenstra's constant as defined above. First, we observe that a sufficient condition for black-box threshold secret sharing is the existence (over an integral extension of $\mathbb{Z}$) of *two* lists $\alpha_0, \ldots, \alpha_n$ and $\beta_0, \ldots, \beta_n$ such that all pairwise differences $\alpha_i - \alpha_j$ and $\beta_k - \beta_l$ are *co-prime* (Section 4.4). And, second, we show how to construct *low degree* integral extensions of $\mathbb{Z}$ satisfying this condition (Section 4.6). For an arbitrary given threshold access structure $\Gamma_{t,n}$, this leads to a black-box secret sharing scheme with expansion factor $\lceil \log(n+1) \rceil + 1$. Using a result of Karchmer and Wigderson [KW93], we prove that this is minimal up to an *additive* factor 2 (Section 4.2).

Furthermore, we show that our black-box secret sharing scheme satisfies certain simulatability properties, which are of importance in the context of threshold cryptography, and we sketch how our scheme allows to construct an RSA threshold signature (or encryption) scheme (Section 4.7). In contrast to Shoup's scheme [Sho00], our solution has a considerably more expensive signature generation phase, however it allows a small public exponent like $e = 3$ and is therefore superior in a setting where the complexity of signature verification has to be minimized. Finally, we propose a concrete implementation of our black-box secret sharing scheme and analyze its complexity with respect to bit and (black-box) group operations (Section 4.8). This shows that our scheme not only minimizes the size of the shares but also that it is *practical* with respect to *computing* the shares as well as reconstructing the secret.

The material in this chapter is based on [CF02].

4.2 Lower Bounds for Threshold ISP's

4.2.1 Lower Bounds on the Size

Definition 4.1 *For any access structure* Γ *and for any commutative ring* Λ *with* 1, $\mathrm{msp}_\Lambda(\Gamma)$ *denotes the minimal size of a span program over* Λ *computing* Γ. *If* $\Lambda = \mathbb{Z}$, *we write* $\mathrm{isp}(\Gamma)$.

This is well defined by Remark 3.3.

Proposition 4.1 *For all* t, n *with* $0 < t < n - 1$,

$$\mathrm{isp}(\Gamma_{t,n}) \;\geq\; n \cdot \log \frac{n+3}{2} \;>\; n \log n - n1\,.$$

In order to compare the expansion factor of our black-box secret sharing scheme, the following bound will be convenient.

Corollary 4.1 *The expansion factor of a black-box secret sharing scheme* $\mathcal{M}$ *for* $\Gamma_{t,n}$ *with* $0 < t < n - 1$ *is lower bounded by*

$$\log(n+3) - 1 \;\geq\; \lfloor \log(n+1) \rfloor\,.$$

Proposition 4.1 follows quite directly from the bound shown in Theorem 4.1 for binary span programs, as proved in [KW93]. Before we give the details of the proof of Proposition 4.1, we include a detailed proof of their bound in order to make constants for their asymptotic bound explicit.

Definition 4.2 *The* dual Γ^* *of a access structure* Γ *on* $\{1, \ldots, n\}$ *is the collection of sets* $A \subseteq \{1, \ldots, n\}$ *whose complement* $\bar{A} = \{1, \ldots, n\} \setminus A$ *is not in* Γ.

Note that Γ^* is an access structure on $\{1, \ldots, n\}$, that $(\Gamma^*)^* = \Gamma$, and that $(\Gamma_{t,n})^* = \Gamma_{n-t-1,n}$. The lemma below generalizes a similar property shown in [KW93] for the case of fields.

Lemma 4.1 *Let* Λ *be a commutative ring with* 1, *and let* Γ *be an access structure. Then,*

$$\mathrm{msp}_R(\Gamma) \;=\; \mathrm{msp}_R(\Gamma^*)\,.$$

In the case of span programs over fields, this has been shown in [Gál95]: there exists an explicit though in general inefficient construction that transforms a span program $\mathcal{M}$ into a span program $\mathcal{M}^*$ for the dual access structure with $\mathrm{size}(\mathcal{M}) = \mathrm{size}(\mathcal{M}^*)$. The following proof, due to [Feh99], proposes a construction that is *efficient*.

Proof. Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a span program for Γ. Select an arbitrary generating set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_l$ for $\ker M^T$, and choose $\boldsymbol{\lambda}$ with $M^T\boldsymbol{\lambda} = \boldsymbol{\varepsilon}$. Let M^* be the matrix defined by the $l+1$ columns $(\boldsymbol{\lambda}, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_l)$, and use ψ to label M^* as well. Define $\mathcal{M}^* = (\Lambda, M^*, \psi, n)$. Note that $\text{size}(\mathcal{M}^*) = \text{size}(\mathcal{M})$. We claim that $\mathcal{M}^*$ computes Γ^*. This is easy to verify. If $\bar{A} \notin \Gamma$, then there exists $\boldsymbol{\kappa} \in \Lambda^e$ such that $M_{\bar{A}}\boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. Define $\boldsymbol{\lambda}^* = M_A\boldsymbol{\kappa}$. Then ${M_A^*}^T\boldsymbol{\lambda}^* = {M_A^*}^T M_A\boldsymbol{\kappa} = {M^*}^T M\boldsymbol{\kappa} = (M^T M^*)^T\boldsymbol{\kappa} = \boldsymbol{\varepsilon}^*$, where $\boldsymbol{\varepsilon}^* \in \Lambda^{l+1}$ is the target vector for $\mathcal{M}^*$. The latter equality holds by construction of M^* and by the fact that $\kappa_1 = 1$. On the other hand, if $\bar{A} \in \Gamma$, then there exists $\hat{\boldsymbol{\lambda}} \in \Lambda^d$ such that $M^T\hat{\boldsymbol{\lambda}} = \boldsymbol{\varepsilon}$ and $\hat{\boldsymbol{\lambda}}_A = \mathbf{0}$. By definition of M^*, there exists $\boldsymbol{\kappa} \in \Lambda^{l+1}$ such that $M^*\boldsymbol{\kappa} = \hat{\boldsymbol{\lambda}}$ and $\kappa_1 = 1$. Hence, $M_A^*\boldsymbol{\kappa} = \hat{\boldsymbol{\lambda}}_A = \mathbf{0}$ and $\kappa_1 = 1$. This concludes the proof. □

Remark 4.1 As a side result of this *dual-construction*, we get the following. If Λ is an integral domain (i.e. has no zero divisors), and if $\mathcal{M}$ is a span program over Λ for an access structure Γ with e linear independent columns, then there exists a span program $\mathcal{M}^*$ of equal size for the dual access structure Γ^* with $e^* = \text{size}(\mathcal{M}) - e + 1$ columns.

The following two propositions are due to [KW93] (though we make constants for their asymptotic bound explicit). We write $\text{msp}_2(\Gamma)$ instead of $\text{msp}_{\mathbb{F}_2}(\Gamma)$.

Proposition 4.2 $\text{msp}_2(\Gamma_{1,n}) \geq n \cdot \log n$.

Proof. Consider a span program $\mathcal{M} = (\mathbb{F}_2, M, \psi, n)$ with $\Gamma(\mathcal{M}) = \Gamma_{2,n}$. Without loss of generality, assume that the rows of each M_i are linearly independent over $\mathbb{F}_2$, $i = 1, \ldots, n$. Let H_1 collect the vectors in $\mathbb{F}_2^e$ with first coordinate equal to 1. Since $\{i\} \notin \Gamma_{1,n}$, we have $\ker M_i \cap H_1 \neq \emptyset$. By assumption on M_i, $|\ker M_i \cap H_1| = 2^{e-1-d_i}$ for $i = 1 \ldots n$. On the other hand, $\{i,j\} \in \Gamma_{1,n}$. Hence, we have $\ker M_i \cap \ker M_j \cap H_1 = \ker M_{\{i,j\}} \cap H_1 = \emptyset$, for all i, j with $1 \leq i < j \leq n$. By counting and normalizing, $2^{-d_1} + \cdots + 2^{-d_n} \leq 1$. By the Log Sum Inequality (see e.g. [CT91]), $d = d_1 + \cdots + d_n \geq n \log n$. □

Theorem 4.1 *For all t, n with $0 < t < n - 1$, it holds that*

$$n \cdot \lceil \log(n+1) \rceil \;\geq\; \text{msp}_2(\Gamma_{t,n}) \;\geq\; n \cdot \log \frac{n+3}{2}.$$

Proof. The upper bound, which is not needed for our purposes, follows by considering an appropriate Vandermonde matrix over the field $\mathbb{F}_{2^u}$, where $u = \lceil \log(n+1) \rceil$. This is easily turned into a binary span program for $\Gamma_{t,n}$ using an adaptation of some of the techniques detailed in Section 4.5.

Concerning the lower bound, note that since $\text{msp}_2(\Gamma_{t,n}) = \text{msp}_2(\Gamma_{n-t-1,n})$ by Lemma 4.1,

we may assume $t \geq (n-1)/2$. We have the following estimates.

$$\begin{aligned} \mathrm{msp}_2(\Gamma_{t,n}) &\geq \frac{n}{t+2} \cdot \mathrm{msp}_2(\Gamma_{t,t+2}) \\ &= \frac{n}{t+2} \cdot \mathrm{msp}_2(\Gamma_{1,t+2}) \\ &\geq \frac{n}{t+2} \cdot (t+2) \cdot \log(t+2) \\ &\geq n \cdot \log \frac{n+3}{2} \end{aligned}$$

The first inequality is argued as follows. Consider an arbitrary span program $\mathcal{M} = (\mathbb{F}_2, M, \psi, n)$ for $\Gamma_{t,n}$. Assume without loss of generality that the number of rows in M_i is at most the number of rows in M_{i+1}, $i = 1, \ldots, n-1$. The first $t+2$ blocks $M_1, \ldots, M_{t+2}$ clearly form a span program for $\Gamma_{t,t+2}$. Hence, the total number of rows in these blocks is at least $\mathrm{msp}_2(\Gamma_{t,t+2})$. Each other block M_j with $j > t+2$ has at least as many rows as any of the first $t+2$ blocks. Therefore, M_j has at least $\mathrm{msp}_2(\Gamma_{t,t+2})/(t+2)$ rows. Summing up over all i according to the observations above gives the first inequality. The equality is implied by Lemma 4.1, the second to last inequality follows from Proposition 4.2, and the last one from $t \geq (n-1)/2$. □

For the proof of Proposition 4.1, let an ISP for $\Gamma_{t,n}$ be given, and consider the ISP matrix, but with all entries reduced modulo 2. This way, a binary span program for $\Gamma_{t,n}$ is obtained. The argument is concluded by applying Theorem 4.1. The statement about black-box secret sharing follows from Theorem 3.1.

Note that our lower bound on black-box secret sharing can also be appreciated without reference to Theorem 3.1, by essentially the same argument as above. Namely, setting $G = \mathbb{Z}_2$ in Definition 3.3, we clearly obtain a binary linear secret sharing scheme. This is well-known to be equivalent to a binary span program, as mentioned before. Hence, we can directly apply the bound from Theorem 4.1.

4.2.2 Lower Bounds on the Column Dimension

In this section we give some lower bounds on the number of columns of threshold ISP's. These obviously translate to lower bounds on the randomness needed for black-box threshold secret sharing. Similar bounds were shown in [Kin00, Kin01]; however, our derivation is much simpler. Since these bounds seem to be rather far away from being tight, we conjecture at the end of the section what we think is the "right" lower bound.

Let $0 < t < n-1$, and let $\mathcal{M} = (\mathbb{Z}, M, \psi, n)$ be an ISP for $\Gamma_{t,n}$.

Proposition 4.3 *The number of columns of M is lower bounded by*

$$e \geq \sqrt{2 \log \binom{n}{t}}\,.$$

Proof. Using similar reasoning as for the lower bound on the size of a threshold ISP, it is sufficient to prove the claim for a threshold span program $\mathcal{M} = (\mathbb{F}_2, M, \psi, n)$ over the field $\mathbb{F}_2$. Let $V = \mathbb{F}_2^e$, and, for any $A \subseteq \{1, \ldots, n\}$, let $V_A \subseteq V$ be the subspace spanned by the rows of M_A. Clearly, the number of proper subspaces of V is upper bounded by $2^{e(e-1)/2}$. Furthermore, for $A, A' \subseteq \{1, \ldots, n\}$ of size $|A| = |A'| = t$, the two spaces V_A and $V_{A'}$ must be different, as $\boldsymbol{\varepsilon}$ is neither in V_A nor in $V_{A'}$ but in the span of the union. It follows that $\binom{n}{t} \leq 2^{e(e-1)/2}$, which implies the claim. □

Using that $\binom{n}{t} < 2^n$ it is easy to see that the given lower bound is of order $O(\sqrt{n})$. The following bound is (asymptotically) better for instance when t is linear in n.

Proposition 4.4 *The number of columns of M is lower bounded by*

$$e \;\geq\; \log(n+3) + t - 1\,.$$

Proof. Again, it suffices to prove the claim for a threshold span program $\mathcal{M}$ over $\mathbb{F}_2$. It follows from the lower bound on the size of $\mathcal{M}$, Theorem 4.1, that there exists $i \in \{1, \ldots, n\}$ such that M_i consists of at least $d_i \geq \log(n+3) - 1$ rows. Without loss of generality, we may assume that the rows of M_i are linear independent. Using the notation from the proof above, it follows that $\dim(V_{\{i\}}) \geq \log(n+3) - 1$. On the other hand, it is obvious that $\dim(V_A) < \dim(V_{A\cup\{j\}})$ for any $A \subseteq \{1, \ldots, n\}$ of size $|A| \leq t$ and $j \notin A$. Both together imply that for $A \subseteq \{1, \ldots, n\}$ with $i \in A$ and $|A| = t+1$, the dimension of V_A and hence e is at least $e \geq \log(n+3) + t - 1$. □

If in this proof it were guaranteed that $\dim(V_{A\cup\{j\}}) = \dim(V_A) + \dim(V_{\{j\}})$, rather than just $\dim(V_A) < \dim(V_{A\cup\{j\}})$, then it would prove the following more realistic lower bound, stated as

Conjecture 4.1 *The number of columns of M is lower bounded by $e \geq t \log n - O(n)$.*

The validity of this conjecture, together with the upper-bound results (on d and on e) from the Section 4.6, would have the following implications.

1. The ISP constructed in Section 4.6 is not only (asymptotically) optimal in terms of its size but also in terms of its number of columns. In other words, the bound is tight.
2. Applying the dual-construction from the proof of Lemma 4.1 to an ISP $\mathcal{M}$ which is (asymptotically) optimal in terms of its size, say $\text{size}(\mathcal{M}) = n \log n \pm O(n)$, *as well as* in terms of its number of columns, $e(\mathcal{M}) = t \log n \pm O(n)$, results in an ISP (for the dual access structure) with the same optimality property:

$$\text{size}(\mathcal{M}^*) \;=\; \text{size}(\mathcal{M}) \;=\; n \log n \,\pm\, O(n) \qquad \text{and}$$

$$e(\mathcal{M}^*) \;=\; \text{size}(\mathcal{M}) - e(\mathcal{M}) + 1 \;=\; (n - t - 1) \log n \,\pm\, O(n)\,.$$

4.3 Some Variations of Secret Sharing and Span Programs

In this section we present some variations of the concepts of linear secret sharing and span programs as introduced earlier. Even though those variations do have some stand-alone motivation to be considered, we will only use them as tools to argue about the well motivated concepts of (ordinary) linear secret sharing and span programs.

4.3.1 Solid Secret Sharing

Let Λ be a commutative ring with 1, let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a labeled matrix, and let Γ be an access structure on $\{1, \ldots, n\}$. If $\mathcal{M}$ is a span program for Γ according to Definition 3.3, then $\mathcal{M}$ allows to share a secret s from any finite Λ-module G such that the correctness and privacy conditions of Definition 3.3 are satisfied. In particular, if Λ is finite, $\mathcal{M}$ allows to share a secret from Λ itself. On the other hand, it has been shown in [CK89] that it is impossible to share a secret from Λ with perfect privacy if Λ is infinite but countable, like $\mathbb{Z}$. The following definition specifies when $\mathcal{M}$ allows to share a secret from Λ with some degraded privacy condition when Λ is infinite.

Definition 4.3 *For any $s \in \Lambda$ let us call $\mathbf{s} \in \Lambda^d$ to be a* sharing of s *(with respect to $\mathcal{M}$) if it is of the form $\mathbf{s} = M\mathbf{b}$ for* some $\mathbf{b} \in \Lambda^e$ *with $b_1 = s$. Then, $\mathcal{M}$ is called a* solid *secret sharing scheme over Λ (respectively a* solid *black-box secret sharing scheme in case $\Lambda = \mathbb{Z}$) for Γ if the following holds.*

- *Correctness: If $A \in \Gamma$, then there exists a* reconstruction vector $\boldsymbol{\lambda} \in \Lambda^{d_A}$ *(for A) such that for any $s \in \Lambda$ and any sharing $\mathbf{s}$ of s, it holds that $\boldsymbol{\lambda}^T \mathbf{s}_A = s$.*
- *Privacy: If $A \notin \Gamma$, then there exists a sharing $\mathbf{s}$ of $1 \in \Lambda$ such that $\mathbf{s}_A = \mathbf{0}$.*

The privacy condition is equivalent to saying that if $A \notin \Gamma$, then for any $s, s' \in \Lambda$ and for any sharing $\mathbf{s}$ of s, there exists a sharing $\mathbf{s}'$ of s' with $\mathbf{s}'_A = \mathbf{s}_A$. Loosely speaking, it says that every s' is *possible* to be the shared secret given $\mathbf{s}_A$ with $A \notin \Gamma$, though not every s' is necessarily *equally likely*.

Definition 4.4 *A solid secret sharing scheme $\mathcal{M} = (\Lambda, M, \psi, n)$ is called* multiplicative *if for any $s, s' \in \Lambda$ and any sharings $\mathbf{s}$ and $\mathbf{s}'$ of s and s', respectively, ss' can be computed as a fixed linear combination of the locally computable products of the shares of s and s', i.e., of the entries of $\mathbf{s} \otimes_\psi \mathbf{s}'$ (as defined in Section 3.5.4).*
$\mathcal{M}$ is called strongly *multiplicative if for every strongly rejected set A, there exists such a linear combination that does not require the shares associated with A.*

The proof of the following characterization of span programs in terms of solid secret sharing follows easily by comparing the two definitions.

Proposition 4.5 *The labeled matrix $\mathcal{M}$ is a span program for Γ if and only if it is a solid secret sharing scheme for Γ. Furthermore, if $\mathcal{M}$ is a span program, then it is (strongly)*

multiplicative as a span program if and only if it is (strongly) multiplicative as a solid secret sharing scheme.

In combination with Proposition 3.1, this shows that every solid secret sharing scheme is a linear secret sharing scheme according to Definition 3.3 (respectively 4.5), while a linear secret sharing scheme is solid if and only if it is a span program.

4.3.2 Weak (Solid) Secret Sharing and Weak Span Programs

Let Λ be a commutative ring with 1, and let Γ be an access structure. Let $\gamma, delta \in \Lambda$.

Definition 4.5 *A* δ-weak *(solid) secret sharing scheme over* Λ *for* Γ *is defined as in Definition 3.3 (respectively 4.3), the only difference being the weaker reconstructability condition that, for all* $A \in \Gamma$, $\delta \cdot s$ *is reconstructed from* $\mathbf{s}_A$*, rather than* s *itself.*
A δ*-weak (solid) secret sharing scheme over* $\mathbb{Z}$ *is called a* δ*-weak (solid)* black-box *secret sharing scheme.*

Definition 4.6 *A possibly weak solid secret sharing scheme is called* γ-weak *(strongly) multiplicative if it is (strongly) multiplicative according to Definition 4.4, with the exception that* $\gamma \cdot ss'$ *rather than* ss' *can be computed as a fixed linear combination of the locally computable products of the shares of* s *and* s'*.*

Definition 4.7 *A* δ-weak *span program over* Λ *for* Γ *is defined as in Definition 3.6, the only difference being the weaker acceptance condition that, for all* $A \in \Gamma$, $\delta \cdot \boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$*, rather than* $\boldsymbol{\varepsilon} \in \mathrm{im} M_A^T$.
A δ*-weak span program over* $\mathbb{Z}$ *is called a* δ*-weak* integer *span program or a* δ*-weak* ISP.

Definition 4.8 *A possibly weak span program* $\mathcal{M} = (\Lambda, M, \psi, n)$ *is called* γ-weak *(strongly) multiplicative if it is (strongly) multiplicative according to Definition 3.8, with the exception that* $M^T D M = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$ *is replaced by the weaker requirement that* $M^T D M = \gamma \cdot \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$*.*

It is straightforward to verify that Proposition 3.1, Theorem 3.1 as well as Proposition 4.5 generalize to the weak versions of the primitives involved. In summary, this means that

- Every δ-weak span program is a δ-weak secret sharing scheme (for the same Γ).
- Every δ-weak integer span program is a δ-weak black-box secret sharing scheme (for the same Γ) *and vice versa.*
- Every δ-weak span program is a δ-weak solid secret sharing scheme (for the same Γ) and vice versa.
- Every δ-weak span program is γ-weak (strongly) multiplicative if and only if it is γ-weak (strongly) multiplicative as δ-weak solid secret sharing scheme.

4.4 On the Construction of Threshold Span Programs

4.4.1 The Lenstra Constants

Let Λ be an arbitrary commutative ring with 1.

Definition 4.9 *The* Lenstra Constant *of* Λ*, denoted as* $\mathcal{L}(\Lambda)$*, is defined as the largest integer* l *such that there exists an* l*-tuple* $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_{l-1}) \in \Lambda^l$ *with the property that* $\omega_i - \omega_j \in \Lambda^*$ *for all* $i \neq j$*. If* l *is unbounded then* $\mathcal{L}(\Lambda) = \infty$*.*

Some trivial examples are $\mathcal{L}(K) = |K|$ for any field K, $\mathcal{L}(\mathbb{Z}) = 2$, $\mathcal{L}(\mathbb{Z}_m)$ equals the smallest prime factor of m, and $\mathcal{L}(\{0\}) = \infty$. The Lenstra constant was developed to provide a method to find new Euclidean fields [Len77].

Definition 4.10 *For any positive integer* l *and any* l*-tuple* $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_{l-1}) \in \Lambda^l$*,* $V(\boldsymbol{\omega})$ *denotes the* l*-dimensional* Vandermonde matrix

$$V(\boldsymbol{\omega}) \;=\; \begin{pmatrix} 1 & \omega_0 & \cdots & \omega_0^{l-1} \\ 1 & \omega_1 & \cdots & \omega_1^{l-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_{l-1} & \cdots & \omega_{l-1}^{l-1} \end{pmatrix},$$

and $\Delta(\boldsymbol{\omega})$ *its determinant, which is well known*[1] *to be*

$$\Delta(\boldsymbol{\omega}) \;=\; \prod_{i>j} (\omega_i - \omega_j).$$

Remark 4.2 An l-tuple $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_{l-1}) \in \Lambda^l$ satisfies the property that $\omega_i - \omega_j \in \Lambda^*$ for all $i \neq j$ if and only if $\Delta(\boldsymbol{\omega}) \in \Lambda^*$.

Definition 4.11 *We define the* 2nd-order Lenstra Constant *of* Λ*, denoted as* $\mathcal{L}_2(\Lambda)$*, as the largest integer* l *such that there exist* two l*-tuples* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Lambda^l$ *with the property that* $\Delta(\boldsymbol{\alpha})$ *and* $\Delta(\boldsymbol{\beta})$ *are* co-prime*. If* l *is unbounded then* $\mathcal{L}_2(\Lambda) = \infty$*.*

Remark 4.3 For any l-tuple $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_{l-1}) \in \Lambda^l$ it holds that

$$\Delta(\boldsymbol{\omega}) \;=\; \Delta(0, \omega_1 - \omega_0, \ldots, \omega_{l-1} - \omega_0)\,.$$

Lemma 4.2 *Let* $I \subsetneq \Lambda$ *be a properly contained ideal. Then* $2 \leq \mathcal{L}(\Lambda) \leq \mathcal{L}_2(\Lambda) \leq |\Lambda/I|$*.*

Proof. It is obvious that $2 \leq \mathcal{L}(\Lambda) \leq \mathcal{L}_2(\Lambda)$. For the remaining inequality, let $\boldsymbol{\alpha}, \boldsymbol{\beta}$ be two tuples of length $l > |\Lambda/I|$. Then, $\Delta(\boldsymbol{\alpha})$ and $\Delta(\boldsymbol{\beta})$ and hence the ideal $(\Delta(\boldsymbol{\alpha}), \Delta(\boldsymbol{\beta}))$ are contained in $I \neq \Lambda$. Therefore, $\Delta(\boldsymbol{\alpha})$ and $\Delta(\boldsymbol{\beta})$ are not co-prime. □

[1] See e.g. [Lan97].

4.4.2 Threshold Span Programs

Let Λ be an arbitrary commutative ring with 1, and let $0 < t < n-1$.

Proposition 4.6 *Let $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_n) \in \Lambda^{n+1}$ be an arbitrary $(n+1)$-tuple with $\omega_0 = 0$. Then there exists a $\Delta(\boldsymbol{\omega})$-weak span program $\mathcal{M} = (\Lambda, M, \psi, n)$ for $\Gamma_{t,n}$ of size n. Furthermore, $\mathcal{M}$ is $\Delta(\boldsymbol{\omega})^2$-weak (strongly) multiplicative if and only if $t < n/2$ (respectively $t < n/3$).*

We describe the claimed weak span program $\mathcal{M}$ in terms of weak solid secret sharing. A sharing $\mathbf{s} = (s_1, \ldots, s_n)^T \in \Lambda^n$ of a secret $s \in \Lambda$ is given by

$$s_i \;=\; b(\omega_i)$$

where $b(X) \in \Lambda[X]$ is a polynomial, the *sharing polynomial*, with degree at most t and constant coefficient

$$b(0) \;=\; \omega_1 \cdots \omega_n \cdot s\,.$$

I.e., for $i = 1, \ldots, n$, the i-th row of M is labeled by i and equals $(\omega_1 \cdots \omega_n, \omega_i, \omega_i^2, \ldots, \omega_i^t)$.

If Λ is an integral domain (i.e. has no zero-divisors), then the weak correctness condition follows from Lagrange interpolation in the fraction field of Λ, and dividing out denominators by multiplying the Lagrange interpolation coefficients with all differences $\omega_i - \omega_j$ where $1 \le j < i \le n$. The general case is treated at the end of this section.

For privacy note that for an arbitrary set $A \subseteq \{1, \ldots, n\}$ of size t, the polynomial

$$\kappa(X) \;=\; \prod_{j \notin A} \omega_j \cdot \prod_{i \in A} (\omega_i - X)$$

has degree at most t, constant coefficient $\kappa(0) = \omega_1 \cdots \omega_n \cdot 1$, and satisfies $\kappa(\omega_i) = 0$ for every $i \in A$.

Concerning the multiplication property, note that if $\mathbf{s} = (s_1, \ldots, s_n)^T$ and $\mathbf{s}' = (s'_1, \ldots, s'_n)^T$ are sharings of s and s' with sharing polynomials $b(X)$ and $b'(X)$, respectively, then $(s_1 s'_1, \ldots, s_n s'_n)^T$ is a sharing of $\omega_1 \cdots \omega_n \cdot ss'$ with respect to the degree-$2t$ sharing polynomial $b(X) \cdot b'(X)$. By the weak correctness property (for an arbitrary t), it follows that $\mathcal{M}$ is $\Delta(\boldsymbol{\omega})^2$-weak multiplicative if $2t < n$ and that it is $\Delta(\boldsymbol{\omega})^2$-weak strongly multiplicative if $3t < n$.

Together with Remark 4.2, Proposition 4.6 implies

Corollary 4.2 *If $\mathcal{L}(\Lambda) > n$ then there exists a span program $\mathcal{M} = (\Lambda, M, \psi, n)$ of size n for the threshold access structure $\Gamma_{t,n}$. Furthermore, $\mathcal{M}$ is (strongly) multiplicative if and only if $t < n/2$ (respectively $t < n/3$).*

The first part of this corollary has also been proven in [DF94, DDFY94] and was the basis for the proposed black-box secret sharing scheme, which we briefly recall here. Consider

the ring Λ of p-th cyclotomic integers, where p is the smallest integer prime greater than n. Concretely, $\Lambda = \mathbb{Z}[X]/\Phi_p(X)$, where $\Phi_p(X) = X^{p-1} + \cdots + X + 1$. It is well-known that $\mathcal{L}(\Lambda) = p$. Indeed, let $\omega \in \mathbb{C}$ be a root of Φ_p, i.e., ω is a primitive p-th root of unity. Define $\omega_i = 1 + \omega + \cdots + \omega^{i-1}$, for $i = 1, \ldots, p-1$. It is straightforward to verify that $\Delta(0, \omega_1, \ldots, \omega_{p-1}) \in \Lambda^*$. Thus $\mathcal{L}(\Lambda) \geq p$. On the other hand, by basic algebraic number theory, the integer prime p totally ramifies in this particular ring Λ. This implies that there is a unique prime ideal $\wp \subset \Lambda$ with $\wp \cap \mathbb{Z} = (p)_{\mathbb{Z}}$ and $|\Lambda/\wp| = p$. Thus, by Lemma 4.2, $\mathcal{L}(\Lambda) \leq p$. Applying above Corollary 4.2 results in an ideal secret sharing scheme over Λ for $\Gamma_{t,n}$. By the construction presented in the next section, this can be transformed into a *black-box* secret sharing scheme for $\Gamma_{t,n}$. However, the expansion factor of that scheme equals $p-1$, the degree of $\Phi_p(X)$, which is lower bounded by n (and upper bounded by $2n$ by Bertrand's Postulate).

In order to construct a black-box secret sharing scheme with *minimal* expansion factor $O(\log n)$, we make use of

Corollary 4.3 *If $\mathcal{L}_2(\Lambda) > n$ then there exists a span program $\mathcal{M} = (\Lambda, M, \psi, n)$ of size $2n$ for the threshold access structure $\Gamma_{t,n}$. Furthermore, $\mathcal{M}$ is (strongly) multiplicative if and only if $t < n/2$ (respectively $t < n/3$).*

Proof. The claim is rather obvious from the point of view of (weak) solid secret sharing. Choose $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Lambda^{n+1}$ such that $\Delta(\boldsymbol{\alpha})$ and $\Delta(\boldsymbol{\beta})$ are co-prime. A sharing of a secret $s \in \Lambda$ with respect to $\mathcal{M}$ is given by two independent sharings of s, one with respect to the $\Delta(\boldsymbol{\alpha})$-weak and the other with respect to the $\Delta(\boldsymbol{\beta})$-weak solid secret sharing scheme guaranteed by Proposition 4.6. The claim about the size is obvious. The same holds for the privacy condition. As to correctness, s can be reconstructed from the shares held by any set $A \in \Gamma$ by first reconstructing $\Delta(\boldsymbol{\alpha}) \cdot s$ and $\Delta(\boldsymbol{\beta}) \cdot s$ separately, and finally computing

$$u \cdot (\Delta(\boldsymbol{\alpha}) \cdot s) + v \cdot (\Delta(\boldsymbol{\beta}) \cdot s) \;=\; s$$

where $u, v \in \Lambda$ are such that $u \cdot \Delta(\boldsymbol{\alpha}) + v \cdot \Delta(\boldsymbol{\beta}) = 1$. The (strong) multiplication property can be shown by a similar argument, using that $\Delta(\boldsymbol{\alpha})^2$ and $\Delta(\boldsymbol{\beta})^2$ are co-prime if (and only if) $\Delta(\boldsymbol{\alpha})$ and $\Delta(\boldsymbol{\beta})$ are. □

To finalize the proof of Proposition 4.6, we need the following remark.

Remark 4.4 It is well known[2] that for any square matrix $A \in \Lambda^{k \times k}$ there exists a matrix $\mathrm{adj}(A) \in \Lambda^{k \times k}$ such that $A \cdot \mathrm{adj}(A) = \det(A) \cdot I$. This implies that the linear equation system $A\mathbf{x} = \mathbf{y}$ is solvable at least if $\mathbf{y}$ is a multiple of $\det(A)$. The matrix $\mathrm{adj}(A)$ is the so called *adjoint*[3] of A, which is the transpose of the cofactor matrix. The cofactor matrix is the matrix of determinants of the minors A_{ij} multiplied by $(-1)^{i+j}$, where the (i,j)-th minor of A is the matrix A without the i-th row or the j-th column.

[2] See e.g. [Lan97].

[3] Some authors, like [Lan97], use the expression adjoint to denote what is also known as the conjugate transpose or the adjugate matrix of A.

Proof of Proposition 4.6. It remains to show the correctness property of the proposed solid secret sharing scheme for the case where Λ is not necessarily an integral domain. Let A be a subset of $\{1,\ldots,n\}$ of cardinality $t+1$. We show that $\Delta(\boldsymbol{\omega})\cdot\boldsymbol{\varepsilon} \in \mathrm{im}M_A^T$. We assume without loss of generality that $A=\{1,\ldots,t+1\}$. The matrix M_A is the $(t+1)$-dimensional Vandermonde matrix $V(\omega_1,\ldots,\omega_{t+1})$ with the first column multiplied by $\omega_1\cdots\omega_n$, and hence its determinant is $\det(M_A)=\omega_1\cdots\omega_n\cdot\Delta(\omega_1,\ldots,\omega_{t+1})$ which divides $\Delta(\boldsymbol{\omega})$. By Remark 4.4 it follows that indeed $\Delta(\boldsymbol{\omega})\cdot\boldsymbol{\varepsilon}\in\mathrm{im}M_A^T$. □

4.5 Span Programs over Integral Extensions of $\mathbb{Z}$

The following observations hold in a more general context; however, for simplicity, we tailor them to our need and restrict our attention to extensions over *the integers.*

Definition 4.12 *Let Λ be an extension ring of $\mathbb{Z}$. We call Λ an* integral *extension of $\mathbb{Z}$ if Λ is of the form $\Lambda=\mathbb{Z}[X]/(f(X))$ for some monic, irreducible polynomial $f(X)\in\mathbb{Z}[X]$.*[4] *We define the* degree *of the extension ring Λ, denoted by $[\Lambda:\mathbb{Z}]$, to be the degree of $f(X)$.*

Let $\Lambda=\mathbb{Z}[X]/(f(X))$ be an integral extension of $\mathbb{Z}$, and write $m=[\Lambda:\mathbb{Z}]$. Note that Λ is a commutative ring with 1 and that it has no zero divisors, i.e., it is an integral domain, but that it is *not* a field. Furthermore, let G be either an arbitrary finite Abelian group, or $G=\mathbb{Z}$. In any case, G is a $\mathbb{Z}$-module. We will now argue that there is a group which contains G as a subgroup, and which is a module over Λ. This has also been used in [DF94]. As Λ and G are both $\mathbb{Z}$-modules, the tensor product $\Lambda\otimes G$ is a well defined $\mathbb{Z}$-module too. Consider the (unique) $\mathbb{Z}$-bilinear map $\Lambda\times(\Lambda\otimes G)\to\Lambda\otimes G$ that maps $(\lambda,\mu\otimes g)$ to $\lambda\mu\otimes g$. It is easy to verify that this map makes $\Lambda\otimes G$ into a module over Λ, which is called the *extension of G over* Λ and is denoted by $\Lambda\otimes_{\mathbb{Z}}G$ [Lan97]. Note that by identifying $g\in G$ with $1\otimes g\in\Lambda\otimes_{\mathbb{Z}}G$, the group G can be viewed as a subgroup of $\Lambda\otimes_{\mathbb{Z}}G$.[5] Finally, if $G=\mathbb{Z}$ then $\Lambda\otimes_{\mathbb{Z}}G$ is isomorphic to Λ, and hence we will identify $\Lambda\otimes_{\mathbb{Z}}G$ with Λ in that case.

Intuitively, these observations immediately imply that a (solid) secret sharing scheme over Λ induces a (solid) black-box secret sharing scheme: to share a secret $s\in G$, where G is an arbitrary finite Abelian group (respectively $G=\mathbb{Z}$), view s as an element of $\Lambda\otimes_{\mathbb{Z}}G$ and apply the (solid) secret sharing scheme over Λ. However, in order to fit into the framework of our formal definition of (solid) secret sharing, we need to take a closer look at how to represent the elements of $\Lambda\otimes_{\mathbb{Z}}G$.

For that, fix $w\in\Lambda$ such that $f(w)=0$ (such as $w=\overline{X}$, the residue class of X modulo $f(X)$) and thus $\Lambda=\mathbb{Z}[w]$. Then for each $\lambda\in\Lambda$, there exists a unique *coordinate-vector* $\vec{\lambda}=(\lambda_0,\ldots,\lambda_{m-1})^T\in\mathbb{Z}^m$ such that $\lambda=\lambda_0\cdot 1+\lambda_1\cdot w+\cdots+\lambda_{m-1}\cdot w^{m-1}$. In other

[4] In standard terminology from algebraic number theory, Λ corresponds to an order in the ring $\mathbb{Z}_K$ of algebraic integers in the number field $K=\mathbb{Q}[X]/(f(X))$. Note that $\Lambda\subseteq\mathbb{Z}_K$, but not necessarily with equality.

[5] We sometimes fail to distinguish between g and $1\otimes g$.

words, $\{1, w, \ldots, w^{m-1}\}$ is a basis for Λ when viewed as a $\mathbb{Z}$-module. Similarly, for each $g \in \Lambda \otimes_{\mathbb{Z}} G$, there exists a unique *coordinate-vector* $\vec{g} = (g_0, \ldots, g_{m-1})^T \in G^m$ such that g can be written in a unique way as $g = 1 \otimes g_0 + w \otimes g_1 + \cdots + w^{m-1} \otimes g_{m-1}$. Note that in case $G = \mathbb{Z}$, and thus $\Lambda \otimes_{\mathbb{Z}} G \simeq \Lambda$, the coordinate vector of an element $\lambda \in \Lambda$ coincides with the coordinate vector of λ when viewed as an element of $\Lambda \otimes_{\mathbb{Z}} G$. Finally, for every $\lambda \in \Lambda$, there exists a matrix $[\lambda] \in \mathbb{Z}^{m \times m}$ such that $\overrightarrow{\lambda \cdot g} = [\lambda]\vec{g}$ holds for every $g \in \Lambda \otimes_{\mathbb{Z}} G$. The columns of $[\lambda]$ are simply the coordinate vectors of $\lambda, \lambda \cdot w, \ldots, \lambda \cdot w^{m-1}$, as can easily be verified.

Remark 4.5 In the special case where G is a finite ring R or $G = R = \mathbb{Z}$, the extension $\Lambda \otimes_{\mathbb{Z}} R$ of R over Λ is in fact a Λ-*algebra*, and R is a sub*ring* of $\Lambda \otimes_{\mathbb{Z}} R$. Also, for $a, b \in R$, the entries of the coordinate vector of ab are $\mathbb{Z}$-linear combinations of the pairwise products $a_i b_j$ $(0 \leq i, j \leq m-1)$ of the entries of the coordinate vectors $\vec{a} = (a_0, \ldots, a_{m-1})^T$ and $\vec{b} = (b_0, \ldots, b_{m-1})^T$ of a and b.

It follows

Lemma 4.3 *Consider an integral extension Λ of $\mathbb{Z}$. Let $\mathcal{M}$ be a span program over Λ for an access structure Γ. Then there exists an ISP $\hat{\mathcal{M}}$ for Γ with* $\text{size}(\hat{\mathcal{M}}) = [\Lambda : \mathbb{Z}] \cdot \text{size}(\mathcal{M})$. *Furthermore, if $\mathcal{M}$ is (strongly) multiplicative then this also holds for $\hat{\mathcal{M}}$.*

Proof. Write $m = [\Lambda : \mathbb{Z}]$. $\hat{\mathcal{M}} = (\mathbb{Z}, \hat{M}, \hat{\psi}, n)$ is constructed from $\mathcal{M} = (\Lambda, M, \psi, n)$ by replacing each entry $\lambda \in \Lambda$ of M by the matrix $[\lambda] \in \mathbb{Z}^{m \times m}$, and then removing the 2nd up to the m-th leftmost columns. Hence, if M consists of d rows and e columns, then $\hat{M}$ results in dm rows and $(e-1)m + 1$ columns. The labeling ψ is extended to $\hat{\psi}$ in the obvious way: if a row of M is labeled by some $i \in \{1, \ldots, n\}$, then the rows it is substituted with in $\hat{M}$ are labeled by i too.

We show that $\hat{\mathcal{M}}$ is a solid secret sharing scheme if $\mathcal{M}$ is. Consider $s \in \mathbb{Z}$ and $\mathbf{s} \in \Lambda^d$, and let $\vec{\mathbf{s}} \in \mathbb{Z}^{dm}$ denote the vector $\mathbf{s}$ with each entry replaced by its coordinate vector. It follows by construction of $\hat{\mathcal{M}}$ that $\mathbf{s}$ is a sharing of s, viewed as an element of Λ, with respect to $\mathcal{M}$, if and only if $\vec{\mathbf{s}}$ is a sharing of s with respect to $\hat{\mathcal{M}}$. It is now easy to see that the correctness, the privacy and the possible (strong) multiplication property carry over from $\mathcal{M}$ to $\hat{\mathcal{M}}$: The privacy condition is obvious. As for correctness, note that if $\boldsymbol{\lambda}^T \mathbf{s}_A = s$ for some $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{d_A})^T \in \Lambda^{d_A}$, then $\hat{\boldsymbol{\lambda}}^T \vec{\mathbf{s}}_A = s$, where $\hat{\boldsymbol{\lambda}}^T$ stands for the first row of the matrix $([\lambda_1], \ldots, [\lambda_{d_A}])$. The claim concerning the (strong) multiplication property follows by using a similar argument in combination with the second part of Remark 4.5 above. □

Remark 4.6 A similar construction obviously applies to *weak* span programs: a δ-weak span program over Λ with $\delta \in \mathbb{Z}$ (rather than in Λ) results in a δ-weak ISP, while a δ-weak span program over Λ with $\delta \in \Lambda$ results in a δ_0-weak ISP, where δ_0 is the first coordinate of δ's coordinate vector $\vec{\delta}$.

4.6 Optimal Black-Box Threshold Secret Sharing

In this section, we prove the main result of this chapter:

Theorem 4.2 *For all integers t, n with $0 < t < n-1$, there exists an ISP $\mathcal{M}$ for $\Gamma_{t,n}$ of size $O(n \log n)$, and hence $\mathrm{isp}(\Gamma_{t,n}) = \Theta(n \log n)$. Furthermore, $\mathcal{M}$ is (strongly) multiplicative if and only if $t < n/2$ (respectively $t < n/3$).*

Corollary 4.4 *For all integers t, n with $1 < t < n-1$, there exists a black-box secret sharing scheme $\mathcal{M}$ for $\Gamma_{t,n}$ with expansion factor $O(\log n)$, which is minimal.*

The minimality follows from Proposition 4.1. In order to prove the existence it suffices by Corollary 4.3 and Lemma 4.3 to construct an integral extension Λ of $\mathbb{Z}$ with 2nd-order Lenstra Constant $\mathcal{L}_2(\Lambda) > n$ and degree $[\Lambda : \mathbb{Z}] = O(\log n)$.

As an aside, both the scheme from [DF94] as well as ours allow that the secret s is in fact chosen as a vector of group elements, rather than just a single group element. Specifically, instead of identifying the secret $s \in G$ with $1 \otimes s \in \Lambda \otimes_{\mathbb{Z}} G$, we may as well choose $s \in \Lambda \otimes_{\mathbb{Z}} G$ arbitrarily. It is easily verified that, with this change, reconstructability and privacy are still satisfied. Thus, from that point of view, both schemes are "ideal," since the entropy of each individual share $\mathbf{s}_i$ is equal to the entropy of the secret. The important difference to note, however, is that a smaller expansion factor is a significant advantage for instance in a setting where the secret is a single group element $s \in G$ for some given "black-box group" G, such as is the case for distributed cryptosystems and for secure general multi-party computation protocols.

4.6.1 Low-Degree Integral Extensions of $\mathbb{Z}$ with $\mathcal{L}_2(\Lambda) > n$

Before we finalize the proof of Theorem 4.2 in Proposition 4.7 below, we state some elementary properties of the Lenstra constants $\mathcal{L}$ and $\mathcal{L}_2$. The following three lemmas hold for an arbitrary commutative ring Λ with 1.

Lemma 4.4 *If Λ is the direct product of rings $\Lambda_1, \ldots, \Lambda_m$, i.e., $\Lambda = \Lambda_1 \times \cdots \times \Lambda_m$, then*

$$\mathcal{L}(\Lambda) \;=\; \min_{1 \le i \le m} \mathcal{L}(\Lambda_i)\,.$$

The proof of this lemma is trivial.

Lemma 4.5 *If I is an ideal in Λ, and $j \ge 1$ is an arbitrary integer, then*

$$\mathcal{L}(\Lambda/I) \;=\; \mathcal{L}(\Lambda/I^j)\,.$$

Proof. The natural ring homomorphism $\phi : \Lambda/I^j \to \Lambda/I$ has the elementary property that $\phi(x) \in (\Lambda/I)^*$ if and only if $x \in (\Lambda/I^j)^*$. The lemma follows. □

Lemma 4.6 *$\mathcal{L}_2(\Lambda) \geq n+1$ if and only if there exists $\boldsymbol{\alpha} \in \Lambda^{n+1}$ with $\mathcal{L}(\Lambda/(\Delta(\boldsymbol{\alpha}))) \geq n+1$.*

Proof. $\mathcal{L}_2(\Lambda) \geq n+1$ holds if and only if there exist $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Lambda^{n+1}$ such that $\Delta(\boldsymbol{\alpha})$ and $\Delta(\boldsymbol{\beta})$ are co-prime, i.e., such that $\overline{\Delta(\boldsymbol{\beta})} \in (\Lambda/(\Delta(\boldsymbol{\alpha})))^*$. This, though, is equivalent to the existence of $\boldsymbol{\alpha} \in \Lambda^{n+1}$ with $\mathcal{L}(\Lambda/(\Delta(\boldsymbol{\alpha}))) \geq n+1$. □

Proposition 4.7 *For any $n > 1$, there is an integer extension Λ such that $\mathcal{L}_2(\Lambda) \geq n+1$ and $[\Lambda : \mathbb{Z}] = \lceil \log(n+1) \rceil$.*

Proof. Fix $\boldsymbol{\alpha} = (0, 1, \ldots, n) \in \mathbb{Z}^{n+1}$. Let Π_n denote the set of integer primes p with $2 \leq p \leq n$. Note that $\Delta(\boldsymbol{\alpha})$ factors as an integer completely over the primes $p \in \Pi_n$, and let

$$\Delta(\boldsymbol{\alpha}) \;=\; \prod_{p \in \Pi_n} p^{t_p}$$

denote its prime factorization.

For the moment, let $\Lambda = \mathbb{Z}[X]/(f(X))$ be an arbitrary integer extension. Then the ideals (p^{t_p}) of Λ with $p \in \Pi_n$ are pair-wise co-prime. Therefore, using the Chinese Remainder Theorem for general rings,

$$\Lambda/(\Delta(\boldsymbol{\alpha})) \;\simeq\; \prod_{p \in \Pi_n} \Lambda/(p^{t_p})$$

For $p \in \Pi_n$, let $f_p(X) \in \mathbb{F}_p[X]$ denote the polynomial $f(X)$ with its coefficients reduced modulo p, and let

$$f_p(X) \;=\; \prod_{k=1}^{l_p} f_{p,k}(X)^{e_{p,k}}$$

denote its factorization in $\mathbb{F}_p[X]$. Furthermore, for each $p \in \Pi_n$, define $r_{p,k} = \deg(f_{p,k})$ and let r_p denote the minimum taken over the $r_{p,k}$'s.

Using the Chinese Remainder Theorem for general rings, it follows that, for each $p \in \Pi_n$,

$$\Lambda/(p) \;\simeq\; \mathbb{Z}[X]/(p, f(X)) \;\simeq\; \mathbb{F}_p[X]/(f_p(X)) \;\simeq\; \prod_{k=1}^{l_p} \mathbb{F}_p[X]/(f_{p,k}(X)^{e_{p,k}})$$

Applying Lemmas 4.4 and 4.5 twice, and using the fact that

$$\mathcal{L}(\mathbb{F}_p[X]/(f_{p,k}(X))) \;=\; \mathcal{L}(\mathbb{F}_{p^{r_{p,k}}}) \;=\; |\mathbb{F}_{p^{r_{p,k}}}| \;=\; p^{r_{p,k}},$$

it follows that

$$\mathcal{L}(\Lambda/(\Delta(\boldsymbol{\alpha}))) \;=\; \min_{p \in \Pi_n} p^{r_p}.$$

Now let $f(X) \in \mathbb{Z}[X]$ be a monic, irreducible polynomial $f(X) \in \mathbb{Z}[X]$ such that

1. for all $p \in \Pi_n$, it holds that $p^{r_p} \geq n+1$.
2. $\deg(f) = \lceil \log(n+1) \rceil$.

Applying Lemma 4.6, it follows that the ring $\Lambda = \mathbb{Z}[X]/(f(X))$ satisfies $\mathcal{L}_2(\Lambda) \geq n+1$ and $[\Lambda : \mathbb{Z}] = \lceil \log(n+1) \rceil$.

We conclude by arguing that such polynomials $f(X)$ can be constructed (efficiently). Set $m = \lceil \log(n+1) \rceil$. For all $p \in \Pi_n$, select an arbitrary monic polynomial $f_p(X) \in \mathbb{F}_p[X]$ of degree m such that $p^{r_p} \geq n+1$. As a special case, the $f_p(X)$ can be chosen as irreducible polynomials of degree m, which is possible by the theory of finite fields. In all cases, the Chinese Remainder Theorem can be applied to each of the coefficients separately and an arbitrary lift to a monic polynomial $f(X) \in \mathbb{Z}[X]$ of degree m can be selected such that $f(X) \equiv f_p(X) \bmod p$, for all $p \in \Pi_n$. By construction, $f(X)$ is irreducible, since if not, the condition $p^{r_p} \geq n+1$ does not hold for $p = 2$. □

Remark 4.7 As an aside, the proof can be easily adapted such that for given $m > 1$, the constructed ring Λ has degree m and $\mathcal{L}_2(\Lambda) = 2^m$, which is maximal by Lemma 4.2 with $I = (2)_\Lambda$.

4.6.2 Some Modifications

Consider $\Lambda = \mathbb{Z}[X]/(f(X))$, $\boldsymbol{\alpha} = (0, \alpha_1, \ldots, \alpha_n)$ and $\boldsymbol{\beta} = (0, \beta_1, \ldots, \beta_n)$ as guaranteed by Proposition 4.7. Since $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}$, the corresponding $\Delta(\boldsymbol{\alpha})$-weak secret sharing scheme can be defined over $\mathbb{Z}$, rather than over Λ, and thus has expansion factor 1. It follows

Corollary 4.5 *For all t, n with $0 < t < n-1$, there exists a black-box secret sharing scheme for $\Gamma_{t,n}$ with expansion factor $\lceil \log(n+1) \rceil + 1$.*

By the explicit lower bound from Corollary 4.1, it follows that the expansion factor of this scheme is minimal, up to an *additive* factor of 2.

The next and last two modifications, Corollaries 4.6 and 4.7, do not modify the scheme per se but the way to reconstruct the shared secret $s \in G$. Note that so far it is guaranteed that s can be reconstructed by reconstructing $\Delta(\boldsymbol{\alpha}) \cdot s$ and $\Delta(\boldsymbol{\beta}) \cdot s$, both in $\Lambda \otimes_{\mathbb{Z}} G$, and computing s as a Λ-linear combination of the two.

Definition 4.13 *For arbitrary $\boldsymbol{\omega} \in \Lambda^{n+1}$, $\delta(\boldsymbol{\omega})$ denotes the unique non-negative integer such that the ideal generated by $\Delta(\boldsymbol{\omega})$ in Λ, intersected with $\mathbb{Z}$, is generated by $\delta(\boldsymbol{\omega})$ as an ideal in $\mathbb{Z}$: $(\Delta(\boldsymbol{\omega}))_\Lambda \cap \mathbb{Z} = (\delta(\boldsymbol{\omega}))_\mathbb{Z}$.*

Lemma 4.7 *For $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as above, $\delta(\boldsymbol{\alpha})$ and $\delta(\boldsymbol{\beta})$ are non-zero and co-prime.*

Proof. Since $\Delta(\boldsymbol{\alpha}) \in \mathbb{Z}$, we have $\delta(\boldsymbol{\alpha}) = \Delta(\boldsymbol{\alpha}) \neq 0$. Concerning $\delta(\boldsymbol{\beta})$, let $h(X) \in \mathbb{Z}[X]$ be an arbitrary lift of $\Delta(\boldsymbol{\beta})$. Since $f(X)$ is irreducible and does not divide $h(X)$, applying the

extended Euclidean algorithm and clearing denominators results in $w_0(X), w_1(X) \in \mathbb{Z}[X]$ and a non-zero integer a such that $w_0(X) \cdot f(X) + w_1(X) \cdot h(X) = a$. It follows that $\delta(\boldsymbol{\beta}) \neq 0$. Now, assume that $\delta(\boldsymbol{\alpha})$ and $\delta(\boldsymbol{\beta})$ are not co-prime. Then, there exists a prime $p \in \Pi_n$ that divides $\delta(\boldsymbol{\beta})$. Let $\mu \in \Lambda$ be such that $\mu \cdot \Delta(\boldsymbol{\beta}) = \delta(\boldsymbol{\beta})$. Since $\Delta(\boldsymbol{\beta})$ is invertible modulo p, μ must be a multiple of p. Hence, we can write $\mu = p \cdot \nu$ for some $\nu \in \Lambda$. Consider $\nu \cdot \Delta(\boldsymbol{\beta}) \in \Lambda$. It holds that $p \cdot \nu \cdot \Delta(\boldsymbol{\beta}) = \delta(\boldsymbol{\beta})$. It is now easy to see that this implies that $\nu \cdot \Delta(\boldsymbol{\beta}) = \delta(\boldsymbol{\beta})/p \in \mathbb{Z}$: if $p \cdot g(X) = \delta(\boldsymbol{\beta}) + k(X)f(X)$ for some $g(X), k(X) \in \mathbb{Z}[X]$, then $k(X)$ must be a multiple of p, and hence $g(X) \equiv \delta(\boldsymbol{\beta})/p \pmod{f(X)}$. However, $\nu \cdot \Delta(\boldsymbol{\beta}) = \delta(\boldsymbol{\beta})/p$ contradicts the definition of $\delta(\boldsymbol{\beta})$. This finishes the proof. □

As a consequence, we have

Corollary 4.6 *In the black-box secret sharing scheme from Theorem 4.2, as well as in the one from Corollary 4.5, the secret can be reconstructed by reconstructing $\delta(\boldsymbol{\alpha}) \cdot s$ and $\delta(\boldsymbol{\beta}) \cdot s$, and computing*

$$s \;=\; u \cdot (\delta(\boldsymbol{\alpha}) \cdot s) + v \cdot (\delta(\boldsymbol{\beta}) \cdot s)$$

for integers *u and v such that $u \cdot \delta(\boldsymbol{\alpha}) + v \cdot \delta(\boldsymbol{\beta}) = 1$.*

Finally, it is not hard to verify (see [Sho00]), that in order to clear the denominators of the Lagrange reconstruction coefficients in the $\Delta(\boldsymbol{\omega})$-weak secret sharing scheme from Proposition 4.6 with $\boldsymbol{\omega} = \boldsymbol{\alpha}$, it suffices to multiply them with $n!$ rather than with $\Delta(\boldsymbol{\alpha}) = \delta(\boldsymbol{\alpha})$. Thus, this scheme is in fact a $(n!)^2$-weak (black-box) secret sharing scheme. This does *not* apply to the $\Delta(\boldsymbol{\beta})$-weak (respectively $\delta(\boldsymbol{\beta})$-weak) scheme. It follows

Corollary 4.7 *In the black-box secret sharing scheme from Theorem 4.2, as well as in the above modifications, the secrets can be reconstructed by reconstructing $(n!)^2 \cdot s$ and $\Delta(\boldsymbol{\beta}) \cdot s$ respectively $\delta(\boldsymbol{\beta}) \cdot s$, and computing s as a linear combination of the two.*

4.7 Simulatability and Application to Threshold Crypto

4.7.1 Share Completion

The black-box secret sharing scheme from Theorem 4.2 as well as the above mentioned modifications additionally satisfy the following *share-completion* property, which is helpful when proving the security of certain threshold cryptosystems. Using the terminology and notation from the definition of (black-box) secret sharing, this property requires additional elements $\mathbf{s}_1^*, \ldots, \mathbf{s}_n^*$ such that

- each $\mathbf{s}_i^*$ is obtained by taking fixed linear combinations (only depending on i) over the coordinates of the $\mathbf{s}_i$.
- the reconstructability property also holds with respect to the $\mathbf{s}_i^*$'s (for appropriate reconstruction vectors).

- for any set $A \notin \Gamma$ that is maximal in the sense that $A \cup \{j\} \in \Gamma$ for any $j \notin A$, the elements $\mathbf{s}_1^*, \ldots, \mathbf{s}_n^*$ are uniquely determined by taking fixed linear combinations (only depending on A) over the coordinates of the $\mathbf{s}_i$ with $i \in A$ and the secret s.

This definition is inspired by an argument developed in [Sho00]. We also define the share-completion property for δ-weak black-box secret sharing schemes, with obvious modification to the definition above.

Proposition 4.8 *The black-box secret sharing scheme from Theorem 4.2, as well as its modifications from Section 4.6.2, satisfy the share-completion property.*

Proof. The proposition follows from the fact that the $\Delta(\boldsymbol{\omega})$-weak secret sharing schemes implied by Proposition 4.6 satisfy the share-completion property: Define $s_i^* = \Delta(\boldsymbol{\omega}) \cdot s_i$ and note that any $t+1$ elements s_i^* allow to reconstruct $\Delta(\boldsymbol{\omega})^2 \cdot s$, while similar arguments as in the proof of Proposition 4.6 imply that t shares s_i together with the secret s allow to compute s_j^* for any j. □

4.7.2 Uniformity

Another additional property a secret sharing scheme may have, is *uniformity*. By this we mean that for all sets $A \subseteq \{1, \ldots, n\}$ with $|A| = t$, the shares $\{\mathbf{s}_i\}_{i \in A}$ are distributed uniformly at random. This property is not satisfied by our schemes. However, we show how to achieve some sort of uniformity if an upper bound on the exponent of the group is known. This is in particular the case when an upper bound on the size of the group is known. We note that the black-box secret sharing scheme from [DF94] has the share-completion as well as the uniformity property.

Let Λ be an arbitrary integral extension of $\mathbb{Z}$. For threshold parameters $0 < t < n-1$, consider the $\Delta(\boldsymbol{\omega})$-weak secret sharing scheme over Λ resulting from Proposition 4.6, which in particular is a $\delta(\boldsymbol{\omega})$-weak secret sharing scheme over Λ and hence gives rise to a $\delta(\boldsymbol{\omega})$-weak *black-box* secret sharing scheme by Remark 4.6.

Lemma 4.8 *The considered $\delta(\boldsymbol{\omega})$-weak black-box secret sharing scheme satisfies the uniformity property when applied to a group G* whose exponent is co-prime with $\delta(\boldsymbol{\omega})$.

Proof. Note that for any Abelian group G, the exponent of $\Lambda \otimes_{\mathbb{Z}} G$ equals the exponent of G, and the coordinate vector of a random element of $\Lambda \otimes_{\mathbb{Z}} G$ consists of random elements of G. Therefore, it suffices to show that the $\Delta(\boldsymbol{\omega})$-weak secret sharing scheme over Λ satisfies the uniformity property when applied to a Λ-module $\hat{G}$ whose exponent is co-prime with $\delta(\boldsymbol{\omega})$. Write $\mathcal{M} = (\Lambda, M, \psi, n)$ for the considered $\Delta(\boldsymbol{\omega})$-weak secret sharing scheme. Recall that $M \in \Lambda^{n \times (t+1)}$. Let $A \subset \{1, \ldots, n\}$ be of size $|A| = t$. Consider the square matrix obtained by adding the target vector $\boldsymbol{\varepsilon}$ of $\mathcal{M}$ as new row to the matrix M_A. This matrix is invertible modulo the exponent of $\hat{G}$: by construction, its determinant divides

$\Delta(\boldsymbol{\omega})$, which is co-prime with $\hat{G}$'s exponent by assumption. It follows that for any secret $s \in \hat{G}$, there exists a one-to-one correspondence between the shares $\mathbf{s}_A = M_A\mathbf{b} \in \hat{G}^t$ and the randomly chosen vector $\mathbf{b} \in \hat{G}^{t+1}$ with first entry being s. Therefore, $\mathbf{s}_A$ is distributed uniformly at random over $\hat{G}^t$. □

It follows that, for any positive integer B, the considered $\delta(\boldsymbol{\omega})$-weak black-box secret sharing scheme gives rise to a $\delta(\boldsymbol{\omega})^{B+1}$-weak black-box secret sharing scheme which satisfies the uniformity property when applied to a group G *whose exponent is upper bounded by* 2^B: to share $s \in G$ simply share $\delta(\boldsymbol{\omega})^B \cdot s$ over the subgroup $\delta(\boldsymbol{\omega})^B \cdot G$ using the $\delta(\boldsymbol{\omega})$-weak black-box secret sharing scheme. By the restriction on G, it is guaranteed that the exponent of $\delta(\boldsymbol{\omega})^B \cdot G$ is co-prime with $\delta(\boldsymbol{\omega})$. Note that if additional information on G's exponent is known like the factors in common with $\delta(\boldsymbol{\omega})$ (see e.g. Section 4.7.3), then it is not necessary to use the full generality of this construction.

Let now the integral extension Λ as well as $\boldsymbol{\alpha} = (0, 1, \ldots, n)$ and $\boldsymbol{\beta} = (0, \beta_1, \ldots, \beta_n)$ be as guaranteed by Proposition 4.7, and consider the corresponding $\delta(\boldsymbol{\alpha})^{B+1}$-weak and $\delta(\boldsymbol{\beta})^{B+1}$-weak black-box secret sharing schemes which satisfy the uniformity property when applied to a group G whose exponent is upper bounded by 2^B. It follows that sharing a secret simultaneously using the $\delta(\boldsymbol{\alpha})^{B+1}$-weak and the $\delta(\boldsymbol{\beta})^{B+1}$-weak black-box secret sharing scheme results in a black-box secret sharing scheme such that applied to a group G whose exponent is upper bounded by 2^B, it satisfies uniformity over $\delta(\boldsymbol{\alpha})^B \cdot G$ for the shares resulting from the $\delta(\boldsymbol{\alpha})^{B+1}$-weak and uniformity over $\delta(\boldsymbol{\beta})^B \cdot G$ for the shares resulting from the $\delta(\boldsymbol{\alpha})^{B+1}$-weak black-box secret sharing scheme.

4.7.3 Application: Threshold RSA

We show the potential usefulness of our black-box secret sharing scheme for threshold cryptography by indicating how it can be used to construct a secure threshold RSA signature (or encryption) scheme. In comparison with Shoup's RSA signature scheme [Sho00], it is less efficient (in terms of the share size and complexity of the signature generation), however it is less tailored to the RSA setting and may very well be applicable in a similar manner to new cryptographic schemes. Note that the secret sharing scheme used in [Sho00] to share the secret RSA exponent is *not* black-box in that correctness and privacy are only guaranteed under specific assumptions, which are satisfied in the considered RSA setting. Furthermore, our threshold RSA signature scheme allows an *arbitrary* public RSA exponent e, in contrast to Shoup's scheme which requires e to be a prime larger than n, the number of signing servers. Therefore, since signatures are typically much more often verified than generated, it might in some cases even be advantageous from a complexity point of view to use our approach which allows a small exponent, say $e = 3$.

The threshold RSA signature scheme based on our black-box secret sharing scheme is very similar to Shoup's scheme. In the following, we assume the reader to be familiar with the latter. The difference is that to share the secret exponent d and to reconstruct a signature from the signature shares, we use our black-box secret sharing scheme, with the above modifications in order to achieve uniformity. Note that restricting the RSA

modulus N to the product of two *strong* primes as in [Sho00], the exponent of $\mathbb{Z}_{\varphi(N)}$ and $\delta(\boldsymbol{\alpha})$ share the only common factor 2 and thus the exponent of $2 \cdot \mathbb{Z}_{\varphi(N)}$ and $\delta(\boldsymbol{\alpha})$ are co-prime, whereas the exponent of $\mathbb{Z}_{\varphi(N)}$ and $\delta(\boldsymbol{\beta})$ are co-prime per se. Hence, we do not have to incorporate the full construction from the previous section to achieve uniformity. Also note the following technical issue. The shares of the secret exponent d are elements of $\Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_{\varphi(N)}$ (rather than of $\mathbb{Z}_{\varphi(N)}$), where $\Lambda = \mathbb{Z}[X]/(f(X))$ is constructed according to the proof of Proposition 4.7. Therefore, we need to be able to compute a signature share "x^r" for $x \in \mathbb{Z}_N^*$ (the hash of a message to be signed) and $r \in \Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_{\varphi(N)}$ (a share). This is done using the fact that $x \in \mathbb{Z}_N^*$ can be identified with $1 \otimes x \in \Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_N^*$ and that $\Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_N^*$ is a module (not only over Λ but also) over $\Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_{\varphi(N)}$: the number-multiplication is given by "multiplication in the first and exponentiation in the second coordinate". Hence, writing $r = 1 \otimes r_0 + w \otimes r_1 + \cdots + w^{m-1} \otimes r_{m-1}$, where $w = \overline{X}$ (the residue class of X modulo $f(X)$) and $m = [\Lambda : \mathbb{Z}] = \deg f(X)$, "$x^r$" is given by

$$r \cdot (1 \otimes x) \;=\; 1 \otimes x^{r_0} + w \otimes x^{r_1} + \cdots + w^{m-1} \otimes x^{r_{m-1}} \;\in\; \Lambda \otimes_{\mathbb{Z}} \mathbb{Z}_N^*$$

Finally, in order to recognize incorrect signature shares, the signing servers can do random-oracle based non-interactive zero-knowledge proofs of the signature share's correctness, similar to [Sho00].

The security proof is also very similar to the proof given in [Sho00]. The shares of the dishonest signing servers can be simulated using the uniformity property, and, given a signature on a message, the complete set of signature shares can be computed using the share-completion property. Finally, the proofs of the correctness of the signature shares can be simulated by having control over the random oracle.

4.8 An Example Implementation

So far we were primarily interested in the asymptotically optimal result from Theorem 4.2. Several choices in its proof have been made to simplify the mathematical exposition, while suppressing computational aspects. There are a number of possible practical implementations of black-box secret sharing based on our result. In this section we show one possibility and analyze its complexity. We summarize the results in

Proposition 4.9 *There exists an implementation for the optimal black-box secret sharing scheme with the following complexity bounds. Computing the shares uses $O(n^3 \log^2 n)$ group operations and no bit operations, and the reconstruction of the secret uses $O(n^3 \log^3 n)$ group operations and $O(n^4 \log^5 n)$ bit operations.*

Let us compare these complexity bounds with the bounds proven in [DF94] for their scheme, which is not optimal in terms of the expansion factor.[6] Two possible implementa-

[6] For this comparison, we assume that $t = \Theta(n)$. Furthermore, note that in [DF94] the work of the reconstruction is divided up among the players (shareholders), and the complexity is given by the number of operations *per player*, while we understand by the complexity of the reconstruction the *total* number of operations, since dividing up the work of the reconstruction only makes sense in a restricted model where the players know in advance which shares are available.

tions are presented in [DF94]. The first one (Corollary 6.2) takes $O(n^5)$ group operations to compute the shares and $O(n^4)$ group operations to reconstruct the secret (and no bit operations at all). The second one (Corollary 6.3) takes $O(n^4 \log n)$ group and $O(n^7 \log^2 n)$ bit operations to compute the shares and $O(n^3 \log n)$ group and $O(n^5 \log^2 n)$ bit operations to reconstruct the secret.

Before we present the implementation that meets the claimed complexity bounds, we analyze below the size of the involved numbers and the complexity of some elementary operations.

For the rest of the section, let $\Lambda = \mathbb{Z}[X]/(f(X))$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)$, $\beta_0 = 0$, be as guaranteed by (the proof of) Proposition 4.7, and write $m = [\Lambda : \mathbb{Z}] = \lceil \log(n+1) \rceil$. Note that for a given parameter n, both $f(X)$ and $\boldsymbol{\beta}$ can be fixed once and for all.

4.8.1 The Bit Length of the Involved Numbers

Recall from Section 4.5 that any number $\lambda \in \Lambda$ can be represented by its coordinate vector $\vec{\lambda} = (\lambda_0, \ldots, \lambda_{m-1})^T \in \mathbb{Z}^m$ such that $\lambda = \lambda_0 \cdot 1 + \lambda_1 \cdot w + \cdots + \lambda_{m-1} \cdot w^{m-1}$, where $w = \overline{X}$ (the residue class of X modulo $f(X)$). Equivalently, λ can be represented by the integer polynomial $\lambda(X) = \lambda_0 \cdot 1 + \lambda_1 \cdot X + \cdots + \lambda_{m-1} \cdot X^{m-1}$, whose residue class modulo $f(X)$ equals λ. We call $\lambda(X)$ the *canonical (polynomial) representation* of λ. At some point it will be convenient to use a *non-canonical* (polynomial) representation, which is an integer polynomial of degree m or larger whose residue class modulo $f(X)$ equals λ.

Fix some off-the-shelf binary encoding schemes for integers and for integer polynomials. To binary encode a number $\lambda \in \Lambda$, encode *some* polynomial representation of λ.

Definition 4.14 *Let x be a variable representing either an integer, an integer polynomial, or a number from Λ. We write $\ell(x)$ for the length of the binary encoding of x, and we call it the* bit length *of x.*

The ambiguity of $\ell(\lambda)$ for $\lambda \in \Lambda$ caused by the ambiguity of the representation is resolved at each case by either specifying the considered representation or otherwise taking the canonical representation.

Lemma 4.9 *The polynomial $f(X) \in \mathbb{Z}[X]$ and the numbers $\beta_1, \ldots, \beta_n \in \Lambda$ can be chosen such that $\ell(f(X)) = O(n \log n)$ and $\ell(\beta_i) = O(n \log n)$ for $i = 1, \ldots, n$.*

Proof. By the constructive method from the proof of Theorem 4.2, $f(X) \in \mathbb{Z}[X]$ can be chosen such that each of its $\lceil \log(n+1) \rceil$ coefficients has bit length smaller than $\log(\prod_{p \in \Pi_n} p)$. By the Prime Number Theorem, this is $O(n)$. Thus the length of an encoding of $f(X)$, which is essentially the sum of the bit lengths of the coefficients, is $\ell(f(X)) = O(n \log n)$.

If $f(X)$ is chosen to be irreducible modulo every prime $p \in \Pi_n$, then a simple possible choice for the β_i's, respectively their canonical polynomial representations, is to select n

distinct, non-zero integer polynomials $\beta_i(X)$ of degree smaller than m, such that each of the coefficients is either 0 or 1. For instance, $\beta_i(X)$ can point to i by basing it on the binary representation of the integer i. In this case, $\ell(\beta_i) = O(\log n)$.

In general, the $\beta_i(X)$'s can be chosen as follows. For each $p \in \Pi_n$, choose n distinct non-zero polynomials $\beta_1^{(p)}(X), \ldots, \beta_n^{(p)}(X) \in \mathbb{F}_p[X]$ of degree smaller than r_p, where r_p is defined as in the proof of Proposition 4.7. This is possible by the choice of $f(X)$. Using the Chinese Remainder Theorem and an arbitrary lift, compute integer polynomials $\beta_1(X), \ldots, \beta_n(X)$ of degree smaller than $\deg(f)$ such that $\beta_i(X) \equiv \beta_i^{(p)}(X) \bmod p$, for $i = 1, \ldots, n$ and for all $p \in \Pi_n$. Similar to the encoding of the polynomial $f(X)$, this results in an encoding with bit length $\ell(\beta_i) = O(n \log n)$. □

Lemma 4.10 *Consider $\tilde{\beta} = \beta_i - \beta_j \in \Lambda$ for some $0 \leq j < i \leq n$. Then, there exist (degree-$O(\log n)$ representations of) $u, v \in \Lambda$ with $u \cdot \tilde{\beta} + v \cdot (n!)^2 = 1$ such that both $\ell(u)$ and $\ell(v)$ are upper bounded by $O(n \log^2 n)$. Furthermore, u and v can be computed using $O(n^2 \log^5 n)$ bit operations.*

Proof. Recall that $\tilde{\beta}$ and $(n!)^2$ are co-prime. It follows that there exist $u(X), w(X) \in \mathbb{Z}[X]$ of degree at most m such that

$$u(X) \cdot \tilde{\beta}(X) + w(X) \cdot f(X) \equiv 1 \pmod{(n!)^2},$$

where $\tilde{\beta}(X)$ is the canonical polynomial representation of $\tilde{\beta}$. They can be computed for instance by solving a system of linear equations modulo $(n!)^2$.[7] This requires $O(n^2 \log^5 n)$ bit operations, and the coefficients can be chosen to have bit length $O(n \log n)$. Note that $1 - u(X) \cdot \tilde{\beta}(X) - w(X) \cdot f(X)$ is a multiple of $(n!)^2$, and hence

$$v(X) = \frac{1}{(n!)^2} \cdot (1 - u(X) \cdot \tilde{\beta}(X) - w(X) \cdot f(X))$$

is an integer polynomial. Its degree is at most $2m$, and its coefficients have bit length $O(n \log n)$. By construction,

$$u(X) \cdot \tilde{\beta}(X) + v(X) \cdot (n!)^2 \equiv 1 \pmod{f(X)},$$

and $\ell(u(X))$ and $\ell(v(X))$ are upper bounded by $O(n \log^2 n)$. This proves the claim. □

4.8.2 The Complexity of the Number-Multiplication

Let G be an arbitrary finite Abelian group, and let $\Lambda \otimes_{\mathbb{Z}} G$ be its extension over Λ. Similar to the representation of numbers in Λ, an element $g \in \Lambda \otimes_{\mathbb{Z}} G$ can be represented by its coordinate-vector $\vec{g} = (g_0, \ldots, g_{m-1}) \in G^m$ such that $g = 1 \otimes g_0 + w \otimes g_1 + \cdots + w^{m-1} \otimes g_{m-1}$ for $w = \overline{X} \in \Lambda$.

[7] Solving a $l \times l$ linear equation system with entries modulo a k-bit integer using Gauss elimination requires $O(l^3 k^2)$ bit operations. Another possibility is to apply the extended Euclidean algorithm modulo all prime factors of $(n!)^2$, lift the solution to solutions modulo the prime *powers* and combine them to a solution modulo $(n!)^2$ using Chinese Remainder Theorem. This approach might be slightly more efficient.

Lemma 4.11 *Let μ be an integer. Let λ be a number from Λ, represented by a polynomial of degree $r = O(\log n)$. Multiplying μ with an element of G requires $O(\ell(\mu))$ group operations (i.e., addition and subtraction) and no bit operations. And multiplying λ with an element of $\Lambda \otimes_{\mathbb{Z}} G$ requires $O(\ell(\lambda) \log n + n \log^2 n)$ group operations in G and no bit operations.*

Proof. The first claim is obvious, using the standard double-and-add technique based on the binary representation of μ.

Concerning the second claim, write $f(X) = f_0 + \cdots + f_{m-1} \cdot X^{m-1} + X^m$. Note that for $g = 1 \otimes g_0 + w \otimes g_1 + \cdots + w^{m-1} \otimes g_{m-1} \in \Lambda \otimes_{\mathbb{Z}} G$

$$\begin{aligned} w \cdot g &= w \otimes g_0 + \cdots + w^m \otimes g_{m-1} = w \otimes g_0 + \cdots + w^m \otimes g_{m-1} - f(w) \cdot g_{m-1} \\ &= 1 \otimes (-f_0 \cdot g_{m-1}) + w \otimes (g_0 - f_1 \cdot g_{m-1}) + \cdots + w^{m-1} \otimes (g_{m-2} - f_{m-1} \cdot g_{m-1}) \end{aligned}$$

It follows that computing (the representation of) $\lambda \cdot g$ from (the representations of) λ and g requires $O(\ell(\lambda) \log n + rn \log n)$ group and no bit operations using Horner's rule based on λ's polynomial representation $\lambda(X) = \lambda_0 \cdot 1 + \lambda_1 \cdot X + \cdots + \lambda_r \cdot X^r$, i.e., computing $\lambda \cdot g$ as

$$\lambda \cdot g = \lambda_0 \cdot g + (\cdots (\lambda_{r-1} \cdot g + (\lambda_r \cdot g) \cdot w) \cdots) \cdot w$$

□

4.8.3 The Implementation and Its Analysis

Now we are equipped to present an implementation of the optimal black-box secret sharing scheme and show that it meets the complexity bounds claimed in Proposition 4.9. As basis, we use the scheme guaranteed by (the proof of) Theorem 4.2 in combination with the modifications given by Corollaries 4.5 and 4.7.

Let G be an arbitrary finite Abelian group, and write $\hat{G}$ for its extension $\Lambda \otimes_{\mathbb{Z}} G$ over Λ. The sharing phase is rather straightforward:

> SHARING PHASE
>
> To share a secret $s \in G$, choose $g_1, \ldots, g_t \in G$ and $\hat{g}_1, \ldots, \hat{g}_t \in \hat{G}$ at random, and compute
>
> $$s_i = n! \cdot s + i \cdot g_1 + \cdots + i^t \cdot g_t \in G$$
>
> and
>
> $$\hat{s}_i = \beta_1 \cdots \beta_n \cdot s + \beta_i \cdot \hat{g}_1 + \cdots + \beta_i^t \cdot \hat{g}_t \in \hat{G}$$
>
> for $i = 1, \ldots, n$, where in the latter $s \in G$ is identified with $1 \otimes s \in \hat{G}$.

By the observations in the previous section, using Horner's rule, and computing $\beta_1 \cdots \beta_n \cdot s$ (and similarly $n! \cdot s$) as $\beta_1 \cdot (\beta_2 \cdot (\cdots (\beta_n \cdot s) \cdots))$, this requires $O(n^3 \log^2 n)$ group operations

and no bit operations. In the remainder, we show how to reconstruct a shared secret with complexities as claimed in Proposition 4.9.

First, compute $(n!)^2 \cdot s \in G$ and $\Delta(\boldsymbol{\beta}) \cdot s \in \hat{G}$ from, say, the first $t+1$ shares. Computing the latter in the obvious way as

$$\begin{aligned}\Delta(\boldsymbol{\beta}) \cdot s &= \sum_{k=1}^{t+1} \left(\prod_{1 \le j < i \le n} (\beta_i - \beta_j) \cdot \prod_{\substack{l=1 \\ l \neq k}}^{t+1} \frac{\beta_l}{\beta_l - \beta_k} \cdot \hat{s}_k \right) \\ &= \pm \prod_{\substack{1 \le j < i \le n \\ i > t+1}} (\beta_i - \beta_j) \cdot \sum_{k=1}^{t+1} \left(\prod_{\substack{1 \le j < i \le t+1 \\ i,j \neq k}} (\beta_i - \beta_j) \cdot \prod_{\substack{l=1 \\ l \neq k}}^{t+1} \beta_l \cdot \hat{s}_k \right)\end{aligned}$$

requires $O(n^4 \log^2 n)$ group operations, assuming that $t = O(n)$. Namely, every $\hat{s}_k$ has to be multiplied with the $O(n^2)$ factors $\beta_i - \beta_j$ where $1 \le j < i \le t+1$ and $i, j \neq k$. Note that it is well defined (and straightforward to compute) in what cases the sign in the above expression changes, but for simplicity we do not make this explicit.

We show how to compute $\Delta(\boldsymbol{\beta}) \cdot s$ using $O(n^3 \log^3 n)$ group operations. For that, we define $\hat{r}_A \in \hat{G}$ for any set $A \subseteq \{1, \ldots, t+1\}$ by

$$\hat{r}_A = \sum_{k \in A} \left(\prod_{\substack{1 \le j < i \le t+1 \\ i,j \neq k \\ i \text{ or } j \text{ in } A}} (\beta_i - \beta_j) \cdot \prod_{\substack{l=1 \\ l \neq k}}^{t+1} \beta_l \cdot \hat{s}_k \right)$$

Note that $\hat{r}_{\{1,\ldots,t+1\}}$ coincides with the sum in the second expression for $\Delta(\boldsymbol{\beta}) \cdot s$ above. Furthermore, if $A = A' \cup A''$ for two disjoint A', A'', then $\hat{r}_A$ can be computed by multiplying $\hat{r}_{A'}$ with at most $2|A''|n$ and $\hat{r}_{A''}$ with at most $2|A'|n$ appropriate factors $\beta_i - \beta_j$ and adding the two results. It follows that $\hat{r}_{\{1,\ldots,t+1\}}$ can be computed inductively, starting with $\hat{r}_{\{1\}}, \ldots, \hat{r}_{\{t+1\}}$ and always combining pairs of $\hat{r}$'s. This requires $\lceil \log(t+1) \rceil$ steps. In the k-th step, there are at most $2(t+1)/2^{k-1}$ elements $\hat{r}_A$ that have to be multiplied with at most $2^k n$ factors $\beta_i - \beta_j$. In total this results in $O(n^3 \log^3 n)$ group operations. $\Delta(\boldsymbol{\beta}) \cdot s$ can then be computed from $\hat{r}_{\{1,\ldots,t+1\}}$ by another $O(n^3 \log^2 n)$ group operations.

Second, compute s from $(n!)^2 \cdot s$ and $\Delta(\boldsymbol{\beta}) \cdot s$ as follows by inductively removing all factors of $\Delta(\boldsymbol{\beta})$ from $\Delta(\boldsymbol{\beta}) \cdot s$. To simplify notation, write $l = n(n+1)/2$, and write $\tilde{\beta}_1, \ldots, \tilde{\beta}_l$ for an arbitrarily ordered sequence consisting of all differences $\beta_i - \beta_j \in \Lambda$ with $0 \le j < i \le n$, such that $\Delta(\boldsymbol{\beta}) = \prod \tilde{\beta}_i$. Also, write $s_{\boldsymbol{\alpha}} = (n!)^2 \cdot s \in G$, and write $\tilde{\beta}^{(k)} = \tilde{\beta}_1 \cdots \tilde{\beta}_k$ and $\hat{s}_{\boldsymbol{\beta}}^k = \tilde{\beta}^{(k)} \cdot s \in \hat{G}$ for $k = 0, \ldots, l$. Note that $\hat{s}_{\boldsymbol{\beta}}^0 = s$ and $\hat{s}_{\boldsymbol{\beta}}^l = \Delta(\boldsymbol{\beta}) \cdot s$. First, pre-compute the list $\tilde{\beta}^{(1)} \cdot s_{\boldsymbol{\alpha}}, \ldots, \tilde{\beta}^{(l-1)} \cdot s_{\boldsymbol{\alpha}}$. This requires $O(n^3 \log^2 n)$ group operations. Then, for k from l down to 0, compute $u_k, v_k \in \Lambda$ such that

$$u_k \cdot \tilde{\beta}_k + v_k \cdot (n!)^2 = 1$$

and compute $\hat{s}_{\boldsymbol{\beta}}^{k-1}$ inductively from $\hat{s}_{\boldsymbol{\beta}}^k$ as

$$\hat{s}_{\boldsymbol{\beta}}^{k-1} = u_k \cdot \hat{s}_{\boldsymbol{\beta}}^k + v_k \cdot (\tilde{\beta}^{(k-1)} \cdot s_{\boldsymbol{\alpha}})$$

Note that the right hand side equals $u_k \cdot \tilde{\beta}_k \cdot \tilde{\beta}^{(k-1)} \cdot s + v_j \cdot \tilde{\beta}^{(k-1)} \cdot (n!)^2 \cdot s$, which indeed coincides with $\tilde{\beta}^{(k-1)} \cdot s$ by the choice of u_k and v_k. All in all, by the observations in the previous section, this requires $O(n^3 \log^3 n)$ group operations and $O(n^4 \log^5 n)$ bit operations.

4.9 Concluding Remarks

4.9.1 Black-Box Secret Sharing for Non-Abelian Groups

We briefly address the problem of black-box secret sharing for *non*-Abelian groups. For a threshold access structure with threshold $t = n-1$, there exists a simple solution: to share $s \in G$ the dealer chooses the first $n-1$ shares $s_1, \ldots, s_{n-1}$ randomly from G and computes the last share s_n as $s_n = -(s_1 + \cdots + s_{n-1}) + s$. In combination with the technique of Ito *et al.* [ISN87], this allows to construct a black-box secret sharing scheme (for arbitrary finite groups) for *any* access structure. However, this construction leads to schemes that are inefficient—in general, but in particular in the important threshold-case: if t is linear in n then the expansion factor of the scheme is *exponential* in n.

Threshold black-box secret sharing schemes (for arbitrary groups) with *polynomial* expansion factor can be constructed by combining the above "sum-sharing" with the technique of Benaloh-Leichter [BL88] and the classical result of complexity theory that all monotone threshold functions are representable by poly-size monotone Boolean formulas. An alternative solution is given in [DF99].

4.9.2 Open Problems

One open problem is that of further improving the computational complexity of the sharing and the reconstruction phase, by finding a better implementation and/or by finding a way to construct a polynomial $f(X) \in \mathbb{Z}[X]$, as guaranteed by Proposition 4.7, with coefficients of size strictly less than $O(n)$, for instance of size $O(\sqrt{n})$. Alternatively, one might be able to prove a lower bound on the size of the coefficients of $f(X)$, showing that the constructive method from the proof of Theorem 4.2 is (asymptotically) optimal.

Recall that our black-box secret sharing scheme does not have full uniformity, though "some sort of uniformity" can be incorporated if an upper bound on the size of the group is known. It would definitely be nice to have an optimal black-box secret sharing scheme with full uniformity (as well as the share-completion property).

Finally, a line of research could be in the construction of *non-threshold* black-box secret sharing schemes. Very vaguely speaking, the main problem in trying to adopt the techniques used here to general-access-structure schemes is that the Δ-value, appropriately defined as smallest common multiple of sub-determinants of the span program matrix, seems in general to be exponential in its bit length. This even holds for access structures that are related to threshold access structures, like the multi-level access structure proposed in [Sim88] and studied in [Bri89].

Chapter 5

A Framework for Linear VSS and Distributed Commitments

5.1 Introduction

A man can't be too careful in the choice of his enemies. — Oscar Wilde

Secret sharing schemes, as studied in the previous two chapters, guarantee that the shared secret can be correctly reconstructed even if some of the n shares are *missing*. Its stronger version, *verifiable secret sharing* (VSS), introduced in [CGMA85], guarantees the correct reconstruction of the secret even if some shares are *incorrect*, and even if the dealer tries to distribute incorrect shares. Informally, the security of a VSS scheme requires that after the distribution of the shares, which may now be an interactive protocol among the dealer and the players, there exists a unique secret, the dealer's secret in case of an honest dealer, which can be reconstructed by the players, even if some players hand in incorrect shares. If the (efficient) reconstruction of the secret requires the cooperation of the dealer, then such a scheme is called a *distributed commitment* (DC) scheme.

VSS is a fundamental cryptographic primitive, and its applications range from secure data storage over threshold cryptography to multi-party computation (see e.g. Section 3.5.4). A DC achieves the same functionality as a cryptographic commitment (see e.g. Section 7.2.2): having shared a secret, the dealer is committed to it in that he can only refuse to help to reconstruct it, but he cannot achieve that a different secret is reconstructed. However, in contrast to its cryptographic counterpart, a distributed commitment can be *perfectly* secure, both with respect to privacy and correctness.

The goal of this chapter is a unified treatment of perfectly secure linear VSS and DC schemes. We present a very natural and general sharing protocol which converts an arbitrary given linear secret sharing scheme into a DC (or VSS) scheme, provided of course that this is possible at all, by enforcing *pairwise* consistency among the shares of the (honest) players (Section 5.3). Namely, by pairwise checking, complaining and accusing, it ensures that pairwise linear dependencies among the shares that should hold *do* hold. This seems to be not only a very natural but the only *possible* approach for the construction of per-

fectly secure (linear) DC and VSS schemes, and indeed, all known schemes can be seen as concrete instances of this general approach (e.g. [BGW88, CDM00, GIKR01, Mau03]).

Then we state the condition under which such a scheme is a secure DC (or VSS) scheme. This characterization is in terms of pure linear algebra, depending only on the parameters of the underlying secret sharing scheme and of the sharing protocol.

As a consequence, the security of all known schemes (and possibly even all future ones) follow as corollaries whose proofs are linear-algebra arguments, in contrast to some hybrid arguments used in the literature. This is demonstrated for two schemes, for the CDM DC scheme of [CDM00] and for the classical BGW VSS scheme of [BGW88]. We show how the security of the CDM DC scheme can be proven by a simple linear-algebra argument (Section 5.4). For the BGW VSS scheme, we show that some of the checks between players are superfluous, i.e., the scheme is not optimal (Section 5.5). This also shows that arguing about the security of such schemes becomes conceptually simpler.

Finally, our approach, establishing the minimal conditions for security, may lead to the design of ad-hoc VSS or DC schemes for general adversary structures which are more efficient than the schemes resulting from generic constructions as for instance that of [CDM00].

The material in this chapter is based on [FM02]. However, in order to synchronize with the other chapters, we do not consider a *mixed* adversary here (but merely mention it in Section 5.6.1), in contrast to [FM02]. On the other hand, we generalize the results of [FM02] by not restricting to schemes over fields.

5.2 Preliminaries

5.2.1 Model

Throughout this chapter, we assume a communication network among a dealer and n players $P_1, \ldots, P_n$ as described in Section 2.2. We assume a general (not necessarily threshold) adversary that is adaptive and rushing. Furthermore, he may be computationally unbounded, and we allow no non-zero failure probability. The security of the protocols will require the adversary to be Q^3.

5.2.2 Definition of VSS and DC

Let n be a positive integer, and let $\mathcal{A}$ be an adversary structure on $\{1, \ldots, n\}$. We recall the definition of a perfectly secure VSS and DC scheme.

Definition 5.1 *A* verifiable secret sharing *(VSS) scheme is a pair of protocols (phases), the* sharing phase*, where the dealer shares a secret s, and the* reconstruction phase*, where the players try to reconstruct s. It is called* perfectly $\mathcal{A}$-secure *if the following two properties hold (with probability one), even if a computationally unbounded adversary corrupts the players of a set $A \in \mathcal{A}$:*

- *Privacy: If the dealer remains honest, then the adversary gains no information about the secret s during the sharing phase.*
- *Correctness: After the secret is shared, there exists a unique value s' that can be reconstructed by the players, and, in case the dealer remains honest during the sharing phase, s' is equal to the shared secret s.*

If the (efficient) reconstruction of the secret requires the cooperation of the dealer, then such a scheme is called a perfectly $\mathcal{A}$-secure distributed commitment (DC) *scheme, and the sharing and reconstruction phases are also called* committing phase *and* opening phase, *respectively.*

In a DC scheme, a corrupted dealer can prevent the (efficient) reconstruction by refusing to cooperate correctly, but he cannot achieve that a different secret is reconstructed, not even with the help of the corrupted players. Note that if one would define a default value for the case the dealer refuses to reconstruct, then a corrupted dealer would not be committed because he could open a sharing in two different ways: as the real or as the default value.

We call the space from which the secret s is sampled the *domain* of the VSS (or DC) scheme. In case where it is a group G, the VSS (or DC) is called *linear* if from VSS (respectively DC) sharings of two secrets s and s', the players can *non-interactively* compute a sharing of $s + s'$. This implies in particularly that they can compute a sharing of $a + s$ for any publicly known $a \in G$. In case where the domain is a ring R, it is convenient to additionally required that the players can compute a sharing of $a \cdot s$ for any publicly known $a \in R$.

5.2.3 Notation

Let Λ be a commutative ring with 1. Let $\mathcal{M} = (\Lambda, M, \psi, n)$ be a linear secret sharing scheme over Λ for an access structure Γ on $\{1, \ldots, n\}$. For an arbitrary vector $\mathbf{x} \in \Lambda^d$, $\mathrm{supp}(\mathbf{x})$ denotes the set

$$\mathrm{supp}(\mathbf{x}) \;=\; \{i \mid \mathbf{x}_i \neq \mathbf{0}\} \;\subseteq\; \{1, \ldots, n\}$$

For $E \subseteq \Lambda^d$ and $A \subseteq \{1, \ldots, n\}$, $E|_A$ denotes the subset $E|_A = \{\mathbf{y} \in E \mid \mathrm{supp}(\mathbf{y}) \subseteq A\}$. As usual in linear algebra, for E as above, $\mathrm{span}(E) \subseteq \Lambda^d$ denotes the subspace consisting of all Λ-linear combinations of vectors in E, and we write $\mathbf{x} \perp E$ to denote that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ for all $\mathbf{y} \in E$, where $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the standard in-product of $\mathbf{x}$ and $\mathbf{y}$, and where $\mathbf{x}$ may be in Λ^d or in G^d for an arbitrary Λ-module G.

In this chapter, it will be convenient to change the notation and terminology introduced in Definition 3.3 (and 4.3) slightly. Namely, we define a *reconstruction vector* for a set $A \in \Gamma$ to be a vector $\boldsymbol{\lambda} \in \Lambda^d$ with $\mathrm{supp}(\boldsymbol{\lambda}) \subseteq A$ such that $\boldsymbol{\lambda}^T \mathbf{s} = s$ (rather then $\boldsymbol{\lambda} \in \Lambda^{d_A}$ such that $\boldsymbol{\lambda}^T \mathbf{s}_A = s$). Furthermore, we write $\boldsymbol{\lambda}^T \mathbf{s}$ as in-product $\langle \boldsymbol{\lambda}, \mathbf{s} \rangle$.

5.3 VSS and DC Based on Secret Sharing and Checking

For this section, let Λ be a commutative ring with 1, and let G be a finite Λ-module. Furthermore, let $\mathcal{M}$ be a linear secret sharing scheme over Λ for an access structure Γ, and let $\mathcal{A} = \bar{\Gamma}$ be the corresponding adversary structure. We write $\mathcal{H}$ for the collection of sets H of the form $H = \{1, \ldots, n\} \setminus A$ where $A \in \mathcal{A}$, i.e., $\mathcal{H}$ is the dual access $\mathcal{H} = \Gamma^*$. The intuition behind $\mathcal{H}$ is that if the adversary is restricted to corrupt the players of a set $A \in \mathcal{A}$ then the players of a set $H \in \mathcal{H}$ are guaranteed to remain honest.

5.3.1 Consistency

It is rather well known (at least for the threshold case) and straightforward to verify that if $\mathcal{A}$ is Q^3 (respectively $t < n/3$ in case $\mathcal{A} = \mathcal{A}_{t,n}$), then a sharing $\mathbf{s} \in G^d$ whose restriction $\mathbf{s}_H$ to some set $H \in \mathcal{H}$ is guaranteed to be a *correct* sharing of a secret $s \in G$ uniquely defines s, in the sense that quantifying over all $H' \in \mathcal{H}$, $\mathbf{s}_{H'}$ cannot be a correct sharing of a secret $s' \neq s$. All currently known VSS (and DC) schemes rely on this observation. We show below that in fact a seemingly weaker condition than *correctness* of the shares in H suffices for the uniqueness of s.

By definition, for any set $A \in \Gamma$ there exists at least one reconstruction vector $\boldsymbol{\lambda} \in \Lambda^d$ for A. Let $\mathcal{R} \subset \Lambda^d$ be an arbitrary set consisting of reconstruction vectors (for possibly different A's).

Definition 5.2 *We call* $\mathcal{R}$ complete *if for every* $A \in \Gamma$ *there exists a reconstruction vector* $\boldsymbol{\lambda}$ *for* A *that is contained in* $\mathcal{R}$.

Definition 5.3 *Let* $A \in \Gamma$ *and* $\mathbf{s} \in G^d$ *(not necessarily of the form* $\mathbf{s} = M\mathbf{b}$*). We call* $\mathbf{s}$ *a* consistent sharing *for* A *with respect to* $\mathcal{R}$ *if* $\langle \boldsymbol{\lambda}, \mathbf{s} \rangle = \langle \boldsymbol{\lambda}', \mathbf{s} \rangle$ *for all* $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_A$*, i.e., if*

$$\mathbf{s} \perp \{\boldsymbol{\lambda} - \boldsymbol{\lambda}' \mid \boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_A\}.$$

We also say that $\mathbf{s}$ *is a consistent sharing of* s *(for* A*), where* s *is given by* $s = \langle \boldsymbol{\lambda}, \mathbf{s} \rangle$ *for any* $\boldsymbol{\lambda} \in \mathcal{R}|_A$*, or that the* $\mathbf{s}_i$*'s for* $i \in A$*, are* consistent shares *of* s.

Remark 5.1 Obviously, correct shares are always consistent (with respect to any set $\mathcal{R}$ of reconstruction vectors). On the other hand, if Λ is a field, then it is easy to see that shares which are consistent with respect to the set of *all* reconstruction vectors must be correct.

Lemma 5.1 *Let* $\mathcal{R}$ *be a complete set of reconstruction vectors. The following holds if and only if* $\mathcal{A}$ *is* Q^3*. Quantifying over all* $H \in \mathcal{H}$*, an arbitrary vector* $\mathbf{s} \in G^d$ *is a consistent sharing for* H *(with respect to* $\mathcal{R}$*) of* at most one $s \in G$.

Remark 5.2 Defining consistency in an appropriate way, Lemma 5.1 also holds for non-linear secret sharing schemes, as shown in [FM02].

Proof. As can easily be seen, $\mathcal{A}$ is Q^3 if and only if the intersection $H \cap H'$ of any two sets $H, H' \in \mathcal{H}$ is contained in Γ. The latter, however, is equivalent to the uniqueness of s: Let $H, H' \in \mathcal{H}$. If $H \cap H' \in \Gamma$ then there exists a reconstruction vector $\boldsymbol{\lambda} \in \Lambda^d$ for $H \cap H'$, and hence if $\mathbf{s}$ is a consistent sharing of $s \in G$ for H and simultaneously of $s' \in G$ for H', then $s = \langle \boldsymbol{\lambda}, \mathbf{s} \rangle = s'$. On the other hand, if $H \cap H' \notin \Gamma$ then it follows from privacy that one can construct $\mathbf{s} \in G^d$ in such away that $\mathbf{s}_H$ is a correct sharing of some secret s while $\mathbf{s}_{H'}$ is a correct sharing of some secret $s' \neq s$. □

5.3.2 Pairwise Consistency

According to the above, if $\mathcal{A}$ is Q^3, and if by some means it is *checked*, as part of the sharing procedure, that the dealer distributes a *consistent* sharing $\mathbf{s}$ for the honest players (with respect to some complete set $\mathcal{R}$ of reconstruction vectors), then no matter what are the shares of the corrupted players, there is only one secret s that can be "explained" by $\mathbf{s}$. However, it seems to be impossible to *directly* check this kind of consistency, i.e. to verify whether $\langle \boldsymbol{\lambda}, \mathbf{s} \rangle = \langle \boldsymbol{\lambda}', \mathbf{s} \rangle$ for two different reconstruction vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$, *without violating privacy.* The only kind of consistency that can be checked without violating privacy is *pairwise consistency*, i.e. whether (some) pairwise Λ-linear dependences $\langle \boldsymbol{\gamma}, \mathbf{s} \rangle = 0$ with $\boldsymbol{\gamma} \in \Lambda^d$ and $\mathrm{supp}(\boldsymbol{\gamma}) = \{i, j\}$ that should hold indeed *do* hold; namely by comparing in private the respective contributions $\langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle$ and $\langle \boldsymbol{\gamma}_j, \mathbf{s}_j \rangle$ (which, up to the sign, are supposed to be equal) of the two involved players P_i and P_j. In case of an inconsistency, the players complain and the dispute is resolved with the help of the dealer, in such a way that finally the shares of all honest players are guaranteed to be pairwise consistent *and* privacy is not violated. This is described in full detail in the protocol in the next section. Below, a simple linear-algebra condition is given that is sufficient (and also necessary) in order for the pairwise consistency to imply consistency with respect to a given set of reconstruction vectors.

Consider the set

$$Checks(\mathcal{M}) \;=\; \{\boldsymbol{\gamma} \in \ker M^T \mid |\mathrm{supp}(\boldsymbol{\gamma})| \leq 2\} \subseteq \Lambda^d$$

of all possible *(pairwise) checking vectors*: for any *correct* sharing $\mathbf{s} = M\mathbf{b}$ of a secret $s \in G$, we have $\langle \boldsymbol{\gamma}, \mathbf{s} \rangle = \langle \boldsymbol{\gamma}, M\mathbf{b} \rangle = \langle M^T\boldsymbol{\gamma}, \mathbf{b} \rangle = \langle \mathbf{0}, \mathbf{b} \rangle = 0$ for every $\boldsymbol{\gamma} \in Checks(\mathcal{M})$. And let $\mathcal{C}$ be a subset of $Checks(\mathcal{M})$.

Definition 5.4 *Let $\mathbf{s} \in G^d$ (not necessarily of the form $\mathbf{s} = M\mathbf{b}$). We say that the share $\mathbf{s}_i$ is* pairwise consistent *with a share $\mathbf{s}_j$ (with respect to $\mathcal{C}$) if $\langle \boldsymbol{\gamma}, \mathbf{s} \rangle = 0$ for all $\boldsymbol{\gamma} \in \mathcal{C}|_{\{i,j\}}$. For $A \subseteq \{1, \ldots, n\}$, we call $\mathbf{s}$* pairwise consistent *for A (with respect to $\mathcal{C}$) if $\langle \boldsymbol{\gamma}, \mathbf{s} \rangle = 0$ for all $\boldsymbol{\gamma} \in \mathcal{C}|_A$, i.e., if*

$$\mathbf{s} \;\perp\; \mathcal{C}|_A\,.$$

The following lemma is now trivial.

Lemma 5.2 *Let $A \in \Gamma$. Then, a pairwise consistent sharing for A with respect to $\mathcal{C}$ is guaranteed to be a consistent sharing for A with respect to some given set $\mathcal{R}$ of reconstruction vectors if and only if*

$$\{\boldsymbol{\lambda} - \boldsymbol{\lambda}' \mid \boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_A\} \subseteq \operatorname{span}(\mathcal{C}|_A)\,.$$

5.3.3 Achieving Secure VSS and DC

For an arbitrary but fixed subset $\mathcal{C} \in Checks(\mathcal{M})$, the following sharing protocol enforces pairwise consistency with respect to the checking vectors $\boldsymbol{\gamma} \in \mathcal{C}$ among the players that remain honest during the execution, without revealing any information about the shared secret. It is similar to (and in fact generalizes) the sharing phase of the BGW VSS scheme [BGW88], though the concrete choice of the protocol is somewhat arbitrary, in that it can be modified in different ways without losing its functionality and without nullifying the outcoming results. For instance, techniques from [GIKR01] can be applied to improve the round complexity (at the cost of an increased communication complexity), and some linear secret sharing schemes $\mathcal{M}$ allow "early stopping".

SHARING (or COMMITTING) PHASE

1. To share (respectively commit to) a secret $s \in G$, the dealer chooses a random $\mathbf{b} = (b_1, b_2, \ldots, b_e) \in G^e$ such that $b_1 = s$, computes $\mathbf{s} = M\mathbf{b}$ and sends to every player P_i the corresponding share $\mathbf{s}_i$.
2. For every checking vector $\boldsymbol{\gamma} \in \mathcal{C}$, it is checked whether $\langle \boldsymbol{\gamma}, \mathbf{s} \rangle = 0$:
   If $\operatorname{supp}(\boldsymbol{\gamma}) = \{i\}$, then player P_i verifies whether $\langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle = 0$, and he broadcasts an "accusation" against the dealer if it does not hold.
   If $\operatorname{supp}(\boldsymbol{\gamma}) = \{i, j\}$ with $i < j$, then player P_i sends $c_{ij} = \langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle$ to P_j who verifies whether $c_{ij} + \langle \boldsymbol{\gamma}_j, \mathbf{s}_j \rangle = 0$ and broadcasts a "complaint" if it does not hold. The dealer answers such a complaint by broadcasting $c_{ij} = \langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle$, and if this value does not coincide with P_i's c_{ij} respectively if it does not fulfill $c_{ij} + \langle \boldsymbol{\gamma}_j, \mathbf{s}_j \rangle = 0$, then player P_i respectively P_j broadcasts an "accusation" against the dealer.
3. The following is repeated until there is no further "accusation" or the dealer is declared faulty (which requires at most n rounds). For every "accusation" from a player P_i, the dealer answers by broadcasting P_i's share $\mathbf{s}_i$, and P_i replaces his share by this $\mathbf{s}_i$. If this share contradicts the share of some player P_j, in the sense that $\langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle + \langle \boldsymbol{\gamma}_j, \mathbf{s}_j \rangle \neq 0$ for some $\boldsymbol{\gamma} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{\gamma}) = \{i, j\}$, then P_j broadcasts an "accusation" (if he has not yet done so in an earlier step). If this share $\mathbf{s}_i$ contradicts itself, in the sense that $\langle \boldsymbol{\gamma}_i, \mathbf{s}_i \rangle \neq 0$ for some $\boldsymbol{\gamma} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{\gamma}) = \{i\}$, or it contradicts a share $\mathbf{s}_j$ that has already been broadcast, then the dealer is publicly declared to be faulty.

Convention. In the above protocol, as well as in all other protocols in this work, we take it as understood that whenever a player expects to receive a message from another player, but no message arrives or it is not in the right format, he takes some fixed default value as the received message.

It is easy to see that if the dealer remains honest, then, on one hand, no matter what the corrupted players do, nobody learns anything beyond his share, and hence an adversary corrupting the players of a set $A \in \mathcal{A}$ learns nothing about the shared secret, and, on the other hand, the shares of the honest players $H = \{1, \ldots, n\} \setminus A$ form a correct sharing of s as computed by the dealer in step 1: $\mathbf{s}_H = M_H \mathbf{b}$. Furthermore, in case of a corrupted dealer, the protocol achieves pairwise consistency for H: $\mathbf{s} \perp \mathcal{C}|_H$. Note that this is well defined even though $\mathbf{s}_A$, the shares of the corrupted players, is not in general.

In combination with Lemma 5.1 and 5.2 this leads to

Theorem 5.1 *Let Λ be a commutative ring with 1. Let $\mathcal{M}$ be a secret sharing scheme over Λ for an adversary structure $\mathcal{A}$. Write $\Gamma = \bar{\mathcal{A}}$ and $\mathcal{H} = \Gamma^*$. Finally, let $\mathcal{C} \subseteq Checks(\mathcal{M})$ be a subset of all possible checking vectors. Then the considered sharing phase together with the opening phase below forms a perfectly $\mathcal{A}$-secure linear DC scheme with domain an arbitrary finite Λ-module if (and only if) $\mathcal{A}$ is Q^3 and if the condition*

$$\{\boldsymbol{\lambda} - \boldsymbol{\lambda}' \mid \boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_H\} \subseteq \operatorname{span}(\mathcal{C}|_H) \quad \text{for every } H \in \mathcal{H} \qquad (*)$$

holds for some complete set $\mathcal{R}$ of reconstruction vectors.

OPENING PHASE

1. The dealer broadcasts the vector $\mathbf{b} = (b_1, b_2, \ldots, b_e)^T$ used in the sharing phase, and every player P_i broadcasts the share $\mathbf{s}_i$ he agreed on in the sharing phase. Let A be the set of $i \in \{1, \ldots, n\}$ with $\mathbf{s}_i \neq M_i \mathbf{b}$.
2. If $A \in \mathcal{A}$ then the output is $s = b_1$. Else, the opening is rejected.

Proof of Theorem 5.1. Linearity follows from the linearity of the underlying secret sharing scheme. Privacy and correctness in case of an honest dealer follow straight from the above observations. Concerning correctness in case of a corrupted dealer, let $H \in \mathcal{H}$ collect the honest players. Furthermore, let $\mathbf{b} = (b_1, \ldots, b_e)^T \in G^e$ denote the vector broadcast by the dealer, and let $\mathbf{s} \in G^d$ consist of the shares $\mathbf{s}_i$ broadcast by the players, both during the opening phase. As mentioned above, it is guaranteed by the sharing protocol that $\mathbf{s}$ is pairwise consistent for H (with respect to $\mathcal{C}$). If condition $(*)$ is satisfied, then by Lemma 5.2 this implies that $\mathbf{s}$ is a consistent sharing of some secret $s \in G$ for H (with respect to $\mathcal{R}$). On the other hand, in case the opening phase is not rejected, there exists $H' \in \mathcal{H}$ such that $\mathbf{s}_{H'} = M_{H'} \mathbf{b}$. In particular, $\mathbf{s}$ is a consistent sharing of b_1 for H'. Lemma 5.1 implies that $b_1 = s$ is guaranteed if (and only if) $\mathcal{A}$ is Q^3. □

In order to have an efficient reconstruction procedure, we need an additional property:

Proposition 5.1 *If additionally*

$$\mathcal{C}|_{\{i,j\}} \subseteq \operatorname{span}(\mathcal{C}|_{\{i\}\cup A} \cup \mathcal{C}|_{\{j\}\cup A})$$

for all i, j and every $A \in \Gamma$, then the sharing phase together with the reconstruction phase below forms a perfectly $\mathcal{A}$-secure linear VSS scheme.

RECONSTRUCTION PHASE

1. Every player P_i broadcasts the share $\mathbf{s}_i$ he agreed on in the sharing phase. Such a share $\mathbf{s}_i$ is said to be "good" if it is pairwise consistent (with respect to the checking vectors in $\mathcal{C}$) with all shares $\mathbf{s}_j$ except maybe those belonging to a set $A \in \mathcal{A}$.
2. The secret s is reconstructed from the "good" shares using some suitable reconstruction vector from $\mathcal{R}$.

Proof of Proposition 5.1. It is straightforward to verify that correctness holds in case of an honest dealer. Privacy follows from the privacy of the DC scheme. Concerning correctness in case of a corrupted dealer, let $H \in \mathcal{H}$ collect the honest players. Furthermore, let $\mathbf{s} \in G^d$ consist of the shares $\mathbf{s}_i$ broadcast by the players during the reconstruction phase. Recall that $\mathbf{s}$ is pairwise consistent for H and hence the share of any honest player is "good". Let $\mathbf{s}_i$ be a "good" share, i.e., pairwise consistent with all shares belonging to some set $\tilde{H} \in \mathcal{H}$. For $A = H \cap \tilde{H}$ and an arbitrary $j \in H$, it follows that $\mathbf{s} \perp \mathcal{C}|_{\{i\}\cup A}$ and $\mathbf{s} \perp \mathcal{C}|_{\{j\}\cup A}$. If $\mathcal{A}$ is Q^3 then $A \in \Gamma$. Using the assumption in the claim, it follows that $\mathbf{s} \perp \mathcal{C}|_{\{i,j\}}$. As $j \in H$ is arbitrary, this means that $\mathbf{s}_i$ is pairwise consistent with *all* shares from honest players. Hence, $\mathbf{s}$ is pairwise consistent for $H \cup \{i\}$. Inductively, it follows that $\mathbf{s}$ is pairwise consistent for the set H' pointing to all "good" shares. Correctness follows now using similar arguments as in the proof of Theorem 5.1. □

The following lemma will be helpful in the next section.

Lemma 5.3 *Condition $(*)$ from Theorem 5.1 is fulfilled if every pair $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}$ satisfies*

$$\boldsymbol{\lambda} - \boldsymbol{\lambda}' \in \operatorname{span}(\mathcal{C}|_{\operatorname{supp}(\boldsymbol{\lambda}) \cup \operatorname{supp}(\boldsymbol{\lambda}')}) .$$

Proof. Let $H \in \mathcal{H}$ be arbitrary but fixed. Then, for $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_H \subseteq \mathcal{R}$, we have by assumption $\boldsymbol{\lambda} - \boldsymbol{\lambda}' \in \operatorname{span}(\mathcal{C}|_{\operatorname{supp}(\boldsymbol{\lambda}) \cup \operatorname{supp}(\boldsymbol{\lambda}')})$, which is contained in $\operatorname{span}(\mathcal{C}|_H)$. □

5.4 Application I: Proving the Security of the CDM Scheme

We now demonstrate the power of Theorem 5.1 and prove the security of the CDM DC scheme [CDM00] by proving a pure linear-algebra statement. We have to show that the condition $(*)$ from Theorem 5.1 holds, or, and that is what we are going to do, that the condition from Lemma 5.3 holds.

5.4.1 The CDM Scheme

In [CDM00], a generic construction was presented to convert any linear secret sharing scheme over a finite field into a (linear) DC scheme. It is straightforward to generalize this construction to *solid* secret sharing schemes over rings:

Let Λ be a commutative ring with 1. Let $\mathcal{M}_\circ = (\Lambda, M_\circ, \psi_\circ, n)$ be a solid secret sharing scheme over Λ for an access structure $\Gamma_\circ$ on $\{1, \ldots, n\}$. For simplicity, we assume that $d(\mathcal{M}_\circ) = n$ and $\psi_\circ(i) = i$ for all $i \in \{1, \ldots, n\}$. Write $e_\circ = e(\mathcal{M}_\circ)$ and $\boldsymbol{\varepsilon}_\circ = \boldsymbol{\varepsilon}(\mathcal{M}_\circ)$. The CDM DC scheme works as follows. Let G be an arbitrary finite Λ-module. To share (or commit to) a secret $s \in G$, the dealer chooses a random *symmetric matrix* $B \in G^{e_\circ \times e_\circ}$ with s in the upper left corner and sends the share $U_i = M_{\circ i} B \in G^{1 \times e_\circ}$ to player P_i. Now, every pair P_i, P_j of players verifies whether $M_{\circ i} \cdot U_j^T = M_{\circ j} \cdot U_i^T$ and, in case it does not hold, start complaining and accusing exactly as in the sharing protocol from the above section.

The rule how to compute the shares U_i defines a new solid secret sharing scheme $\mathcal{M} = (\Lambda, M, \psi, n)$. To fit into the formal Definition 3.3, the upper triangular part of the matrix B (which completely determines B) and the matrix $U = M_\circ B \in G^{n \times e_\circ}$ have to be written as vectors in $G^{e_\circ(e_\circ+1)/2}$ and $G^{ne_\circ}$, respectively, using some fixed ordering for the entries. The entries of M are then determined by the entries of $M_\circ$. Their positions as well as the labeling function depend on how exactly the matrices B and U are arranged as vectors. However, it will be convenient to keep this obviously equivalent "matrix view", for instance in showing that $\mathcal{M}$ is indeed a solid secret sharing scheme. Note that in this view, a reconstruction vector $\boldsymbol{\lambda}$ for some set A is seen as a matrix in $\Lambda^{n \times e_\circ}$, and the inproduct $\langle \boldsymbol{\lambda}, U \rangle$ is given by the sum of the pairwise products of all corresponding entries of $\boldsymbol{\lambda}$ and U.

Lemma 5.4 *$\mathcal{M}$ as above is a solid secret sharing scheme for the access structure $\Gamma = \Gamma_\circ$.*

Proof. By construction of $\mathcal{M}$, the first row of a sharing $U = M_\circ B \in \Lambda^{n \times e_\circ}$ of a secret $s \in \Lambda$ with respect to $\mathcal{M}$ is a sharing of the same secret with respect to $\mathcal{M}_\circ$. It follows correctness. In particular, if $\boldsymbol{\lambda}_\circ \in \Lambda^n$ is a reconstruction vector for a set $A \in \Gamma_\circ$ with respect to $\mathcal{M}_\circ$, then the matrix $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_\circ |\ 0\ ] \in \Lambda^{n \times e_\circ}$ is a reconstruction "vector" for A with respect to $\mathcal{M}$. For privacy, note that by assumption on $\mathcal{M}_\circ$, for every $A \notin \Gamma_\circ$ there exists $\boldsymbol{\kappa}(A) \in \Lambda^{e_\circ}$ such that $\mathbf{s} = M_\circ \boldsymbol{\kappa}(A)$ is a sharing of 1 with $\mathbf{s}_A = \mathbf{0}$. It follows that $U = M_\circ(\boldsymbol{\kappa}(A)\boldsymbol{\kappa}(A)^T)$ is (with respect to $\mathcal{M}$) a sharing of 1 with $U_A = 0$ (the zero matrix). $\square$

It can now easily be seen that the [CDM00] DC scheme is a concrete instance of the class of schemes described in the previous section. Indeed, it coincides with the sharing protocol from the previous section for $\mathcal{M}$ as above, and for $\mathcal{C}$ given by

$$\mathcal{C} \;=\; \{\boldsymbol{\gamma}^{ij} = \boldsymbol{\mu}^{ij} - \boldsymbol{\mu}^{ji} \mid 1 \le i < j \le n\}$$

where $\boldsymbol{\mu}^{ij} \in \Lambda^{n \times e_\circ}$ has $M_{\circ j}$ as i-th row and zero-entries otherwise, i.e.,

$$\boldsymbol{\mu}^{ij} \;=\; \begin{pmatrix} 0 \\ \boxed{M_{\circ j}} \\ 0 \end{pmatrix} \leftarrow i\text{-th row}$$

Note that $\langle \boldsymbol{\gamma}^{ij}, U\rangle = \langle \boldsymbol{\mu}^{ij}, U\rangle - \langle \boldsymbol{\mu}^{ji}, U\rangle = M_{\circ j}U_i^T - M_{\circ i}U_j^T = 0$ for $U = MB$, and hence $\mathcal{C}$ is a "valid" set of checking vectors. We stress once more that we describe and view the reconstruction vectors $\boldsymbol{\lambda}$ and the checking vectors $\boldsymbol{\gamma}^{ij}$ as matrices.

5.4.2 The Security Proof

As mentioned in the proof of Lemma 5.4, if $\boldsymbol{\lambda}_\circ \in \Lambda^n$ is a reconstruction vector for a set $A \in \Gamma_\circ$ with respect to the original scheme $\mathcal{M}_\circ$, then the matrix $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_\circ |\ 0\ ] \in \Lambda^{n \times e_\circ}$ is a reconstruction vector/matrix for A with respect to $\mathcal{M}$. Since $\Gamma = \Gamma_\circ$, the set $\mathcal{R}$ consisting of all such $\boldsymbol{\lambda}$'s is complete. Furthermore, we will show the following linear-algebraic fact.

Lemma 5.5 *For every pair $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}$,*

$$\boldsymbol{\lambda} - \boldsymbol{\lambda}' \;\in\; \mathrm{span}(\mathcal{C}|_{\mathrm{supp}(\boldsymbol{\lambda}) \cup \mathrm{supp}(\boldsymbol{\lambda}')})\,.$$

The following corollary now follows directly from Theorem 5.1 and Lemma 5.3.

Corollary 5.1 *The CDM DC scheme based on a solid secret sharing scheme with adversary structure structure $\mathcal{A}$ is $\mathcal{A}$-secure if and only if $\mathcal{A}$ is Q^3.*

Proof of Lemma 5.5. Let $\boldsymbol{\lambda}_\circ = (\lambda_{\circ 1}, \ldots, \lambda_{\circ n})^T$ and $\boldsymbol{\lambda}'_\circ = (\lambda'_{\circ 1}, \ldots, \lambda'_{\circ n})^T$ be reconstruction vectors with respect to $\mathcal{M}_\circ$ (for possibly two different sets from $\Gamma_\circ$), such that $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_\circ |\ 0\ ]$ and $\boldsymbol{\lambda}' = [\boldsymbol{\lambda}'_\circ |\ 0\ ]$, both in $\Lambda^{n \times e_\circ}$, are reconstruction vectors from $\mathcal{R}$. We have $\sum_j \lambda'_{\circ j} M_{\circ j} = \boldsymbol{\lambda}'^T_\circ \cdot M_\circ = \boldsymbol{\varepsilon}_\circ{}^T = (1, 0, \ldots, 0)$ and hence

$$\sum_j \lambda'_{\circ j} \boldsymbol{\mu}^{ij} \;=\; \sum_j \lambda'_{\circ j} \begin{pmatrix} 0 \\ \boxed{M_{\circ j}} \\ 0 \end{pmatrix} \;=\; \begin{pmatrix} 0 \\ \boxed{1\,0 \cdots 0} \\ 0 \end{pmatrix}$$

where the non-zero row is at the i-th position, and hence $\boldsymbol{\lambda}$ can be written as

$$\boldsymbol{\lambda} \;=\; [\boldsymbol{\lambda}_\circ |\, 0\,] \;=\; \sum_i \lambda_{\circ i} \begin{pmatrix} 0 \\ \boxed{1\,0\cdots 0} \\ 0 \end{pmatrix} \;=\; \sum_i \lambda_{\circ i} \left(\sum_j \lambda'_{\circ j} \boldsymbol{\mu}^{ij} \right) \;=\; \sum_{ij} \lambda_{\circ i} \lambda'_{\circ j} \boldsymbol{\mu}^{ij} .$$

Similarly $\boldsymbol{\lambda}' = \sum_{ij} \lambda_{\circ i} \lambda'_{\circ j} \boldsymbol{\mu}^{ji}$ and therefore

$$\boldsymbol{\lambda} - \boldsymbol{\lambda}' \;=\; \sum_{ij} \lambda_{\circ i} \lambda'_{\circ j} \left(\boldsymbol{\mu}^{ij} - \boldsymbol{\mu}^{ji} \right) \;=\; \sum_{ij} \lambda_{\circ i} \lambda'_{\circ j} \boldsymbol{\gamma}^{ij} \;\in\; \mathrm{span}(\mathcal{C}|_{\mathrm{supp}(\lambda) \cup \mathrm{supp}(\lambda')}) ,$$

which proves the claim. □

5.5 Application II: Reducing the Number of Checks in the BGW Scheme

Theorem 5.1 tells us that as long as the set $\{\boldsymbol{\lambda} - \boldsymbol{\lambda}' | \boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{R}|_H\}$ is contained in $\mathrm{span}(\mathcal{C}|_H)$, where $H \in \mathcal{H}$ collects the honest players, the corresponding scheme is secure. By this it is obvious that if the vectors in $\mathcal{C}|_H$ are not linearly independent, then $\mathcal{C}|_H$ contains more vectors than actually needed. We will now use this simple observation to reduce the number of checks in the (symmetric version of the) BGW VSS scheme [BGW88].

For simplicity, we restrict to the classical case where Λ is a *finite field*. However, in combination with the threshold span program constructions from Section 4.4.2, the results of this section can be generalized to commutative rings with 1.

The variation of the scheme of [BGW88] where a *symmetric* bivariate polynomial is used instead of an arbitrary one can be seen as a special case of the CDM scheme, where the matrix $M_\circ$ is a Vandermonde matrix, i.e., $M_{\circ i} = (1, \omega_i, \omega_i^2, \ldots, \omega_i^t)$ for disjoint and non-zero $\omega_1, \ldots, \omega_n \in \Lambda$. We have the following fact.

Lemma 5.6 *Let $B \in \Gamma_{t,n} = \{A \subseteq \mathcal{P} \mid |A| > t\}$ and $H \supseteq B$. Then*

$$\mathrm{span}(\{\boldsymbol{\gamma}^{ij} \in \mathcal{C}|_H \mid i \in B \text{ or } j \in B\}) \;=\; \mathrm{span}(\mathcal{C}|_H) .$$

Similarly, it can be shown using linear algebra that $\mathcal{C}_{\{i,j\}} \subseteq \mathrm{span}(\mathcal{C}|_{\{i\} \cup A} \cup \mathcal{C}|_{\{j\} \cup A})$ for all i, j and every A with $|A| \geq t+1$. The following corollary follows now from Theorem 5.1, showing that (the symmetric version of) the classical VSS scheme of [BGW88] is not optimal with respect to the number of required pairwise checks.

Corollary 5.2 *The symmetric version of the BGW VSS scheme is $\mathcal{A}_{t,n}$-secure if and only if $n > 3t$, even if $\mathcal{C}$ is replaced by*

$$\bar{\mathcal{C}} \;=\; \{\boldsymbol{\gamma}^{ij} \in \mathcal{C} \mid j > t\} .$$

Proof. Let H be the set of honest players, hence $|H| \geq n - t > 2t$. Clearly, the set $B = \{i \in H \mid i > t\}$ is in Γ and hence $\bar{\mathcal{C}}|_H = \{\boldsymbol{\gamma}^{ij} \in \mathcal{C}|_H \mid i \in B \text{ or } j \in B\}$ fulfills $\text{span}(\bar{\mathcal{C}}|_H) = \text{span}(\mathcal{C}|_H)$. □

Alternatively, this shows that the classical BGW VSS scheme allows "early stopping", as it is also used in the 4-round VSS of [GIKR01].

The proof of Lemma 5.6 is purely technical and does not give any new insight. Nevertheless, for completeness, it is given below. Recall that the checking vectors $\boldsymbol{\gamma}^{ij} \in \mathcal{C}$ (which are actually matrices) are of the following form: The i-th row is $M_{\circ j}$, the j-th row is $-M_{\circ i}$, and all remaining entries are zero, i.e., $\boldsymbol{\gamma}_i^{ij} = M_{\circ j}$ and $\boldsymbol{\gamma}_j^{ij} = -M_{\circ i}$ and $\boldsymbol{\gamma}_l^{ij} = 0$ for $l \neq i, j$.

Proof of Lemma 5.6. We assume without loss of generality that $B = \{n - t, \ldots, n\}$. Consider an arbitrary but fixed checking vector $\boldsymbol{\gamma}^{i_0 j_0}$ with $i_0, j_0 \in H$ and $i_0 < j_0 < n - t$, i.e. $i_0, j_0 \notin B$ (otherwise, nothing needs to be shown). We have to show that $\boldsymbol{\gamma}^{i_0 j_0}$ is contained in $\text{span}(\{\boldsymbol{\gamma}^{ij} \in C|_H \mid i \in B \text{ or } j \in B\}$. This will be done by the following

Claim: For $1 \leq i \leq n$ and $n - t \leq l \leq n$ let $\nu_l^i \neq 0$ be such that $\sum_{l=n-t}^{n} \nu_l^i M_{\circ l} = M_{\circ i}$.[1] Then, there exists a sequence $\boldsymbol{\delta}^{n-t-1}, \ldots, \boldsymbol{\delta}^n \in \text{span}(\{\boldsymbol{\gamma}^{ij} \in C|_H \mid i \in B \text{ or } j \in B\})$ such that for every $n - t - 1 \leq i \leq n$

$$\boldsymbol{\delta}_k^i = \begin{cases} \boldsymbol{\gamma}_k^{i_0 j_0} & \text{if } k \leq i \\ \sum_{k \neq l = i+1}^{n} (\nu_k^{i_0} \nu_l^{j_0} - \nu_l^{i_0} \nu_k^{j_0}) M_{\circ l} & \text{otherwise} \end{cases}$$

Truly, we can set

$$\boldsymbol{\delta}^{n-t-1} = \sum_{l=n-t}^{n} (\nu_l^{j_0} \boldsymbol{\gamma}^{i_0 l} - \nu_l^{i_0} \boldsymbol{\gamma}^{j_0 l}) \in \text{span}(\{\boldsymbol{\gamma}^{ij} \in C|_H \mid i \in Q^* \text{ or } j \in Q^*\})$$

Then for $k \leq n - t - 1$ we indeed have $\boldsymbol{\delta}_k^{n-t-1} = \boldsymbol{\gamma}_k^{i_0 j_0}$, namely

$$\boldsymbol{\delta}_{i_0}^{n-t-1} = \sum_l \nu_l^{j_0} M_{\circ l} = M_{\circ j_0} = \boldsymbol{\gamma}_{i_0}^{i_0 j_0}$$

$$\boldsymbol{\delta}_{j_0}^{n-t-1} = -\sum_l \nu_l^{i_0} M_{\circ l} = -M_{\circ i_0} = \boldsymbol{\gamma}_{j_0}^{i_0 j_0} \quad \text{and}$$

$$\boldsymbol{\delta}_k^{n-t-1} = 0 = \boldsymbol{\gamma}_k^{i_0 j_0} \quad \text{if } k \neq i_0, j_0$$

while for $k > n - t - 1$ we have

$$\begin{aligned} \boldsymbol{\delta}_k^{n-t-1} &= -\nu_k^{j_0} M_{\circ i_0} + \nu_k^{i_0} M_{\circ j_0} = -\nu_k^{j_0} \Big(\sum_{l=n-t}^{n} \nu_l^{i_0} M_{\circ l} \Big) + \nu_k^{i_0} \Big(\sum_{l=n-t}^{n} \nu_l^{j_0} M_{\circ l} \Big) \\ &= \sum_{l=n-t}^{n} (\nu_k^{i_0} \nu_l^{j_0} - \nu_l^{i_0} \nu_k^{j_0}) M_{\circ l} = \sum_{\substack{l=n-t \\ l \neq k}}^{n} (\nu_k^{i_0} \nu_l^{j_0} - \nu_l^{i_0} \nu_k^{j_0}) M_{\circ l} \end{aligned}$$

[1] This is well known to exist.

And inductively for $i = n-t-1, \ldots, n-1$, given $\boldsymbol{\delta}^i$ as demanded, we can set

$$\boldsymbol{\delta}^{i+1} \;=\; \boldsymbol{\delta}^i - \sum_{l=i+2}^{n} (\nu_{i+1}^{i_0}\nu_l^{j_0} - \nu_l^{i_0}\nu_{i+1}^{j_0})\boldsymbol{\gamma}^{i+1,l} \;\in\; \mathrm{span}(\{\boldsymbol{\gamma}^{ij} \in C|_H \mid i \in Q^* \text{ or } j \in Q^*\})$$

Then, clearly, for $k < i+1$ we have $\boldsymbol{\delta}_k^{i+1} = \boldsymbol{\delta}_k^i = \boldsymbol{\gamma}_k^{i_0 j_0}$. Furthermore, we have

$$\boldsymbol{\delta}_{i+1}^{i+1} \;=\; \boldsymbol{\delta}_{i+1}^i - \sum_{l=i+2}^{n} (\nu_{i+1}^{i_0}\nu_l^{j_0} - \nu_l^{i_0}\nu_{i+1}^{j_0})M_{\circ l} \;=\; 0 \;=\; \boldsymbol{\gamma}_{i+1}^{i_0 j_0}$$

while for $k > i+1$

$$\boldsymbol{\delta}_k^{i+1} \;=\; \boldsymbol{\delta}_k^i - (\nu_{i+1}^{i_0}\nu_k^{j_0} - \nu_k^{i_0}\nu_{i+1}^{j_0})M_{\circ i+1} \;=\; \sum_{\substack{l=i+2 \\ l\neq k}}^{n} (\nu_k^{i_0}\nu_l^{j_0} - \nu_l^{i_0}\nu_k^{j_0})M_{\circ l}$$

as required. □

5.6 Concluding Remarks

5.6.1 Mixed Adversaries

In [FM02] we considered a more general adversary, a so called *mixed* adversary, introduced in [FHM98, FHM99]. A mixed adversary can make two types of corruptions: *active* and *passive* ones. An *active* corruption is an ordinary corruption as considered in this work, and as defined in Section 2.3. A *passive* corruption is a weaker kind of corruption, in that the adversary can "only" see all private information of a corrupted player, but the corrupted player continues to correctly execute his program. In other words, a passive corruption allows the adversary to eavesdrop a player. A mixed adversary is characterized by two adversary structures: $\mathcal{A}$ and Δ such that $\mathcal{A} \subseteq \Delta$, with the intended meaning that the adversary can corrupt the players of a set $D \in \Delta$, of which he may corrupt a subset $A \subseteq D$ with $A \in \mathcal{A}$ actively while the remaining players in D he may corrupt passively. In the context of a mixed adversary, the following notation is convenient. For adversary structures $\mathcal{A}$ and Δ, we define a new adversary structure $\mathcal{A} \sqcup \Delta$ by

$$\mathcal{A} \sqcup \Delta \;=\; \{A \cup D \mid A \in \mathcal{A}, D \in \Delta\}.$$

On the other hand, the results in [FM02] are restricted to the case where Λ is a finite field. Combining both leads in particular to the following generalizations of Theorem 5.1 and Corollary 5.1.

Generalized Theorem 5.1 *Let Λ be a commutative ring with 1. Let $\mathcal{M}$ be a secret sharing scheme over Λ for an adversary structure Δ. Finally, let $\mathcal{C} \subseteq Checks(\mathcal{M})$ be*

a subset of all possible checking vectors. Then the considered DC scheme is a perfectly $(\mathcal{A}, \Delta)$*-secure DC scheme for an adversary structure* $\mathcal{A} \subseteq \Delta$ *if (and only if)*

$$\{1, \ldots, n\} \notin \mathcal{A} \sqcup \mathcal{A} \sqcup \Delta$$

and if condition $(*)$ *of Theorem 5.1 holds for some complete set* $\mathcal{R}$ *of reconstruction vectors.*

To our knowledge, the condition $\{1, \ldots, n\} \notin \mathcal{A} \sqcup \mathcal{A} \sqcup \Delta$ for VSS to be possible, which generalizes the Q^3 condition, has not been stated previously in the literature, although the condition for secure MPC computation has been given in [FHM99].

Generalized Corollary 5.1 *The CDM DC scheme based on a solid secret sharing scheme with adversary structure structure* Δ *is* $(\mathcal{A}, \Delta)$*-secure for an adversary structure* $\mathcal{A} \subseteq \Delta$ *if and only if* $\{1, \ldots, n\} \notin \mathcal{A} \sqcup \mathcal{A} \sqcup \Delta$.

In combination with the MPC techniques from [CDM00] (respectively Section 3.5) this allows for efficient MPC protocols, secure against a mixed adversary, with respect to a much broader class of pairs $(\mathcal{A}, \Delta)$ in comparison to [FHM99]. Note that the complexity of the protocol from [FHM99] is polynomial in the number of subsets contained in $\mathcal{A}$ and Δ, while our approach leads to a complexity that is polynomial in the span program size of Δ.

5.6.2 Open Problems

The main open problem of this chapter is to find linear secret sharing schemes $\mathcal{M}$ for Q^3 adversary structures (possibly even for a small n) which

1. have enough redundancy among the shares, such that there exists $\mathcal{C} \subseteq Checks(\mathcal{M})$ which satisfies condition $(*)$ from Theorem 5.1 for some complete set $\mathcal{R}$ of reconstruction vectors, and which
2. do not come from generic constructions as that of [CDM00] and therefore have the potential of being more efficient.

Chapter 6

On the Cost of Reconstructing a Shared Secret

6.1 Introduction

One of the secrets of life is to make stepping stones out of stumbling blocks.

— Jack Penn

In this chapter, we consider VSS schemes that are secure with respect to a threshold adversary structure $\mathcal{A}_{t,n}$. First, we consider a relaxed version where the dealer is guaranteed to be honest, which we call *honest-dealer VSS*, and we ask the following question. How much information must be sent in order to reconstruct a shared secret? This question is interesting only if $n/3 \leq t < n/2$ and if the adversary is computationally unbounded, since otherwise the problem is either "too hard" or "too easy": if $t \geq n/2$ the problem clearly cannot be solved, and if $t < n/3$ or computational security suffices, standard methods ([BGW88] and [Ped91]) immediately give optimal solution.

Somewhat surprisingly, nothing seems to be known for $n/3 \leq t < n/2$, besides the upper bound that follows from the "secret-sharing-when-the-dealer-is-a-knight" protocol from Rabin and Ben-Or [RB89]. It is easy to see that for t in this range, one cannot construct a scheme where the correct secret is *always* reconstructed. At best one can make a scheme where every honest player outputs the correct secret or `failure`, where the latter happens with probability only $2^{-\Omega(k)}$, where k is a security parameter. For schemes that achieve this and where the reconstruction is completed in a single round, we show a lower bound of $\Omega(kn^2/(n-2t))$ bits on the amount of information sent in the reconstruction, for $(1+\epsilon) \cdot n/3 \leq t < n/2$ and any $\epsilon > 0$, i.e. essentially all of the range that was previously open (Section 6.4). This may be seen as an answer to the question "what does it cost to get the best possible security in a minimal number of rounds?". No such bound was known previously, and it holds even for schemes that are not efficient (with respect to communication and/or private computation), and trivially applies also to ordinary VSS (we cannot expect to do better in a more adversarial situation). Finally, it meets the upper bound from the [RB89] protocol for and only for t (close to) maximal, i.e., $t = \lfloor (n-1)/2 \rfloor$.

For an honest dealer, we use known results on authentication codes to show that the lower bound is tight up to a constant factor for *all* t in the considered range (even if we count the total information sent).

This scheme establishing the upper bound is computationally efficient and can—at least in principle—be turned into a VSS, since the honest dealer could always be replaced by a secure multi-party computation using generic methods (e.g. [RB89, CDD+99]). This, however, is not a satisfactory solution: while reconstruction would be the same complexity as before, the sharing phase would become extremely inefficient in comparison. In particular, since multi-party computation uses VSS as a subroutine, the complexity of the resulting sharing phase is condemned to be a magnitude larger than that of the best VSS.

To close this gap, we present a new VSS protocol where the complexity of the sharing phase matches that of the previously best known VSS for our scenario [CDD+99], but where the reconstruction meets our lower bound (Section 6.5). This beats previous VSS protocols by a factor of at least n.

We show an application of this to multi-party computation with *pre-processing*, introduced in [Bea91], where the n players want to compute a function on private inputs (Section 6.6). In order to do this more efficiently than starting from scratch, the players are allowed to a pre-processing phase and store some information obtained in this phase *before* the function and the inputs become known. The computation phase of our protocol has communication complexity $O(n^2 k|\mathcal{C}|/(n-2t))$, where $|\mathcal{C}|$ denotes the number of multiplication gates in the circuit to be computed. This improves the computation phase of earlier similar protocols [RB89, CDD+99] by a factor of at least n without increasing the complexity of the pre-processing.

We conclude the chapter as follows (Section 6.7): Using methods from [CDM00], we show how to generalize our schemes to provide security against a general Q^2 adversary, improving known results by a factor of at least n. And we look at the case where the reconstruction is allowed to use more than one round of interaction and observe, using results from [CPS99], that the amount of information sent by the honest dealer can be brought down to $n(n+k)$ bits, at the expense of a significantly more inefficient reconstruction phase.

The material in this chapter is based on [CDF01].

6.2 Model

Throughout this chapter, we consider a communication network among a dealer and n players $P_1, \ldots, P_n$ as described in Section 2.2 *with* a broadcast channel.[1] The adversary is considered to be computationally unbounded. We allow some small failure probability, although we try to minimize the type of failure. We restrict to a threshold adversary with

[1] Broadcast is necessary for VSS. Indeed, VSS obviously implies broadcast. However, it is well known that broadcast cannot be (unconditionally secure) implemented using bilateral channels only in case of $t \geq n/3$ corrupted players.

threshold $t < n/2$, though some results carry over to a general Q^2 adversary (see Section 6.7.1). In order to put some light on the role of adaptiveness and rushing, we consider static and adaptive as well as non-rushing and rushing adversaries. However, all proposed protocols are secure against an adaptive and rushing adversary.

6.3 Single-Round Honest-Dealer VSS

6.3.1 Definition

We first model the general communication pattern for VSS schemes where the dealer is guaranteed to be honest and whose reconstruction phase consists of a single round of communication. We will call such a scheme *single-round honest-dealer VSS*. Our main point of interest is the communication complexity of the reconstruction phase of such a scheme. Consider schemes of the following general form, and assume an adversary who corrupts up to t of the n players P_i, *but not the dealer.*

> SHARING PHASE
>
> Given a secret s, the honest dealer generates shares $s_i = (\kappa_i, y_i)$, $i = 1 \ldots n$. Privately he sends s_i to player P_i.

> RECONSTRUCTION PHASE
>
> Each player P_i is required to broadcast y_i. Locally and by some fixed (possibly probabilistic) method, each player P_i decides on the secret s based on his private κ_i and on the broadcast $y_1, \ldots, y_n$, i.e., either outputs a value, hopefully equal to s, or `failure`.

It is not difficult to see that in fact we may always and without loss of generality assume our schemes of interest to be of this form. Indeed, by the assumption that the dealer is honest, we may assume without loss of generality that the sharing phase only consists of the dealer sending private information s_i to each of the players P_i, i.e., any secure distributed computation carried out by the players in the sharing phase could as well be carried out by the honest dealer. Similarly, we may assume that in the reconstruction phase each player P_i merely broadcasts a piece of information, y_i, that only depends on the private information s_i received from the dealer. Namely, at the cost of at most a constant factor of increased communication, private channels can be simulated by one-time pads, the keys of which are distributed by the honest dealer. In fact, it can be assumed that in general $s_i = (\kappa_i, y_i)$, where y_i is required to be broadcast in the reconstruction phase, and each player P_i makes a local (possibly probabilistic) decision on the secret s based on the broadcast information and his private κ_i. We will call κ_i the *key* and y_i the *public share* of player P_i.

For each of the at most t corrupted players P_j, the adversary can broadcast a manipulated y_j, which may depend arbitrarily on the private information $s_j = (\kappa_j, y_j)$ of those corrupted players, or broadcast nothing at all in some cases. Note though that for at least $n - t$ of the y_i's are correct. If additionally the adversary is rushing, he can choose to "speak last" in the reconstruction phase. This means that in principle any corrupted shares may additionally depend on the information broadcast by the honest players, in particular they may depend on the secret s. By contrast, a non-rushing adversary is one who selects the corrupted shares before the start of the reconstruction phase. Note that security against non-rushing adversaries makes sense in a communication model enhanced with a "simultaneous broadcast channel", i.e., one by means of which all players broadcast their information at the same time.

We define our notion of security. Assume a computationally unbounded adversary that corrupts at most t of the n players but not the dealer. Additionally, the adversary can be static or adaptive, and rushing or non-rushing.

Definition 6.1 *A single-round honest-dealer VSS scheme is called* $(t, n, 1 - \delta)$-secure *if the following holds.*

- *Privacy: The adversary gains no information about the shared secret s as a result of the sharing phase.*
- *$(1 - \delta)$-Correctness: In the reconstruction phase, each honest player outputs either the correct secret s or* `failure`*, where for every player the latter happens with probability at most $\delta < 1$, independent of s.*

As mentioned earlier, the case $t \geq n/2$ is not interesting and the case $t < n/3$ is completely understood: zero failure probability and optimally efficient communication can be achieved by a combination of Shamir's secret sharing scheme and standard efficient error correction techniques [BW86, BGW88].

We stress that our definition of security captures the best one can achieve in this setting. Negligible error δ^m is achieved by m parallel repetitions. More importantly, it only differs from perfect security in the sense that there is a (small) probability that some player does not reconstruct the secret and outputs `failure` instead. This is unavoidable in the presence of a (not necessarily rushing) active adversary, as we show in Lemma 6.1 below. Furthermore, existing honest-dealer VSS schemes like [RB89] ("secret sharing when the dealer is a knight") fulfill our security definition without any changes in the required communication.

A seemingly stronger security definition would require agreement among the honest players in all cases, i.e., they all recover the correct secret or they all output `failure`, where the latter would happen with probability at most δ. However, this is impossible to achieve in a single round reconstruction phase with a *rushing adversary*, as we show in Lemma 6.2 below.[2]

[2] In Section 6.7.2, it is argued that agreement is possible in the presence of a non-rushing adversary. Agreement can be achieved in all cases by adding one extra round of communication.

Note also that the reconstruction procedure in our definition is completely general in that it does not dictate how the correct secret is recovered by the honest players. The definition merely states that from all broadcast and from his private information, an honest player can reconstruct the secret. In particular, in our definition it need not be the case that an honest player, using his private information, "filters out" false shares and reconstructs the secret from the "good" ones, as it is the case for known schemes [RB89, CDD+99] and the one we present later.

6.3.2 Two Impossibility Lemmas

Consider a single-round honest-dealer VSS as defined in the previous section.

Lemma 6.1 *There exists a static, non-rushing adversary such that with non-zero probability some honest players output* `failure` *in the reconstruction phase.*

Proof. Given that $t \geq n/3$, let B, A_0, A_1 be an arbitrary disjoint partition of $\{1, \ldots, n\}$ such that $|B| = t$ and $1 \leq |A_0|, |A_1| \leq t$. We show a strategy for the adversary that forces all players in B to output `failure` with non-zero probability. The adversary corrupts the players in A_0, selects a random secret $\tilde{s}$ and randomly guesses the shares $s_i = (\kappa_i, y_i)$ held by the players in B. By the privacy of the scheme and assuming that he guessed the shares correctly and that $s \neq \tilde{s}$ (which both happens with non-zero probability), he can sample random shares $\tilde{s}_j$ for the corrupted players, so that these, together with the shares of the players in B, are consistent with the secret $\tilde{s}$, and have the same distribution as when sent by the honest dealer. It is now clear that in the reconstruction phase (assumed that the adversary guessed the shares correctly and that $s \neq \tilde{s}$), every player in B has to output `failure`. Indeed, the players in B must definitely not output the incorrect secret $\tilde{s}$. On the other hand, if some player in B outputs the correct secret s (with positive probability), then by corrupting the players in A_1 instead of A_0, but otherwise playing the corresponding game, the adversary creates the same view for the players in B, however with the correct and the incorrect secrets exchanged, and hence this player would now output the incorrect secret (with positive probability), which is a contradiction. □

Lemma 6.2 *There exists a static, rushing adversary such that with non-zero probability some honest player recovers the secret in the reconstruction phase, while some other honest player outputs* `failure`.

Proof. Consider the case $t > n/3$. Let B, A_0, A_1 be an arbitrary disjoint partition of $\{1, \ldots, n\}$ such that $1 \leq |A_0| \leq t - 1$ and $1 \leq |A_1| \leq t$. Note that $2 \leq |B| \leq t$. Let P_i and P_j be distinct members of B. We consider the same adversary as before, except that in the reconstruction phase, the adversary "rushes", and waits until the players in B have broadcast their public shares. He then makes a guess for player P_i's private κ_i, and broadcasts random public shares for the players in A_0, consistent with κ_i and with

the public shares of the players in B and a random secret different from the correct one (which he knows by now). For similar reasons as before we conclude that player P_i does not reconstruct the secret if the guess for κ_i was correct. However, in that case player P_j must reconstruct the secret with positive probability: for if not, corrupting A_0 *and* player P_i (note that this amounts to at most t corruptions), the adversary would not have to guess κ_i anymore, and hence there would be a strategy that makes at least one honest player output `failure` in the reconstruction with probability equal to 1. This contradicts correctness. □

6.4 Bounds for Single-Round Honest-Dealer VSS

6.4.1 Lower Bound on Reconstruction Complexity

We prove the following lower bound. Note that the standard definitions of entropy, conditional entropy, mutual information and conditional mutual information are used throughout this section. We refer to [Bla87] for an introduction to information theory.

Theorem 6.1 *For any single-round honest-dealer VSS scheme, indexed by a security parameter $k \in \mathbb{N}$, which is $(t, n, 1-\delta)$-secure against a* rushing *adversary, the following holds. If $\delta \leq 2^{-\Omega(k)}$ and $n/3 \cdot (1+\epsilon) \leq t < n/2$ for some constant $\epsilon > 0$, then the total number of bits broadcast in the reconstruction phase is lower bounded by*

$$\Omega(kn^2/(n-2t)) .$$

In particular, for $t = \lfloor (n-1)/2 \rfloor$ and for $t \geq q \cdot n$ with $1/3 < q < 1/2$, the total information broadcast in the reconstruction phase is lower bounded by $\Omega(kn^2)$ and $\Omega(kn)$, respectively.

Note that it is immaterial whether the adversary is adaptive or not. Note also that according to this lower bound an arbitrary small linear gap between t and $n/2$ potentially allows to reduce the communication complexity of the reconstruction by a factor of n. It is shown in the next section that this is indeed achievable.

Theorem 6.1 follows immediately from the following proposition and the well known fact (see e.g. [Bla87, CT91]) that the entropy $H(\mathrm{X})$ of a random variable X is a lower bound on the expected bit length of any binary encoding of X.

Proposition 6.1 *Let the random variables $\mathrm{S}_1 = (\mathrm{K}_1, \mathrm{Y}_1), \ldots, \mathrm{S}_n = (\mathrm{K}_n, \mathrm{Y}_n)$ be distributed according to the shares $s_1, \ldots, s_n$ in the single-round honest-dealer VSS scheme. Then, for any set of $n-2t$ public shares $\mathrm{Y}_{i_1}, \ldots, \mathrm{Y}_{i_{n-2t}}$,*

$$H(\mathrm{Y}_{i_1} \cdots \mathrm{Y}_{i_{n-2t}}) \;\geq\; \Omega(kn) .$$

Before going into the proof, consider the lemma below, which states a well known result from authentication theory, which can be found in various literature starting with [Sim84].

Let K,M,Y and Z be random variables (typically key, message, tag and public information of an authentication scheme) with joint probability distribution P_{KMYZ} such that M is independent of K and Z but uniquely defined by Y and Z.

Lemma 6.3 *Given only* Z*, one can compute* $\tilde{\mathrm{Y}}$ *which is consistent with* K *and* Z*, in the sense that* $P_{\mathrm{KYZ}}(\mathrm{K}, \tilde{\mathrm{Y}}, \mathrm{Z}) > 0$*, with probability*

$$p_I \;\geq\; 2^{-I(\mathrm{K};\mathrm{Y}|\mathrm{Z})}\,.$$

Also, given Z *and* Y*, one can compute* $\tilde{\tilde{\mathrm{Y}}}$ *which is consistent with* K *and* Z *and a* $\tilde{\tilde{\mathrm{M}}} \neq \mathrm{M}$ *with probability*

$$p_S \;\geq\; 2^{-H(\mathrm{K}|\mathrm{Z})}\,.$$

In the context of authentication theory, $\tilde{\mathrm{Y}}$ describes an *impersonation* and $\tilde{\tilde{\mathrm{Y}}}$ a *substitution* attack, and p_I and p_S are the corresponding success probabilities.

In the proof of Proposition 6.1, we apply the following corollary, which follows from the fact that a successful impersonation attack is also a successful substitution attack with probability at least 1/2, assumed that M is uniformly distributed among a set of cardinality at least two.

Corollary 6.1 *Let* M *be uniformly distributed among a non-trivial set. Then, given* Z*, one can compute* $\tilde{\mathrm{Y}}$ *which is consistent with* K *and* Z *and a* $\tilde{\mathrm{M}} \neq \mathrm{M}$ *with probability*

$$p_S \;\geq\; 2^{-I(\mathrm{K};\mathrm{Y}|\mathrm{Z})-1}\,.$$

Proof of Proposition 6.1. Let the random variable S represent the secret, uniquely defined by $\mathrm{S}_1, \ldots, \mathrm{S}_n$. Since by the privacy of the scheme the public shares $\mathrm{Y}_{i_1}, \ldots, \mathrm{Y}_{i_{n-2t}}$ are independent of S and hence $H(\mathrm{Y}_{i_1}, \ldots, \mathrm{Y}_{i_{n-2t}})$ does not depend on the distribution of S, we can assume S to be the uniformly distributed. Furthermore, for symmetry reasons, we can focus on the public shares of the players $P_{t+1}, \ldots, P_{n-t}$. Finally, we assume that ϵn is an integer (otherwise we replace it by $\lceil \epsilon n \rceil$). Note that $\epsilon n + (n - 2t) \leq (1+\epsilon)n - 2t \leq t$.

Let $i \in \{1, \ldots, \epsilon n\}$ be arbitrary but fixed, and consider an adversary corrupting the first $i-1$ players $P_1, \ldots, P_{i-1}$ as well the $n - 2t$ players $P_{t+1}, \ldots, P_{n-t}$. One of the goals of the adversary could be to substitute the public shares $\mathrm{Y}_{t+1}, \ldots, \mathrm{Y}_{n-t}$ by false shares $\tilde{\mathrm{Y}}_{t+1}, \ldots, \tilde{\mathrm{Y}}_{n-t}$ that are consistent with the public shares $\mathrm{Y}_1, \ldots, \mathrm{Y}_t$ of the first t players and player P_i's key K_i (and maybe even the keys $\mathrm{K}_1, \ldots, \mathrm{K}_{i-1}$), but that leads to an incorrect secret $\tilde{\mathrm{S}} \neq \mathrm{S}$. Indeed, if the adversary succeeds in this attack, from player P_i's point of view, the $n - t$ public shares $\mathrm{Y}_1, \ldots, \mathrm{Y}_t, \tilde{\mathrm{Y}}_{t+1}, \ldots, \tilde{\mathrm{Y}}_{n-t}$ *could* come from honest and the t shares $\mathrm{Y}_{t+2}, \ldots, \mathrm{Y}_n$ from corrupted players. Hence, P_i clearly cannot compute the correct secret with certainty, and so outputs `failure`. Therefore, the success probability of this attack is at most $\delta \leq 2^{-\Omega(k)}$. On the other hand however, according to the above corollary, applied to $\mathrm{K} = \mathrm{K}_i$, $\mathrm{M} = \mathrm{S}$, $\mathrm{Y} = (\mathrm{Y}_{t+1}, \ldots, \mathrm{Y}_{n-t})$ and $\mathrm{Z} = (\mathrm{K}_1, \ldots, \mathrm{K}_{i-1}, \mathrm{Y}_1, \ldots, \mathrm{Y}_t)$, the success probability is at least $p_S \geq 2^{-I(\mathrm{K}_i;\mathrm{Y}_{t+1}\cdots\mathrm{Y}_{n-t}|\mathrm{K}_1\cdots\mathrm{K}_{i-1}\,\mathrm{Y}_1\cdots\mathrm{Y}_t)-1}$. Therefore, we have

$$I(\mathrm{K}_i; \mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t} | \mathrm{K}_1 \cdots \mathrm{K}_{i-1}\, \mathrm{Y}_1 \cdots \mathrm{Y}_t) \;\geq\; \Omega(k)\,.$$

This holds for every $i \in \{1, \ldots, \epsilon n\}$, and hence, using the chain rule for mutual information, we get

$$I(\mathrm{K}_1 \cdots \mathrm{K}_t; \mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t} | \mathrm{Y}_1 \cdots \mathrm{Y}_t) \;=\; \sum_{i=1}^{\epsilon n} I(\mathrm{K}_i; \mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t} | \mathrm{Y}_1 \cdots \mathrm{Y}_t \, \mathrm{K}_1 \cdots \mathrm{K}_{i-1}) \;\geq\; \Omega(kn)$$

and therefore $H(\mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t}) \geq I(\mathrm{K}_1 \cdots \mathrm{K}_t; \mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t} | \mathrm{Y}_1 \cdots \mathrm{Y}_t) \geq \Omega(kn)$. □

Remark 6.1 As $\mathrm{S}_1, \ldots, \mathrm{S}_t$ gives no information about S, but $\mathrm{S}_1, \ldots, \mathrm{S}_t, \mathrm{Y}_{t+1}, \ldots, \mathrm{Y}_{n-t}$ on the other hand uniquely determines S, we also have $H(\mathrm{Y}_{t+1} \cdots \mathrm{Y}_{n-t}) \geq H(\mathrm{S})$.

In Section 6.7.2 we illustrate the power of rushing by giving an example of a concrete scheme secure against a non-rushing adversary, that beats the lower bound, and sketch a tight lower bound result. We also briefly discuss the minimal complexity of the sharing phase of schemes secure against a non-rushing adversary.

6.4.2 Tightness of the Lower Bound

We first describe a very natural, generic construction of a single-round honest-dealer VSS and then present a particular instantiation that meets the lower bound from the previous section. Rabin and Ben-Or [RB89] first considered a solution of this type. Their scheme reaches our bound in the special case of $t = \lfloor (n-1)/2 \rfloor$. The scheme below differs from theirs in the choice of the authentication code (which will be relevant later on) and in the fact that it optimizes the reconstruction also for non-maximal t.

Let a threshold secret sharing scheme be given as well as an authentication scheme, e.g. based on a family of strongly universal hash functions $\{h_\kappa\}_{\kappa \in \mathcal{K}}$ (see e.g. [Sti95]). To share a secret s, the dealer generates shares $s_1, \ldots, s_n$ according to the secret sharing scheme, and, for each pair of players P_i, P_j, he selects a random authentication key $\kappa_{ij} \in \mathcal{K}$ which will be sent to P_j who will later use it to verify a share contributed by P_i. Then the dealer computes for each share s_i and for each P_j the authentication tag $y_{ij} = h_{\kappa_{ij}}(s_i)$ that should be revealed by P_i at reconstruction time to convince P_j that P_i's share s_i is valid. The dealer then simply sends shares, tags and keys privately to the players who own them. I.e., player P_i receives the share s_i, the tags $y_{i1}, \ldots, y_{in}$ and the keys $\kappa_{1i}, \ldots, \kappa_{ni}$. To reconstruct, every player first broadcasts his share together with the tags, and then "filters out" incorrect shares by verifying the authenticity of the received shares using his keys.

Remark 6.2 The security of this kind of scheme obviously does not rely on the fact that the shares and tags are *broadcast*, i.e., that there is agreement on the received values among the honest players. Therefore, as an alternative to broadcasting s_i and $y_{i1}, \ldots, y_{in}$, player P_i could send s_i and y_{ij} to P_j, for every P_j, using a bilateral (not necessarily secure) channel. In order to emphasize this ambiguity, we say that P_i *publishes* his share and tags.

We use Shamir's secret sharing scheme [Sha79] over a large field K (with "interpolation points" $\omega_1, \ldots, \omega_n$), and the well-known family of hash functions $h_{(\alpha,\beta)}(X) = \alpha X + \beta$ defined over K. While the resulting sharing phase should be clear form the above, we describe the reconstruction phase in more detail.

RECONSTRUCTION PHASE

First, every player P_i publishes his share s_i and tags $y_{i1}, \ldots, y_{in}$. Then, upon receiving $s_1, \ldots, s_n$ and $y_{1j}, \ldots, y_{nj}$, every P_j does the following. He *accepts* those shares s_i with $y_{ij} = \alpha_{ij} \cdot s_i + \beta_{ij}$, and he uses the decoding algorithm of [BW86] to recover a polynomial $f(X)$ of degree at most t such that $f(\omega_i) = s_i$ for at least $n - t$ accepted shares s_i. If the algorithm fails, or if the number of accepted shares s_i with $f(\alpha_i) \neq s_i$ is $n - 2t$ or more, then P_j outputs `failure`, while otherwise $s = f(0)$.

The success probability of a substitution attack of the authentication scheme is $1/|K|$. It follows that the probability of player P_j accepting a false share s_i from player P_i is $1/|K|$. On the other hand, it is not had to see that as long as P_j accepts less than $n - 2t$ false shares, the correct secret is still uniquely defined by the accepted (correct and incorrect) shares. Namely, accepting at most $n - 2t - 1$ false shares, P_j holds $n - t$ correct shares leading to the correct secret while any set of shares leading to a false secret has cardinality at most $n-t-1$, namely the $n-2t-1$ false and t correct shares. Furthermore, in this case, the correct secret can be computed *efficiently* using the decoding algorithm of [BW86].[3] Finally, by comparing all the accepted shares with the reconstructed polynomial, P_j makes sure not to output an incorrect secret in case he has accepted $n-2t$ or more many incorrect shares. Namely, if the decoded polynomial is not correct, then at most t of the $n-t$ correct shares lie on the polynomial, and hence at least $n - 2t$ of the (correct) shares do not.

Theorem 6.2 *For $n/3 \leq t < n/2$ and $k \geq \Omega((n-2t)\log t)$, there exists a single-round honest-dealer VSS scheme, $(t, n, 1 - 2^{-\Omega(k)})$-secure against an (adaptive and rushing) adversary, with a total communication complexity of $O(kn^2/(n-2t))$ bits.*

In particular, for $t = \lfloor (n-1)/2 \rfloor$ and for $t \leq q \cdot n$ with $1/3 \leq q < 1/2$, the communication complexity is $O(kn^2)$ and $O(kn)$, respectively.

Note that the restriction on k is reasonable and essentially disappears in the case of maximal t. Furthermore, note that the communication complexity holds regardless of whether P_i broadcasts s_i and $y_{i1}, \ldots, y_{in}$, or whether he sends s_i and y_{ij} to P_j using a bilateral channel, for every P_j.

Proof. As argued above, an honest player fails to reconstruct the secret only if he accepts $n - 2t$ or more of the t incorrect shares. The probability that this happens is upper

[3] Alternatively, taking powers of an n-th root of the unit as the interpolation points of the sharing polynomial in the Shamir scheme, one can take the decoding algorithm used in [BGW88].

bounded by

$$\binom{t}{n-2t} \cdot \frac{1}{|K|^{n-2t}} \;\leq\; \frac{t^{n-2t}}{|K|^{n-2t}} \;=\; 2^{(n-2t)\log t-(n-2t)\log|K|}\,,$$

while the communication complexity is $O(n^2 \log|K|)$. Hence, choosing the field K in such a way that $\Omega(k) + (n-2t)\log t \leq (n-2t)\log|K| \leq O(k)$ gives an error probability $2^{-\Omega(k)}$ and a communication complexity of $O(kn^2/(n-2t))$. □

Remark 6.3 The choice of the code is not completely arbitrary, since it is important for our later purposes that computation of tags has low arithmetic complexity (here one multiplication and one addition over K) and that the tags are *linear* if α is fixed. Indeed, if y and y' are authentication tags for m and m' with keys (α, β) and (α, β'), respectively, then for every $\lambda \in K$, $\lambda \cdot y + y'$ is an authentication tag for the message $\lambda \cdot m + m'$ with key $(\alpha, \lambda \cdot \beta + \beta')$. Namely,

$$\alpha \cdot (\lambda \cdot m + m') + (\lambda \cdot \beta + \beta') \;=\; \lambda \cdot (\alpha \cdot m + \beta) + (\alpha \cdot m' + \beta') \;=\; \lambda \cdot y + y'\,.$$

Analogue, it can additionally be shown that y is an authentication tag for the message $m + \lambda$ with key $(\alpha, \beta - \alpha \cdot \lambda)$. Furthermore, it is not difficult to see by induction that after l authentications and verifications with the same α, the substitution probability still is $l/(|K| - l + 1)$ (see e.g. [CDD+99]).

6.5 Upper Bound in the Presence of a Corrupted Dealer

In this section, we present a VSS scheme with a one-round reconstruction, where the complexity of the sharing phase matches that of the previous best known VSS for our scenario [CDD+99], but where the reconstruction phase meets our lower bound up to a constant factor. This is at least a factor of n better than previous VSS protocols.

6.5.1 Definition

Since now the dealer might be corrupt as well and so the sharing of the secret takes the form of an interactive protocol, the adversary can not only intrude faults in the reconstruction, but also in the sharing phase. Therefore, our definition operates with two error probabilities, which for a concrete scheme do not have to be equal: first the probability that the distribution of the shares fails to work as supposed, and second the probability that the reconstruction fails, even though the sharing phase succeeded.

Assume an adversary that may corrupt at most t of the n players and the dealer. Additionally, the adversary can be static or adaptive, and rushing or non-rushing. Consider a VSS scheme with an arbitrary sharing phase which results in every player P_i holding a share consisting of a key κ_i and a public share y_i and with a one-round reconstruction phase as in the honest dealer case.

Definition 6.2 *We call such a scheme* $(t, n, 1-\beta, 1-\delta)$-secure *if the following holds, with probability 1 if the dealer remains honest during the sharing phase, and otherwise with probability at least* $1-\beta$ *(taken over the coin flips during the sharing phase).*

- *Privacy: As long as the dealer remains honest, the adversary gains no information about the shared secret* s *as a result of the sharing phase.*
- $(1-\delta)$*-Correctness: After the secret is shared, there exists a unique value* s' *such that in the reconstruction phase each honest player outputs either* s' *or* `failure`*, where for every player the latter happens with probability at most* $\delta < 1$*, independent of* s'*. If the dealer remains honest during the sharing phase then* $s' = s$*.*

Remark 6.4 We stress that the standard definition of unconditionally secure VSS is slightly weaker than ours in that it allows honest players to reconstruct (with small probability) an incorrect value of the secret, even if the dealer was honest in the sharing phase. However, all known VSS schemes essentially fulfill our stronger definition, in particular the most efficient solution known, [CDD+99], fulfills it without any changes in the required communication, while the [RB89] protocol requires some straightforward modifications.

6.5.2 Towards VSS with Optimized Reconstruction

The security of the scheme from the last section evidently completely breaks down in case the dealer is corrupted. In the sharing phase, he could hand out inconsistent shares and inconsistent authentication tags, and, in the reconstruction phase, since he knows all the keys, he could compute correct tags for false shares. This would allow him to disrupt the reconstruction and even to actually cause different secrets to be reconstructed (see the analysis in [CDD+99] of WSS from [RB89]). To remedy this, we have to ensure that the players that remain honest receive consistent shares, and that they accept each others shares at reconstruction, while rejecting false shares. Of course, as mentioned in the introduction, this could in principal be achieved by replacing the dealer of the honest-dealer VSS by a general MPC. This, however, would result in a rather inefficient sharing phase. Also the following approach seems to be no satisfactory solution because of the same reason. We force the dealer to distribute consistent shares $s_1, \ldots, s_n$ by doing a "two-dimensional sharing" as in [BGW88] or [CDD+99] and then every tag y_{ij} for a share s_i is computed in a multi-party fashion, such that it is guaranteed to be correct and the corresponding key is only known to the verifier P_j. Again, doing general MPC would result in a rather inefficient sharing phase; however, the following points provide some intuition as to why the full generality of MPC protocols is not needed, and instead we can do a *specialized* MPC.

1. The "two-dimensional sharing" from [BGW88] or [CDD+99] not only ensures that the honest players hold consistent shares, but also that every share s_i is again correctly shared. Hence, one input to the MPC, s_i, is already correctly shared.
2. We only have to guarantee that a tag is computed correctly, if the player who will later verify it is honest during the sharing phase. At reconstruction, a corrupted player can

always claim a tag to be invalid, even if it were good. For this reason, full VSS of the authentication key will not be necessary.

3. The function to be computed uses only one multiplication and one addition. This will allow us to do the distributed multiplication locally, i.e. no re-sharing as in [GRR98] will be needed.

6.5.3 The CDDHR VSS Sharing Protocol

To describe the sharing protocol from [CDD+99], we start by reviewing the concept of *information checking* (IC), introduced in [RB89]. In essence, an IC scheme provides unconditionally secure "signatures" with limited transferability. More concretely, it allows a *sender* S to provide a *transmitter* T (also called *intermediary*) with a message m and a "signature" σ, such that T can later pass (m, σ) on to a *recipient* R, claiming that m originates from S. The signature σ enables R to verify this. We use the notation $\sigma_m(S, T; R)$ to refer to such a signature. Although in reality the "signing" procedure is an interactive protocol involving all three players and using a broadcast channel, we abuse language slightly and simply say that S "sends the signature $\sigma_m(S, T; R)$ to T". IC must fulfill the following requirements, except with some small error probability. If T and R are honest, then R indeed accepts T's message m (*consistency*). If, on the other hand, S and R are honest, then R rejects any message $m' \neq m$ (*correctness*). Finally, if S and T are honest, then R gets no information on m before T passes (m, σ) on to him (*secrecy*). It is easy to extend this concept and the corresponding protocols to multiple recipients, say $R_1, \ldots, R_n$, by simply executing the single recipient protocol for each possible recipient. We then use the notation $\sigma_m(S, T) = (\sigma_m(S, T; R_1), \ldots, \sigma_m(S, T; R_n))$. For a formal definition and technical details, please refer to [RB89, CDD+99].

One possible implementation is based on the authentication code used in Section 6.4.2. Let K be a large finite field. An IC signature $\sigma_m(S, T; R)$ of a message $m \in K$ is simply an authentication tag $y = \alpha \cdot m + \beta$, where α and β are given to R. By a cut-and-choose technique it can be guaranteed that y is correct (except with small probability), as long as T and R are honest. We refer to [CDD+99] for more details.[4] As argued in Remark 6.3, it does not harm the security of the authentication code to fixed the value α (individually for each R), and, on the other hand, provides the following *linearity* properties. If T holds two signatures $\sigma_m(S, T; R)$ and $\sigma_{m'}(S, T; R)$ and if λ is known to R and T, then T can compute a signature $\sigma_{m+m'}(S, T; R)$ for $m + m'$ and a signature $\sigma_{\lambda m}(S, T; R)$ for λm. This holds analogously in the multi-recipient case. As to efficiency, generating a signature $\sigma_m(S, T; R)$ costs $O(\log |K|)$ bits of communication (see [CDD+99]), generating a signature $\sigma_m(S, T)$ with n recipients costs $O(n \log |K|)$ bits of communication. Furthermore, the secrecy condition holds perfectly while correctness and consistency hold with probability $1 - 2^{-\log |K|}$ for a single-recipient and $1 - 2^{-\log |K| + \log(n)}$ for a multi-recipient signature.

We present the sharing phase of the VSS sharing protocol from [CDD+99], which we will

[4] Note that a different authentication code was chosen in [CDD+99] in order to have a more geometric interpretation, but it is straightforward to adapt their techniques to our choice of the code.

call Pre Share, in a slightly modified version. Namely, for ease of exposition, we use a *symmetrical* polynomial and we omit the signatures made by the dealer (since these are needed only to catch a corrupted dealer early on). Let K be a finite field of size $|K| > n$ and $\omega_1, \ldots, \omega_n \in K$ pairwise different and non-zero.

PROTOCOL Pre Share

1. To share a secret $s \in K$, the dealer chooses a random symmetrical bivariate polynomial f of degree at most t in both variables with s as constant coefficient, i.e. $f(0,0) = s$.
2. To every player P_i, the dealer privately sends the share $s_i = f(\omega_i, 0)$ and the sub-shares $s_{i1} = f(\omega_i, 1), \ldots, s_{in} = f(\omega_i, \omega_n)$ of s_i.
3. For every two players P_i and P_j, the following is done. P_i sends s_{ij} together with a signature

$$\sigma_{s_{ij}}(P_i, P_j) \;=\; (\sigma_{s_{ij}}(P_i, P_j; P_1), \ldots, \sigma_{s_{ij}}(P_i, P_j; P_n))$$

   to P_j. If $s_{ij} \neq s_{ji}$, then P_j broadcasts a complaint, to which the dealer has to answer by broadcasting s_{ji}. If this value does not coincide with P_j's s_{ji}, then P_j accuses the dealer publicly who then has to broadcast P_j's share s_j and subshares $s_{j1}, \ldots, s_{jn}$.
4. If at some point, the broadcast information is inconsistent, the players take some publicly known default sharing.

This protocol stands as a VSS sharing protocol on its own, but with "expensive" reconstruction: $O(n^3 \log |K|)$ bits (all subshares s_{ij} together with their signatures) need to be communicated in order to reconstruct the secret. The fact that it is nevertheless a VSS sharing protocol is based on the following observations.

Proposition 6.2 *After the execution of* Pre Share, *every honest player P_i holds a share s_i and signed sub-shares $s_{i1} \ldots s_{in}$ such that*

1. *If the dealer remains honest, then the adversary has no information about s.*
2. *The sub-shares $s_{i1} \ldots s_{in}$ of any honest player P_i are a correct sharing of s_i, and $s_{ij} = s_{ji}$ holds for all P_i and P_j who remain honest.*
3. *The shares s_i of the honest players are correct shares of a unique value s', which is the secret s if the dealer remains honest.*
4. *For any (corrupted or honest) player P_j, the sub-shares s_{ij} of the honest players P_i are correct shares of P_j's share s_j, which is well defined by the shares s_i of the honest players.*

The communication complexity of this Pre Share protocol is $O(n^3 \log |K|)$ bits, the dealer essentially distributes n^2 sub-shares and each of these sub-shares is signed, where signing costs $O(n \log |K|)$ bits of communication per signature.

Proof.

1. First note that the adversary does not gain any new information by making players complain. Let A be the set of players who have been corrupted during the execution of Pre Share. The existence the symmetrical polynomial

$$d(X,Y) \;=\; \prod_{P_i \in A} \frac{(X-\omega_i)(Y-\omega_i)}{\omega_i^2}$$

   of degree at most t, with $d(0,0) = 1$ and $d(\omega_i, \cdot) = d(\cdot, \omega_i) = 0$ for all $i \in A$, shows that the privacy condition of a solid secret sharing scheme (Definition 4.3) is satisfied. This together with the secrecy property of the signatures proves the claim.

2. If this was not the case, then there would have been complaining.

3. Let the set A consist of $t+1$ honest players. Their shares define a unique secret s'. Let now A' consist of the players in A and a further honest player (if there are only $t+1$ honest players, then we are finished anyway). Let λ_i, $i \in A$, be reconstruction coefficients for the players in A and λ'_i, $i \in A'$, for the players in A'. So we have $s' = \sum_{i \in A} \lambda_i s_i$ and (according to 2.) $s_k = \sum_{i \in A} \lambda_i s_{ki} = \sum_{j \in A'} \lambda'_j s_{kj}$ for all $k \in A'$. It follows that $\sum_{k \in A'} \lambda'_k s_k = \sum_{k \in A'} \lambda'_k \sum_{i \in A} \lambda_i s_{ki} = \sum_{i \in A} \lambda_i \sum_{k \in A'} \lambda'_k s_{ki} = \sum_{i \in A} \lambda_i \sum_{k \in A'} \lambda'_k s_{ik} = \sum_{i \in A} \lambda_i s_i = s'$, hence the shares of the players in A' are still consistent. Inductively, it follows that the shares of all honest players are consistent and hence correct by Remark 5.1.

4. Can be shown with a similar argumentation as above using the fact that every share s_j can be written as a fix linear combination $\sum_k \mu_k s_k$ of the shares of the honest players P_k. □

6.5.4 Computing Tags by a Specialized MPC

Consider now a fixed player P_i after the execution of Pre Share, holding his share s_i and the corresponding sub-shares $s_{i1}, \ldots, s_{in}$ with signatures $\sigma_{s_{i1}}(P_1, P_i), \ldots, \sigma_{s_{in}}(P_n, P_i)$. We now want to compute authentication tags $y_{ij} = \alpha_{ij} \cdot s_i + \beta_{ij}$ for s_i as they are computed by the dealer in the honest-dealer VSS protocol, but without letting the dealer know the keys, $(\alpha_{ij}, \beta_{ij})$ should only be known to P_j.

At the heart, there is the following problem. A player P wants to compute the tag $y = \alpha \cdot m + \beta$ for his secret message m with respect to a player V's secret key α, β. As already mentioned earlier, this will be done by a *specialised* MPC.

We assume that P's message m is already correctly shared by shares $m_1, \ldots, m_n$ and that P holds signatures $\sigma_{m_1}(P_1, P; V), \ldots, \sigma_{m_n}(P_n, P; V)$, verifiable by V. If the protocol

Pre Share from the previous section has been executed, and if P's message m stands for P_i's share s_i, then this is fulfilled with $m_k = s_{ik}$ and $\sigma_{m_k}(P_k, P; V) = \sigma_{s_{ik}}(P_k, P_i; P_j)$.

PROTOCOL MP Auth

1. V chooses a random polynomial f_α of degree at most t with $f_\alpha(0) = \alpha$ and a random polynomial f_β of degree at most $2t$ with $f_\beta(0) = \beta$. For every player P_k, V sends the shares $\alpha_k = f_\alpha(\omega_k)$ and $\beta_k = f_\beta(\omega_k)$ to P_k together with signatures $\sigma_{\alpha_k}(V, P_k; P)$ and $\sigma_{\beta_k}(V, P_k; P)$, verifiable by P.
2. Every player P_k, having received the shares α_k and β_k with the corresponding signatures and holding the share m_k of m, computes $y_k = \alpha_k \cdot m_k + \beta_k$ and, using the linearity property of the signatures, the corresponding signature $\sigma_{y_k}(V, P_k; P)$ and passes y_k and $\sigma_{y_k}(V, P_k; P)$ on to P, who verifies the signature (see point 3. in Section 6.5.2).
3. If P receives all the y_k and all the signatures are good, then he can reconstruct y by interpolation, i.e. by computing a polynomial f_y of degree at most $2t$ with $f_y(\omega_k) = y_k$ for all k and computing $y = f_y(0)$. If some signature $\sigma_{y_k}(V, P_k; P)$ is not correct, then before computing y as above, P passes m_k and $\sigma_{m_k}(P_k, P; V)$ on to V, who verifies the signature and in case of a good signature returns $y_k = \alpha_k \cdot m_k + \beta_k$ to P (see point 2. in Section 6.5.2 for the case V refuses).

Proposition 6.3 *Under the assumptions stated before the protocol, the following holds except with probability* $2^{-\log|K|+O(\log n)}$.

1. *If P and V remain honest during the execution, then $y = \alpha \cdot m + \beta$.*
2. *If P remains honest, then the adversary learns nothing about m.*
3. *If V remains honest, then the adversary learns nothing about α.*

Hence, the tag y can be thought of being computed by some honest player.

Proof. We will prove 1., 2. and 3. under the assumption that the security properties of the signatures hold without error probability; this proves the claim.

1. Let f_m be the polynomial of degree at most t with $f_m(k) = m_k$ and hence $f_m(0) = m$. The n shares $y_k = \alpha_k \cdot m_k + \beta_k$ define a unique polynomial f_y of degree at most $2t$ with $f_y(\omega_k) = y_k$ and $f_y(0) = y = \alpha \cdot m + \beta$, namely $f_y = f_\alpha \cdot f_m + f_\beta$. So, if all n players P_k behave and send y_k with the correct signature to P, then P can compute f_y and hence y. If on the other hand some corrupted player P_k misbehaves and sends an incorrect y_k to P (or an incorrect signature or nothing at all), then P recognizes this and gets the correct y_k from V. Hence, even in this case P gets all the correct y_k's and can therefore reconstruct y.

2. We assume wlog that V is corrupted. If all the corrupted players P_k follow the protocol, then the adversary definitely gets no information at all. If some corrupted player P_k misbehaves (e.g. by sending a bad y_k), then the adversary only learns m_k, which he already knows.

3. We assume that P is corrupted. Note that the adversary does not learn anything new by asking V for a y_k in step 3., since the correct value m_k must be sent to V (otherwise V would not accept the signature and return nothing).
We have to show that the adversary's view of this protocol gives no information about α. The adversary's view, excluding the signatures, consists of m, $m_1, \ldots, m_n$, $y_1, \ldots, y_n$ and α_k and β_k for $k \in A$, where A is the set of corrupted players, with $y_k = \alpha_k \cdot m_k + \beta_k$. Consider the polynomial $d_\alpha(X) = \prod_{k \in A}(\omega_k - X)/\omega_k$ of degree t and the polynomial $d_\beta = -d_\alpha \cdot f_m$ of degree at most $2t$. Note that $d_\alpha(0) = 1$ and $d_\beta(0) = -m$ and $d_\alpha(\omega_k) = 0 = d_\beta(\omega_k)$ for all k in A. This implies that if f_α and f_β are the sharing polynomials for α and β, then for any α', β' with $\alpha' \cdot m + \beta' = y$, the polynomials $f_{\alpha'} = f_\alpha + (\alpha' - \alpha)d_\alpha$ and $f_{\beta'} = f_\beta + (\alpha' - \alpha)d_\beta$ are sharing polynomials for α' and β', consistent with the adversary's view. Note that $f_{\beta'}(0) = \beta - (\alpha' - \alpha)m = y - \alpha' \cdot m = \beta'$. Since f_α and f_β are randomly chosen with $f_\alpha(0) = \alpha$ and $f_\beta(0) = \beta$, the adversary's view of the protocol, excluded the signatures, is independent of α. This together with the secrecy property of the signatures proves the claim. □

The communication complexity of one execution of MP Auth is $O(n \log |K|)$ bits. Namely, V essentially shares α and β. Note that the signatures involved are signatures verifiable by one player, hence they only cost $O(\log |K|)$ bits of communication.

6.5.5 The VSS Protocol

The VSS sharing protocol now works as follows. First, Pre Share is applied to the secret and then, by applying MP Auth to the shares, the sub-shares and signatures are stripped off and replaced by tags for the actual shares:

SHARING PHASE

1. The above protocol Pre Share is executed on the secret s. As a result, every player P_i holds a share s_i, sub-shares $s_{i1}, \ldots, s_{in}$ and signatures $\sigma_{s_{i1}}(P_1, P_i), \ldots, \sigma_{s_{in}}(P_n, P_i)$.
2. For every player P_i, tags $y_{i1}, \ldots, y_{in}$ for s_i are computed by executing MP Auth with every player P_j on the message s_i and P_j's randomly chosen key $(\alpha_{ij}, \beta_{ij})$.

Note that all the sub-shares s_{ij} and signatures $\sigma_{s_{ij}}(P_j, P_i)$ are only temporarily used and can be deleted at the end of the protocol. For the reconstruction, which works as in the

honest-dealer case (see Section 6.4.2), only the shares, the tags and the keys are needed.

Theorem 6.3 *For $n/3 \leq t < n/2$ and $k \geq \Omega((n-2t)\log t)$, there exists a single-round VSS scheme, $(t, n, 1-2^{-\Omega(k)}, 1-2^{-\Omega(k)})$-secure against an (adaptive and rushing) adversary, with a sharing complexity of $O(kn^3)$ and a reconstruction complexity of $O(kn^2/(n-2t))$ bits.*

In particular, for $t = \lfloor (n-1)/2 \rfloor$ and for $t \leq q \cdot n$ with $1/3 \leq q < 1/2$, the reconstruction complexity is $O(kn^2)$ and $O(kn)$, respectively.

Proof sketch. In the case of a maximal t, i.e. $t = \lfloor (n-1)/2 \rfloor$, we can take the above scheme over a field K with $\log |K| = \Theta(k)$. Privacy and correctness follow from Propositions 6.2 and 6.3. The communication complexity of the Pre Share protocol is $O(kn^3)$, of the MP Auth protocol it is $O(kn)$. Therefore, the communication complexity of the sharing protocol, which calls Pre Share once and MP Auth n^2-times, is $O(kn^3)$. The communication complexity of the reconstruction is as in the honest-dealer VSS $O(kn^2)$ bits.
In the case of an arbitrary t, we take the above scheme over a field K with $(n-2t)\log |K| = \Theta(k)$. However, to keep the error probabilities from Proposition 6.2 and 6.3 at $2^{-\Omega(k)}$, the signatures (but not the shares) used in the protocols Pre Share and MP Auth are computed in an extension field $K' \supseteq K$ of degree $n-2t$, such that $\log |K'| = \Theta(k)$. □

6.6 Applications to MPC with Pre-processing

As an application of the above described VSS scheme, we will now present a general MPC protocol in the pre-processing model [Bea91]. Our protocol is secure against an adversary who can corrupt up to t of the players, where $t < n/2$.

6.6.1 MPC with Pre-processing

The idea behind MPC with pre-processing, introduced by Beaver [Bea91], is to do as much work as possible in a *pre-processing phase*, before the inputs and even the function respectively the circuit to be computed are known. This is to reduce the work and the assumptions on the communication network required in the *computation phase* when the inputs and circuit have actually become available.

This is based on circuit randomization and a generic construction that can be applied to any MPC protocol based on a *linear* VSS. The computation phase doesn't require secure channels, it only consists of broadcasting information and performing the local computations necessary for VSS reconstructions. It should therefore be clear that MPC in the pre-processing model benefits from VSS with optimized reconstruction.

We sketch the generic protocol for MPC with pre-processing. Assume that adequate upper bounds on the number of inputs and multiplication gates in the future circuit are known. In the pre-processing phase, each player chooses a sufficient number of independent

random values a and VSS'es them. Next, the players jointly prepare a sufficient number of random triples r, r' and r'' such that $r'' = rr'$ and such that each of these values is VSS'ed. Note that mutual randomness is easily achieved by having players VSS random values, and taking the sum of those as a mutually random value. By the linearity property, this random value is effectively VSS'ed. By invoking the general MPC protocol, products can be securely computed with the result VSS'ed.
In the computation phase, inputs and circuit are known. Assume for simplicity that each player has a single private input value. Each player then takes his actual private input s, and simply broadcasts the difference $\epsilon = a - s$ between this input s and the random value a he VSS'ed in the pre-processing phase. Subsequently, all players locally compute their shares in s from the shares in a they hold and the now public value ϵ. In the computation phase, the addition gates are handled locally while to multiply two shared values s and s', a fresh precomputed random triple (r, r', r'') is taken, the differences $\delta = s - r$ and $\delta' = s' - r'$ are revealed by invoking the reconstruction phase. Since

$$ss' \;=\; (r+\delta)(r'+\delta') \;=\; rr' + \delta' r + \delta r' + \delta\delta' \;=\; r'' + \delta' r + \delta r' + \delta\delta'$$

every player P_i can locally compute a share of ss' from the shares of r, r' and r'' and the values δ and δ'. Note that linearity of the VSS facilitates these steps.

6.6.2 Applying Our VSS to MPC with Pre-processing

We first argue that our VSS can be made to have the required linearity properties. Note that Shamir shares trivially possess these properties, so it suffices to focus on the authentication code. As shown in Section 6.4.2, the only thing we need to do is to fix throughout the computation for each individual player the values α that are part of his verification keys (α, β).

For a field K with $\log|K| = \Theta(k)/(n-2t)$, the protocol now works as follows. In the pre-processing phase, the random input values a are treated just as above, based on our VSS. In order to prepare the random triples, we use the general MPC techniques of [CDD+99] to prepare triples r, r' and r'' with $r'' = rr'$ as described earlier, except that, for non-maximal t, the signatures are computed in an extension field $K' \supseteq K$ of degree $n - 2t$, as described in the proof of Theorem 6.3. This results in a VSS of these values according to [CDD+99] (i.e. according to the protocol Pre Share from Section 6.5.3). We can convert these to sharings as they would have been produced by our VSS, we simply apply the protocol MP Auth (see Section 6.5). Hence, all necessary pre-processing information will be shared according to our VSS. The computation phase can now proceed based on the reconstruction phase of our VSS.

As to efficiency, generating the sharings of r and r' consists essentially of $O(n)$ executions of Pre Share, and thus this has complexity $O(kn^3)$ bits. The computation of the sharing of r'' costs according to [CDD+99] $O(kn^4)$ bits of communication, assuming everyone coorperates. Multi-party computing the tags is negligible compared to the rest, namely $O(kn^3)$. Hence, we have a best case complexity of $O(kn^4)$. If a corrupted player P_i refuses to coorperate at some point during the computation of a triple (r, r', r''), then this triple

is recomputed while player P_i's part of the protocol is executed in public (with default randomness) till the end of the pre-processing. This way, the adversary cannot slow down the computation substantially. Hence we have

Theorem 6.4 *Let $\mathcal{C}$ be an arithmetic circuit over a field K with $|\mathcal{C}|$ multiplication gates, where $\log |K| = \Theta(k)/(n-2t)$ and $n/3 \leq t < n/2$. Communicating $O(|\mathcal{C}|kn^4)$ bits in a pre-processing phase, there exists a MPC protocol, secure against an adversary who can corrupt up to t of the n players, computing the circuit $\mathcal{C}$ with $O(|\mathcal{C}|kn^2/(n-2t))$ bits of communication. The failure probability of the MPC protocol is $2^{-\Omega(k)+|\mathcal{C}|}$.*

In particular, for $t = \lfloor (n-1)/2 \rfloor$ and for $t \leq q \cdot n$ with $1/3 \leq q < 1/2$, the computation phase requires $O(|\mathcal{C}|kn^2)$ and $O(|\mathcal{C}|kn)$ bits of communication, respectively.

The most efficient previously known protocol for MPC with pre-processing in our model is based on [CDD+99]. Note that this would result in a pre-processing phase with complexity of the same order as in our case. However, due to VSS with optimized reconstruction, we gain an efficiency improvement of a multiplicative factor of at least n in the computation phase of our protocol.

6.7 Concluding Remarks

6.7.1 General Adversaries

We now go beyond threshold security by sketching how to adjust our VSS and MPC protocols to be secure against a general Q^2 adversary.

By replacing the bivariate polynomial sharing in Pre Share by the span program based distributed commitment scheme from [CDM00] (see Section 3.5.4 or 5.4), we are in the same position as described by Proposition 6.2, except that 4. is not guaranteed, i.e. the share s_i of a corrupted player P_i is not necessarily correctly shared by the sub-shares s_{ji} of the honest players P_j. But this can easily be achieved by doing another level of sharing: every player P_i shares his share s_i with the DC protocol from [CDM00] where every player P_j insists that the share they get of s_i is the sub-share s_{ji}.
In the MP Auth protocol, replacing the threshold sharings of the values α and β by sharings based on *mutliplicative* span programs, Proposition 6.3 remains intact.
This results in a VSS scheme secure against a general Q^2 adversary. Furthermore, the sharing and reconstruction complexities are $O(knd^2)$ and $O(knd)$ bits, respectively, where $d \geq n$ is the size of the span program, while the respective complexities of the general adversary VSS scheme suggested in [CDD+99] are both $O(knd^3)$ bits (even though one could achieve $O(knd^2)$ using their techniques in a more elaborate way).

Based on this general adversary VSS scheme, similar to the previous section, one can achieve a general MPC protocol, secure against a general Q^2 adversary, which in the pre-processing model has a communication complexity of $O(|\mathcal{C}|knd)$ bits, compared to $O(|\mathcal{C}|knd^3)$ which would be achieved by the general adversary MPC protocol from [CDD+99] (respectively $O(|\mathcal{C}|knd^2)$ using their techniques in a more elaborate way).

6.7.2 The Power of Rushing (Honest Dealer Case)

We show that our tight lower bound from Section 6.4.1 does not hold if the adversary does not rush, and instead selects the corrupted shares he will broadcast in the reconstruction phase before it has started. We also sketch a lower bound and outline some applications, namely to a scenario in which the amount of information sent in the *sharing phase* is to be minimized.

Let K be a finite field with $|K| > n$, and take Shamir's threshold scheme defined over K with threshold parameters t and n. Cabello, Padró and Sáez [CPS99] have proposed the following so-called robust secret sharing scheme. To share a secret s in this scheme, the honest dealer selects a random field element ρ, independently generates full sets of Shamir-shares for the secrets s, ρ and $\rho \cdot s$, and privately distributes the shares to the players.

Given a set A of at least $t+1$ shares (which possibly contains corrupted shares), consider the three values s', ρ' and τ' that are computed by applying the reconstruction procedure of Shamir's scheme to the shares in A. The crucial observation is that if $s \neq s'$ and if the corrupted shares are *independently distributed from* ρ, the probability that $s' \cdot \rho' = \tau'$ is at most $1/|K|$. Hence, given for instance a trusted party available for reconstruction, connected with each player by an independent private channel, the independence requirement is satisfied and although the secret may not always be reconstructed from a qualified set, an incorrect secret is detected with high probability.

We note the following application of this scheme in our scenario of a *non-rushing* adversary. By assumption, this is an adversary who chooses the corrupted shares before the reconstruction phase. This ensures that the independence requirement stated before is satisfied. Let k be a security parameter and $t < n/2$ and assume additionally that $|K| \geq 2^{n+k}$. Let the sharing phase be according to the scheme of [CPS99]. The reconstruction works as follows. Every player publishes his shares s_i, ρ_i and τ_i. Then, every player reconstructs for every set A of size $t+1$ from the shares of the players in A s', ρ' and τ', supposedly equal to s, ρ and τ, respectively. If all such triples which additionally satisfy $s' \cdot \rho' = \tau'$ coincide in s', then this s' is output and otherwise `failure`. This way, the probability with which an honest player outputs `failure` is at most $2^n \cdot 2^{-n-k} = 2^{-k}$. Note however, that the computational complexity of the reconstruction phase is exponential in n.

For the case that $k > n$ and $n = 2t+1$, we now sketch an argument showing that the sharing phase of this scheme is optimal, up to constant factors. A basic result in secret sharing says, informally speaking, that the size of individual shares is at least the size of the secret, and hence the question that remains is whether the error probability ϵ of the above scheme is optimal. We define an adversary who flips a random coin and either corrupts the first t players, or the last t players. In either case, he makes a random guess $\tilde{s}_{t+1}$ for the share s_{t+1} that player P_{t+1} received from the honest dealer in the sharing phase (consisting of s_{t+1}, ρ_{t+1} and τ_{t+1} in the above scheme), deletes the correct shares received from the dealer by the corrupted players, and instead chooses random corrupted shares, consistent with his guess $\tilde{s}_{t+1}$ and with a random secret $\tilde{s}$. Assuming that the correct secret s was chosen at random by the honest dealer, if the adversaries' guess for

player P_{t+1}'s share is correct, then there is no way for any reconstruction procedure to distinguish between s and $\tilde{s}$. Hence, in order for ϵ to be upper bounded by $2^{-\Omega(k)}$, the size of each individual share must be $\Omega(k)$.

Although it is generally not very realistic to assume that the adversary is not rushing, it is possible to construct a "simultaneous broadcast" channel on top of the "secure channels with broadcast model". Namely, simply have each player first VSS their values, e.g. by using the schemes of [RB89, CDD+99], after which all VSSes are opened. Using the concrete scheme above, this procedure would ensure that shares are "broadcast simultaneously", and hence that the required independence is achieved, at the cost of increased complexity of the reconstruction phase and use of private channels in that phase. The advantage, however, is that the efficiency of the *sharing phase* has been substantially improved.

6.7.3 Open Problems

The lower bound proof from Section 6.4.1 only applies to single-round honest-dealer VSS schemes where it is guaranteed that no incorrect secret is reconstructed (the worst that can happen is that some player outputs `failure`). It would be nice to find a lower bound for schemes which are somewhat more liberal in that with small probability *anything* may happen, even that players reconstruct an incorrect secret (without detecting it). We have been able to prove the same lower bound for this larger class of schemes assuming that the keys $k_1, \ldots, k_n$ are independently distributed given all public shares $y_1, \ldots, y_n$. Even though this is satisfied by all known schemes it is by no means clear that this can always be assumed. Hence, the problem is still unsolved.

With respect to the above section, another open problem is that of finding an (honest-dealer) VSS scheme with cheap sharing phase as above, but where the reconstruction is still efficient.

Chapter 7

Non-interactive Distributed-Verifier Proofs

7.1 Introduction

> *All truth passes through three stages. First, it is ridiculed. Second, it is violently opposed. Third, it is accepted as being self-evident.*
>
> — Arthur Schopenhauer

Commitment schemes play an important role as a primitive in cryptographic protocols. Applications are found for instance in the construction of zero-knowledge proofs and arguments, MPC and threshold cryptography. Using a commitment scheme, a player can commit to a secret value s by publishing a *commitment* C, in such a way that the commitment C reveals nothing about the secret s, i.e., the scheme is *hiding*. The player can later *open* C to reveal s in a way verifiable by everyone else, i.e., it is *binding* in the sense that the player can't open C to any other value than s.

Many protocols using commitments require a player at some point to prove certain relations among a set of committed values, without revealing these committed values in the process. Assuming that addition or multiplication of secret values is well-defined, a player committed to s, s' and s'' will typically be required to prove that $s'' = s + s'$ or that $s'' = ss'$. If the commitment scheme is *homomorphic*, as is the case with many known commitment schemes, the additive relation is trivial to handle, even non-interactively. A *commitment multiplication proof*, i.e., a secure protocol to handle the multiplicative relation, is generally less trivial to design.

In the two-player setting, there exist efficient *interactive* zero-knowledge protocols for all known homomorphic schemes [CD98]. These protocols can be adapted in a natural way to a distributed setting with n players acting as verifiers (and where up to t of them might be corrupted) using standard methods, for instance by simply letting each of the players be engaged in a separate run of the two-player protocol with the prover, or by letting the players *jointly* generate the random challenge for the zero-knowledge proof.

An elegant alternative approach was proposed in [Abe99] by taking advantage of the fact that sufficiently many players are guaranteed to be honest. Namely, it was shown how to

handle commitment multiplication proof in this distributed setting *non-interactively* in the case of Pedersen's discrete logarithm based commitment scheme. This commitment multiplication proof essentially consists of a few Pedersen VSS's and is non-interactive (from the prover's point of view) in case everyone plays honestly, while the prover might have to answer accusations otherwise. We call this *non-interactive with accusing*. Moreover, it is totally non-interactive if $t < n/3$.

In this chapter, we first introduce the notion of *zero-knowledge distributed-verifier proofs* (Section 7.3), which shall capture the security properties of Abe's commitment multiplication proof protocol. We then improve that protocol by proving that a building block used in its construction in fact already constitutes a commitment multiplication proof (Section 7.4). This not only leads to a simplified exposition, but also saves on the required number of invocations of Pedersen's VSS. Next we show a new technique to construct non-interactive proofs of partial knowledge in this distributed-verifier setting, thereby extending the results of [CDS94] for the interactive two-player case. This allows for instance to prove non-interactively the knowledge of ℓ out of m given secrets, without revealing which ones. As an application, it allows to make the proof of correctness of a ballot in the [CFSY96] voting scheme non-interactive without resorting to random oracles. We also show how to construct efficient non-interactive zero-knowledge proofs for circuit satisfiability in the distributed setting.

The material in this chapter is based on [ACF02].

7.2 Preliminaries

7.2.1 Model

We consider a communication network among a dealer and n players $P_1, \ldots, P_n$ as described in Section 2.2. However, due to the somewhat different context, we also refer to the dealer as *prover* and to the players as *verifiers*. We assume the adversary to be *static*, meaning that he has to corrupt the players *before* the execution of the protocol. For simplicity, we only consider a *threshold* adversary, where either $t < n/3$ or $t < n/2$, though all results carry over to general Q^2 and Q^3 adversaries (see Section 7.5). Some of the protocols are computationally and some are perfectly secure; depending on this, the adversary is assumed to be computationally bounded or unbounded. In case of a computationally bounded adversary, the requirements on the network may be relaxed, since secure channels as well as broadcast can be implemented from insecure channels by cryptographic means. But since the adversary is restricted to be static, we may without loss of generality treat the cryptographically implemented channels as being perfect.

7.2.2 Commitments and Pedersen's Commitment Scheme

Informally, a *commitment scheme* (of the kind we consider) is given by a function family

$$\mathsf{com}_{pk} : \ \mathcal{S} \times \mathcal{R} \ \rightarrow \ \mathcal{C}$$

indexed by a *public key* pk, where $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{C}$ are finite sets that may depend on pk. In a set-up phase, a concrete public key pk and thus function com_{pk} is fixed in a prescribed manner. By publishing a *commitment* $C = \mathsf{com}_{pk}(s, r)$ for a random $r \in \mathcal{R}$, such a scheme allows a party, P_i, to *commit* herself to a secret $s \in \mathcal{S}$, such that the commitment C reveals nothing about the secret s (*hiding property*) while on the other hand P_i can *open* C to s by publishing (s, r) *but only to* s (*binding property*).

More formally, a commitment scheme consists of two poly-time algorithms: A key-generation algorithm which takes as input 1^k, where k is a security parameter, and outputs a public key pk; and an algorithm which allows to compute $C = \mathsf{com}_{pk}(s, r)$ given a public key pk, $s \in \mathcal{S}$ and $r \in \mathcal{R}$. The hiding property requires that no *distinguisher* (on input pk) can distinguish $C = \mathsf{com}_{pk}(s, r)$ from $C' = \mathsf{com}_{pk}(s', r')$ with non-negligible advantage, where $s, s' \in \mathcal{S}$ are chosen by the distinguisher and $r, r' \in \mathcal{R}$ are chosen at random. The binding property requires that no *forger* (on input pk) can compute $s \in \mathcal{S}$ and $r, r' \in \mathcal{R}$ such that $\mathsf{com}_{pk}(s, r) = \mathsf{com}_{pk}(s, r')$. If the distinguisher respectively the forger is restricted to be poly-time, then the scheme is said to be *computationally* hiding respectively binding, while without restriction on the distinguisher respectively the forger, it is said to be *unconditionally* hiding respectively binding. It is well known, and easy to see, that only one of the two properties, hiding and binding, can be unconditional, while the other can at most be computational.

For a public key pk, generated by the key-generation algorithm, we call $\mathsf{com}_{pk} : \mathcal{S} \times \mathcal{R} \to \mathcal{C}$ an *instantiation* of the commitment scheme.

If $\mathcal{S}$ is a field K (or, more generally, a ring), then such a commitment scheme is called *homomorphic*, if the following holds: There exists a poly-time algorithm which takes as input the public key pk, commitments $C = \mathsf{com}_{pk}(s, r)$ and $C' = \mathsf{com}_{pk}(s', r')$, and a number $\lambda \in K$, and outputs a commitment C'', and there exists a poly-time algorithm which take as input pk, s, s', r, r' and λ, and outputs r'' such that $C'' = \mathsf{com}_{pk}(s + \lambda s', r'')$.

A well known homomorphic commitment scheme is the Pedersen commitment scheme [BKK87, BCC88, Ped91]: The public key consists of a description of a (multiplicative) group G of prime order q, in which the *DL-assumption* holds, i.e., in which computing discrete logarithms is (assumed to be) hard, and of two randomly chosen generators g and h of G. G could for instance be a suitable subgroup of $\mathbb{F}_p^*$ for a prime p, or a suitable subgroup of an elliptic curve. For such a public key pk, com_{pk} is then given by

$$\begin{array}{rccc} \mathsf{com}_{pk}: & \mathbb{F}_q \times \mathbb{F}_q & \to & G \\ & (s, r) & \mapsto & g^s h^r \end{array}$$

To simplify notation, we leave G implicit and simply write $pk = (g, h)$. This scheme is unconditionally hiding and computationally binding, based on the underlying hardness of computing discrete logarithms in G. Furthermore, it is homomorphic: If $C = \mathsf{com}_{g,h}(s, r)$, $C' = \mathsf{com}_{g,h}(s', r')$ and $\lambda \in K$, then $\mathsf{com}_{g,h}(\lambda s + s', \lambda r + r') = C^\lambda \cdot C'$.

Note that the group G is a module over $\mathbb{F}_q$ (and, as $\mathbb{F}_q$ is a field, it is even a vector space over $\mathbb{F}_q$) where the number multiplication is given by exponentiation. Because of that, and because it will be convenient, we will from now on view G as an *additive* group, even

though this is very uncommon in the considered context. Therefore, for $C = \mathsf{com}_{g,h}(s,r)$, now given by $\mathsf{com}_{g,h}(s,r) = s \cdot g + r \cdot h$, and $C' = \mathsf{com}_{g,h}(s',r')$, we can write

$$\mathsf{com}_{g,h}(\lambda s + s', \lambda r + r') \;=\; \lambda C + C' .$$

7.2.3 Pedersen's Verifiable Secret Sharing Scheme

We recall Pedersen's VSS scheme [Ped91] which is based on Shamir's secret sharing scheme and (an instantiation of) Pedersen's commitment scheme $\mathsf{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \to G$. Let t and n be threshold parameters, and let $\omega_1, \ldots, \omega_n \in \mathbb{F}_q$ be pairwise distinct and non-zero.

SHARING PHASE

1. To share a secret $s \in \mathbb{F}_q$, the dealer chooses a random polynomial $f_s(X) = a_0 + a_1 X + \ldots + a_t X^t \in \mathbb{F}_q[X]$ of degree at most t with constant coefficient $a_0 = s$, and he commits to this *sharing polynomial* $f_s(X)$ by broadcasting commitments $A_0, \ldots, A_t$ of $a_0, \ldots, a_t$, respectively. For every player P_i, the dealer computes the share

   $$s_i \;=\; f_s(\omega_i) \;=\; s + a_1\omega_i + \cdots + a_t\omega_i^t \in \mathbb{F}_q$$

   and he opens the corresponding commitment

   $$C_i \;=\; A_0 + A_1\omega_i + \cdots + A_t\omega_i^t \in G$$

   privately to P_i, using the homomorphic property of the commitment scheme.
2. If P_i does not accept the opening, then P_i broadcasts an accusation against the dealer.
3. To any accusation of a player P_i, the dealer responds by opening C_i publicly.
4. If he fails to do this correctly then the sharing is rejected, otherwise it is accepted.

After the execution of this protocol, assumed that it has been accepted, every player P_i is committed to his share s_i by the commitment C_i, and he holds the corresponding information to open it. Hence, the reconstruction works as follows.

RECONSTRUCTION PHASE

Every player P_i publicly opens C_i to s_i. The shares s_i that have been correctly opened are then taken to reconstruct the secret by interpolation.

The above sharing and reconstruction phases form a computationally secure VSS if (and only if) $t < n/2$. Privacy holds perfectly while correctness holds under the assumption that computing discrete logarithms is hard.
The scheme can be made completely non-interactive from the dealer's point of view in case $t < n/3$ by replacing the steps 3. and 4. by

3.' If the number of accusations is larger than t, then the sharing is rejected, otherwise it is accepted.

Namely, in this case, if the sharing is accepted then there are at least $t+1$ honest players that have not accused the dealer in step 2. and hence have a consistent sharing that allows to reconstruct the secret (see also the proof of Proposition 7.1).

Remark 7.1 Using the notation from Chapters 3 and 4, Shamir's secret sharing scheme is given by $\mathcal{M} = (\mathbb{F}_q, M, \psi, n)$ where, for $i = 1, \ldots, n$, the i-th row of M equals $(1, \omega_i, \ldots, \omega_i^t)$ and is labeled by i (see Example 3.3). It is interesting to note that not only $\mathbf{s} = (s_1, \ldots, s_n)^T \in \mathbb{F}_q^n$ is computed as a sharing of s but also $\mathbf{C} = (C_1, \ldots, C_n)^T \in G^n$ is a sharing of A_0. This makes sense (with respect to Definition 3.3) as G is a $\mathbb{F}_q$-module. However, in contrast to the s_i's, the C_i's are *a priory* guaranteed to be correct, since they are publicly computed.

Remark 7.2 Consider an accepted execution of the sharing protocol. By correctness, a secret value s' is fixed that can later be reconstructed. Since the information used by the players in the reconstruction originated with the dealer, we can conclude that the dealer knows this secret. In fact, it is straightforward to show, as we do later on, that the dealer not only knows s' but he also knows how to open the commitment A_0 used in the sharing protocol (to s').

7.3 Distributed-Verifier Proofs

7.3.1 Definition

Consider two sets $\mathcal{W}$ and $\mathcal{I}$ and an efficiently verifiable relation $R \subseteq \mathcal{W} \times \mathcal{I}$. Given some *public information* $I \in \mathcal{I}$, the prover wants to convince the verifiers $P_1, \ldots, P_n$ that he knows a *witness* $w \in \mathcal{W}$ with $(w, I) \in R$.

Definition 7.1 *A* distributed-verifier proof (of knowledge) *for relation R is a protocol among the prover and the verifiers $P_1, \ldots, P_n$, with a common input I, a private input w by the prover and a public output* `accept` *or* `reject`*, such that the following security properties hold, even if up to t of the n verifiers as well as the prover might be corrupted by the adversary.*

- *Correctness: If the prover remains honest and $(w, I) \in R$, then the output is* `accept`.

- *Soundness: There exists a knowledge extractor that can efficiently compute from the joint view of the honest verifiers a witness w' satisfying $(w', I) \in R$, assuming that the output of the protocol is* `accept`.

This soundness condition can come in three flavors: perfect, unconditional, *or* computational. *Meaning that the condition holds with probability 1, with overwhelming probability, or with respect to a computationally bounded adversary under some computational assumption, respectively.*

A distributed-verifier proof is called non-interactive, *if the structure of the protocol is as follows. The prover sends to every verifier one message, a personal* partial proof, *and then every verifier votes to either accept or reject the proof, depending on whether he accepts or rejects his partial proof, and the outcome of the protocol is* `accept` *if and only if not more than t verifiers vote for rejection.*
It is called non-interactive with accusing, *if it is non-interactive except that in case there are some rejections, the prover must broadcast the corresponding partial proofs, and the outcome of the protocol is* `accept` *if and only if none of these broadcast proofs is rejected.*
Finally, it is called zero-knowledge, *if the adversary can simulate his view of the protocol.*

The above soundness condition highlights the power of the distributed-verifier setting in two ways: (1) The prover is *not* given to the knowledge extractor as a *rewindable* black-box. Thus, no rewinding argument is needed to prove the soundness of a protocol. (2) In case of *perfect* soundness it asserts that there is no knowledge error. Hence, acceptance of the proof always implies the knowledge of a witness w'. Of course, one can relax the definition by allowing to rewind the prover so that it becomes seamless with the standard definition of proof of knowledge [BG92] with a single verifier.

7.3.2 Pedersen's VSS as a Distributed-Verifier Proof

Let $\mathsf{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \to G$ be an instantiation of Pedersen's commitment scheme. For a commitment $C = \mathsf{com}_{g,h}(s, r)$ let $\mathsf{Proof}_{g,h}(C)$ denote an execution of the sharing phase of Pedersen's VSS scheme with secret s, except that in step 1. of the protocol, $A_0 = C$ is taken as commitment of $a_0 = s$. Then, $\mathsf{Proof}_{g,h}(C)$ is a zero-knowledge proof that the dealer can open the commitment C. More formally, for relation

$$R_{g,h} \;=\; \{((s,r), C) \mid s, r \in \mathbb{F}_q, C = \mathsf{com}_{g,h}(s,r)\} \;\subseteq\; (\mathbb{F}_q)^2 \times G$$

we have

Proposition 7.1 *Protocol $\mathsf{Proof}_{g,h}$ is a perfectly sound zero-knowledge distributed-verifier proof for relation $R_{g,h}$, non-interactive in case $t < n/3$ and non-interactive with accusing in case $t < n/2$.*

We stress that we have soundness and zero-knowledge independent of the quality of the commitment scheme $\mathsf{com}_{g,h}$. In fact, this holds even in case the discrete logarithm $\log_g h$ is known and hence the binding property does not hold at all.

Proof of Proposition 7.1. Since, to any possible accusation, the honest prover only broadcasts correct information, the proof will be accepted. It remains to show soundness and zero-knowledge.

Soundness: Assume that the proof has been accepted, and let A be the set of honest players, respectively, in case of $t < n/3$, the set of honest players who have not accused the dealer. In any case, $|A| \geq t+1$ and every player P_i in A can open his commitment C_i to, say, s_i'. Write $\mathbf{C} = (C_1, \ldots, C_n)^T$ and let $\boldsymbol{\lambda} \in \mathbb{F}_q^{d_A}$ be a reconstruction vector for A (as defined in Definition 3.3). By the homomorphic property, the players in A can jointly open the commitment $C' = \boldsymbol{\lambda}^T \cdot \mathbf{C}_A$, on the other hand, by the observations from Remark 7.1, it holds that $\boldsymbol{\lambda}^T \cdot \mathbf{C}_A = C$.

Zero-knowledge: We make use of the following well known property of Shamir's secret sharing scheme. Any t shares together with the secret allow to reconstruct the sharing coefficients (by taking $\mathbb{F}_q$-linear combinations). This not only holds when the secret and the sharing coefficients are sampled from $\mathbb{F}_q$ but also when sampled from a finite $\mathbb{F}_q$-module.

Let A be the set of corrupted players. We may assume that $|A| = t$. Given the commitment C for s, the players in A can simulate their view of the protocol as follows. For every $i \in A$ they choose $s_i \in \mathbb{F}_q$ at random and compute a (random) commitment $C_i \in G$ for s_i. Clearly, the s_i and C_i's are distributed as in the protocol. Then, viewing the computed C_i's as shares of C, they compute the corresponding sharing coefficients $A_1, \ldots, A_t$. □

7.3.3 Distributed-Verifier Proof versus Adapted Two-Party Proofs

The standard approach to construct a zero-knowledge proof in a distributed-verifier setting is to take a two-party public-coin interactive zero-knowledge proof (see e.g. [BCC88] for general or [CD98] for efficient special-purpose proof protocols) and adapt it to the distributed-verifier setting by letting the players/verifiers *jointly* generate the random challenge required for the interactive proof. For this, all players simultaneously share a random value using a VSS scheme. Then, all these values are reconstructed and the sum is taken as the random challenge. Therefore, this solution involves about n VSS executions. Note that this approach does not take into account that the number of corrupted players is limited.

On the other hand, we have seen that there exists a zero-knowledge distributed verifier proof which consists of one single VSS execution. Also all distributed-verifier proofs from the coming Section 7.4 essentially consists of a constant (in n) number of VSS executions.

However, this advantage vanishes in the typical application to MPC, where for the secure multiplication of two shared values (with committed shares) usually all players have to invoke a commitment multiplication proof simultaneously. Namely, in that case, it suffices in the above standard solution for the players to jointly generate one random challenge, valid for all simultaneous executions. Hence, the expensive part of the proof protocol has to be executed only *once*, while the commitment multiplication proof protocol from the coming section needs to be executed n times in that same situation.

Nevertheless, besides the advantages pointed out in Section 7.3.1, the proof protocols presented below seem to be superior from a complexity point of view in a setting where only one proof has to be executed, or where proofs are executed in sequence rather than parallel, for instance when a party sets up a distributed key for a threshold cryptosystem and needs to prove correctness of the key.

7.4 Constructing Distributed-Verifier Proofs

7.4.1 An Improved Commitment Multiplication Proof

Consider two instantiations of Pedersen's commitment scheme over the same group G: $\mathsf{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \to G$ and $\mathsf{com}_{C',h} : \mathbb{F}_q \times \mathbb{F}_q \to G$. Note that

$$\mathsf{com}_{C',h}(s,r) \;=\; s \cdot C' + \mathsf{com}_{g,h}(0,r)$$

and that breaking the binding property of $\mathsf{com}_{C',h}$, which is equivalent to computing the discrete logarithm of C' with respect to h, is equivalent to opening C' to 0 as a commitment with respect to $\mathsf{com}_{g,h}$.

Lemma 7.1 *Let $C'' \in G$. Being able to open (wrt. $\mathsf{com}_{g,h}$) C' and C'' to values s' and s'', respectively, allows to open C'' wrt. $\mathsf{com}_{C',h}$ to a value s satisfying $ss' = s''$, and being able to open C'' to 0 wrt. $\mathsf{com}_{g,h}$ allows to open C'' to 0 wrt. $\mathsf{com}_{C',h}$.*

Proof. Let $s, s', s'', r', r'' \in \mathbb{F}_q$ satisfy $C' = \mathsf{com}_{g,h}(s', r')$, $C'' = \mathsf{com}_{g,h}(s'', r'')$ as well as $ss' = s''$. Then, for $r^* = r'' - sr'$ we have $\mathsf{com}_{g,h}(s'' - ss', r^*) = C'' \cdot C'^{-s}$ and therefore $\mathsf{com}_{C',h}(s, r^*) = C'^{s} \cdot \mathsf{com}_{g,h}(0, r^*) = C''$. This also holds if $s = 0$ and thus $s'' = 0$, in which case $r^* = r''$. □

This gives rise to the following commitment multiplication proof, which allows the prover to prove that he can open commitments C, C' and C'' to values s, s' and $s'' = ss'$, respectively. Note that the 4 steps can be executed in parallel.

PROTOCOL $\mathsf{MultProof}(C, C'; C'')$

1. The prover executes $\mathsf{Proof}_{g,h}(C)$.
2. The prover executes $\mathsf{Proof}_{C',h}(C'')$ using the *same* sharing polynomial $f_s(X)$ as in the above step (but new independent commitments with respect to $\mathsf{com}_{C',h}$).
3. Every player verifies whether his shares from step 1. and 2. coincide and accuses the dealer if it does not hold. In case $t < n/2$ (but not $t < n/3$) the dealer responds by opening the two corresponding commitments in public.
4. The prover executes $\mathsf{Proof}_{g,h}(C'')$.

This protocol also appeared in [Abe99]. However, the security proof given there did *not* cover the case where the prover can open C' to $s' = 0$, and therefore the protocol was extended to "also deal with the case $s' = 0$" by essentially adding another Pedersen VSS sharing. Our analysis shows that this is superfluous, and that the protocol as it stands is secure also in case $s' = 0$. Furthermore, we show that the case $s = 0$ is somewhat special. Namely, we show that if the prover can open the commitment C to $s = 0$, then he can execute the protocol even *without being able to open* C', as long as he can open C'' to $s'' = 0$. This of course also guarantees that $s'' = ss'$ (no matter what s' is), but, as we will see in the next section, it also opens the door for new constructions in this setting like proofs of partial knowledge.

Theorem 7.1 *The above protocol* $\mathsf{MultProof}(C, C'; C'')$ *is a perfectly sound zero-knowledge distributed-verifier proof, non-interactive in case* $t < n/3$ *and non-interactive with accusing in case* $t < n/2$*, that the prover can open* C*,* C' *and* C'' *to values* s*,* s' *and* $s'' = ss'$*,* or *that he can open both* C *and* C'' *to* 0*.*

Proof. Correctness: Follows from Lemma 7.1 and the correctness of the protocol $\mathsf{Proof}_{g,h}$.

Soundness: According to Proposition 7.1, from the information received during Step 1., the honest players can compute s and r with $C = \mathsf{com}_{g,h}(s, r)$. Also, from the information received during Step 2., the honest players can compute *the same* s and some r^* with $C'' = \mathsf{com}_{C',h}(s, r^*) = sC' + \mathsf{com}_{g,h}(0, r^*)$. Finally, from the information received during Step 3., the honest players can compute s'' and r'' with $C'' = \mathsf{com}_{g,h}(s'', r'')$. It now follows that either $s = 0$ and hence $C'' = \mathsf{com}_{g,h}(0, r^*)$, which means that the honest players can open C'' to zero, or that

$$\begin{aligned} C' &= (C'' - \mathsf{com}_{g,h}(0, r^*))/s \\ &= (\mathsf{com}_{g,h}(s'', r'') - \mathsf{com}_{g,h}(0, r^*))/s \\ &= \mathsf{com}_{g,h}(s''/s, (r'' - r^*)/s)\,, \end{aligned}$$

which means that the honest players can open C' to $s' = s''/s$.

Zero-Knowledge: The adversary can simulate his view of the protocol by simulating independently the protocols $\mathsf{Proof}_{g,h}(C)$, $\mathsf{Proof}_{C',h}(C'')$ and $\mathsf{Proof}_{g,h}(C'')$, as described in the proof of Proposition 7.1, *except* that he chooses the same shares for the simulation of $\mathsf{Proof}_{g,h}(C)$ and of $\mathsf{Proof}_{C',h}(C'')$. □

7.4.2 Proofs or Partial Knowledge

In [CDS94], an efficient solution was presented to construct proofs of *partial knowledge* in the two-players setting. Such a proof of partial knowledge allows for instance to prove the knowledge of (at least) ℓ out of m committed secrets without revealing *which* ℓ secrets. We will now present corresponding non-interactive protocols in the distributed-verifier setting. While the proof protocols of [CDS94] rely on concepts like the dual access structure and

the simulation of protocols, our distributed-verifier proof protocols are based on the fact that the commitment multiplication proof $\mathsf{MultProof}_{g,h}(C, C'; C'')$ can be executed by the prover even if he does not know s' (as long as $s = s'' = 0$).

Let $\mathsf{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \to G$ be an instantiation of Pedersen's commitment scheme, and let, to begin with, $C_0, C_1 \in G$ be two commitments (with respect to $\mathsf{com}_{g,h}$). Let $w \in \{0,1\}$. Assume that the proven can open C_w, i.e., that he knows s_w and r_w in $\mathbb{F}_q$ such that $C_w = \mathsf{com}_{g,h}(s_w, r_w)$. Consider the following protocol.

OR-Proof

The prover sets $b_w = 1$ and $b_{1-w} = 0$ as well as $d_w = s_w$ and $d_{1-w} = 0$, and he commits to b_0, b_1, d_0 and d_1 by B_0, B_1, D_0 and D_1, respectively. Then, he executes in parallel $\mathsf{MultProof}(B_0, C_0; D_0)$ and $\mathsf{MultProof}(B_1, C_1; D_1)$ such that $f_{b_0}(X) + f_{b_1}(X) = 1$, i.e. using sharing polynomials for b_0 and b_1 that add up to 1. The latter is locally verified by the players as part of the parallel executions of $\mathsf{MultProof}(B_0, C_0; D_0)$ and $\mathsf{MultProof}(B_1, C_1; D_1)$.

According to Theorem 7.1, the prover can execute $\mathsf{MultProof}_{g,h}(B_{1-w}, C_{1-w}; D_{1-w})$ even without being able to open C_{1-w} as long as he can open B_{1-w} and D_{1-w} to zero. On the other hand, consider a possibly corrupted prover. From the two proofs MultProof, the knowledge extractor can compute openings for B_0 and B_1 such that the resulting opening for $B = B_0 + B_1$ is 1. Hence, $B_{\tilde{w}}$ is not opened to zero for at least one $\tilde{w} \in \{0,1\}$. Therefore, the extractor can compute an opening for $C_{\tilde{w}}$ from $\mathsf{MultProof}_{g,h}(B_{\tilde{w}}, C_{\tilde{w}}; D_{\tilde{w}})$. Finally, the (perfect) hiding property of the commitment scheme and the zero-knowledge property of MultProof allow to show that the protocol is zero knowledge, in particular it reveals no information about w.

This can easily be generalized to ℓ-out-of-m proofs: Let $C_1, \ldots, C_m$ be a list of commitments. The following protocol allows the prover to prove in zero-knowledge that he can open (at least) ℓ of them. Let $E = \mathsf{com}_{g,h}(1,0)$ and $O = \mathsf{com}_{g,h}(0,0)$ be default commitments for 1 and 0, respectively.

ℓ-out-of-m-Proof

For $i = 1, \ldots, m$, the prover sets $b_i = 1$ and $d_i = s_i$ if he can open C_i (to s_i) and $b_i = d_i = 0$ otherwise, and he commits to b_i and d_i by B_i and D_i, respectively. He proves that for all i indeed $b_i \in \{0,1\}$ by executing $\mathsf{MultProof}(B_i, E - B_i; O)$. Finally, he executes $\mathsf{MultProof}(B_i, C_i; D_i)$ for all $i \in \{1, \ldots, m\}$ such that $\sum_i f_{b_i}(X) = \ell$ (which is verified by the players).

The following protocol is a general access structure proof, which allows the prover to prove in zero-knowledge that he can open *some* subset $\{C_{i_1}, \ldots, C_{i_l}\}$ out of the set of the

m commitments such that the subset belongs to a given access structure Γ on $\{1,\ldots,m\}$, i.e., $\{i_1,\ldots,i_l\} \in \Gamma$. The resulting ℓ-out-of-m proof when applied to the threshold access structure $\Gamma_{\ell-1,m}$ is somewhat more efficient than the above solution.

Let $\mathcal{M} = (\mathbb{F}_q, M, \psi, m)$ be a span program over $\mathbb{F}_q$ for Γ. We assume for simplicity that $d = m$, and that $\psi(i) = i$ for all $i \in \{1,\ldots,m\}$. Let $A \subseteq \{1,\ldots,m\}$ be the set that points to the commitments the prover can open, and let $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_m)^T \in \mathbb{F}_q^m$ be such that $M^T\boldsymbol{\lambda} = \boldsymbol{\varepsilon}$ and $\lambda_i = 0$ for $i \notin A$. Clearly, such a $\boldsymbol{\lambda}$ exists if and only if $A \in \Gamma$.

> Γ-Proof
>
> For $i = 1,\ldots,m$, the prover puts $b_i = \lambda_i$ and $d_i = b_i s_i$. Note that if $i \notin A$ then $b_i = d_i = 0$, independent of s_i. And, he generates commitments $B_1,\ldots,B_m$ and $D_1,\ldots,D_m$ for $b_1,\ldots,b_m$ and $d_1,\ldots,d_m$, respectively. Then he executes $\mathsf{MultProof}(B_i, C_i; D_i)$ for all i such that $M^T\mathbf{f} = \boldsymbol{\varepsilon}$ for $\mathbf{f} = (f_{b_1}(X),\ldots,f_{b_m}(X))^T$ (which is verified by the players).

Note that soundness of the above protocol relies on the binding property of the Pedersen commitment scheme, and hence only holds computationally.

Theorem 7.2 *Let $C_1,\ldots,C_m$ be a list of commitments, and let Γ be an access structure on $\{1,\ldots,m\}$. There exists a perfectly sound zero-knowledge distributed-verifier proof, non-interactive in case $t < n/3$ and non-interactive with accusing in case $t < n/2$, that the prover can open a subset $C_{i_1},\ldots,C_{i_\ell}$ of the commitments $C_1,\ldots,C_m$ for some set $A = \{i_1,\ldots,i_\ell\} \in \Gamma$.*

Concerning complexity, the distributed-verifier proof essentially consists of $3 \cdot \mathrm{msp}_{\mathbb{F}_q}(\Gamma)$ executions of Pedersen's VSS.

7.4.3 General Circuit Evaluation Proofs

Let $\mathcal{C}$ be a binary circuit consisting of NAND gates.

Theorem 7.3 *Under the DL-assumption, there exists a computationally sound zero-knowledge distributed-verifier proof, non-interactive in case $t < n/3$ and non-interactive with accusing in case $t < n/2$, that the prover knows a satisfying input to the circuit $\mathcal{C}$.*

Proof sketch. Let $\mathsf{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \to G$ be an instantiation of Pedersen's commitment scheme, set up once and for all e.g. by a trusted party or jointly by the verifiers. Let $b = (b_1,\ldots,b_m)$ be a satisfying input for the circuit $\mathcal{C}$. To prove knowledge of b, the prover generates a commitment B_i for every input bit b_i and proves that $b_i \in \{0,1\}$ by executing $\mathsf{MultProof}_{g,h}(B_i, E - B_i; O)$. Inductively, for every NAND gate with input

bits b_l and b_r to which he has already computed corresponding commitments B_l and B_r, respectively, the prover computes a commitment B_{out} for the output bit $b_{out} = b_l$ NAND b_r and proves its correctness by executing $\mathsf{MultProof}_{g,h}(B_l, B_r; E - B_{out})$. Finally, he opens the commitment of the circuit-output bit to 1. □

Note that in contrast to the well-known two-party non-interactive zero-knowledge proofs in the *common-reference-string model* [BFM88], where the simulation is with respect to a correctly distributed but *new* common-reference-string, the simulation of the distributed-verifier proof is with respect to the *given* public-key $pk = (g, h)$.

7.5 Concluding Remarks: General Adversaries

It is straightforward to generalize the results of this chapter to a general adversary. For that, simply replace Shamir's secret sharing scheme in Pedersen's VSS scheme by an secret sharing scheme $\mathcal{M} = (\mathbb{F}_q, M, \psi, n)$ for a general access structure Γ. The threshold conditions $t < n/2$ and $t < n/3$ translate to the Q^2 and Q^3 conditions, respectively.

Bibliography

[Abe99] Masayuki Abe. Robust distributed multiplication without interaction. In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[ACF02] Masayuki Abe, Ronald Cramer, and Serge Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In *Advances in Cryptology—ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*. Springer, 2002.

[BB89] Judit Bar-Ilan and Don Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *8th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 1989.

[BBDW96] Simon Blackburn, Mike Burmester, Yvo Desmedt, and Peter Wild. Efficient multiplicative sharing schemes. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Science*, 37(2), 1988.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*. Springer, 1991.

[Bea00] Donald Beaver. Minimal-latency secure function evaluation. In *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000.

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Technion, Haifa, 1996.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, 1988.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*. Springer, 1992.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1989.

[BGW88] Michael Ben-Or, Shari Goldwasser, and Avi Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, 1988.

[BI93] Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In *Advances in Cryptology—AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*. Springer, 1993.

[BI01] Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. In *Annual IEEE Conference on Computational Complexity*, volume 16, 2001.

[BKK87] Joan F. Boyar, Mark W. Krentel, and Stuart A. Kurtz. A discrete logarithm implementation of zero-knowledge blobs. Technical Report TR-87-02, Department of Computer Science, University of Chicago, March 1987.

[BL88] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*. Springer, 1988.

[Bla79] George R. Blakley. Safeguarding cryptographic keys. In *National Computer Conference '79*, volume 48 of *AFIPS Proceedings*, 1979.

[Bla87] Richard E. Blahut. *Priciples and Practice of Information Theory*. Addison-Wesley, 1987.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, 1990.

[Bri89] Ernest F. Brickell. Some ideal secret sharing schemes. In *Advances in Cryptology—EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*. Springer, 1989.

[BW86] Elwyn R. Berlekamp and Lloyd R. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[BW98] Donald Beaver and Avishai Wool. Quorum-based secure multi-party computation. In *Advances in Cryptology—EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*. Springer, 1998.

[Can01] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2001. Full version available from `http://eprint.iacr.org/2000/067.ps`.

[CCD88] David Chaum, Claude Crepeau, and Ivan Damgård. Multiparty unconditional secure protocols. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, 1988.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic or: Can zero-knowledge be for free? In *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*. Springer, 1999.

[CDF01] Ronald Cramer, Ivan Damgård, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *Advances in Cryptology—CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multiparty computation from any linear secret-sharing scheme. In *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000. Full version available from the authors.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*. Springer, 1994.

[CF02] Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. In *Advances in Cryptology—CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on the Theory of Computing (STOC)*, 1996.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient mutiparty computation over rings. In *Advances in Cryptology—EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakes, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1985.

[CK89] Beny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. In *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1989.

[CPS99] Sergio Cabello, Carles Padró, and German Sáez. Secret sharing schemes with detection of cheaters for a general access structure. In *12th International Symposium on Fundamentals of Computation Theory (FCT)*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1999.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Number ISBN 0-471-06259-6. John Wiley & Sons, Inc., 1991.

[DDB94] Yvo Desmedt, Giovanni Di Crescenzo, and Mike Burmester. Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In *Advances in Cryptology—ASIACRYPT '94*, volume 917 of *Lecture Notes in Computer Science*. Springer, 1994.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th Annual ACM Symposium on the Theory of Computing (STOC)*, 1994.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1989.

[DF94] Yvo Desmedt and Yair Frankel. Homomorphic zero-knowledge threshold schemes over any finite Abelian group. *SIAM Journal on Discrete Mathematics*, 7(4), 1994.

[DF99] Giovanni Di Crescenzo and Yair Frankel. Existence of multiplicative secret sharing schemes with polynomial share expansion. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. ACM Press, 1999.

[DKKK98] Yvo Desmedt, Brian King, Wataru Kishimoto, and Kaoru Kurosawa. A comment on the efficiency of secret sharing scheme over any finite Abelian group. In *4th Australasian Conference on Information Security and Privacy (ACISP)*, volume 1438 of *Lecture Notes in Computer Science*. Springer, 1998.

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally comopsable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—CRYPTO '03*, Lecture Notes in Computer Science. Springer, 2003.

[Dzi00] Stefan Dziembowski. *Multiparty Computations Information-Theoretically Secure Against an Adaptive Adversary*. PhD thesis, University of Århus, 2000.

[Feh99] Serge Fehr. Efficient construction of the dual span program. Manuscript, May 1999.

[Fel87] Paul Feldman. A practical scheme for noninteractive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1987.

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

[FHM99] Matthias Fitzi, Martin Hirt, and Ueli Maurer. General adversaries in unconditional multi-party computation. In *Advances in Cryptology—ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*. Springer, 1999.

[FM98] Matthias Fitzi and Ueli Maurer. Efficient Byzantine agreement secure against general adversaries. In *12th International Symposium on Distributed Computing (DISC)*, volume 1499 of *Lecture Notes in Computer Science*. Springer, 1998.

[FM02] Serge Fehr and Ueli Maurer. Linear VSS and distributed commitments based on secret sharing and pairwise checks. In *Advances in Cryptology—CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[FY92] Matthew Franklin and Moti Yung. Communication complexity of secure computation. In *24th Annual ACM Symposium on the Theory of Computing (STOC)*, 1992.

[Gál95] Anna Gál. *Combinatorial Methods in Boolean Function Complexity*. PhD thesis, University of Chicago, 1995.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd Annual ACM Symposium on Theory of Computing (STOC)*, 2001.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th Annual ACM Symposium on Theory of Computing (STOC)*, 1987.

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *17th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 1998.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zürich, 2001.

[HM97] Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *16th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 1997. Final version appeared in Journal of Cryptology 2000.

[HM01] Martin Hirt and Ueli Maurer. Robustness for free in unconditional multi-party computation. In *Advances in Cryptology—CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.

[HMP00] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*. Springer, 2000.

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

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*. Springer, 2002.

[ISN87] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structures. In *IEEE/IEICE Global Telecommunications Conference (GLOBECOM Tokyo '87)*. IEEE Computer Society Press, 1987.

[Kin00] Brian King. *Some Results in Linear Secret Sharing*. PhD thesis, University of Wisconsin-Milwaukee, 2000.

[Kin01] Brian King. Randomness required for linear threshold sharing schemes defined over any finite Abelian group. In *6th Australasian Conference on Information Security and Privacy (ACISP)*, volume 2119 of *Lecture Notes in Computer Science*. Springer, 2001.

[KW93] Maurizio Karchmer and Avi Wigderson. On span programs. In *8th Annual Conference on Structure in Complexity Theory (SCTC '93)*. IEEE, 1993.

[Lan97] Serge Lang. *Algebra, 3rd ed.* Addison-Wesley Publishing Company, 1997.

[Len77] Hendrik W. Lenstra. Euclidean number fields of large degree. *Inventiones Mathematicae*, 38, 1977.

[Mau03] Ueli Maurer. Secure multi-party computation made simple. In *3rd Conference on Security in Communication Networks (SCN '02)*, volume 2576 of *Lecture Notes in Computer Science*. Springer, 2003.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *10th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 1991.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*. Springer, 1991.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing (STOC)*, 1989.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11), 1979.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1997.

[Sho00] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*. Springer, 2000.

[Sim84] Gustavus J. Simmons. Authentication theory/coding theory. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*. Springer, 1984.

[Sim88] Gustavus J. Simmons. How to (really) share a secret. In *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*. Springer, 1988.

[Sti95] Douglas R. Stinson. *Cryptography — Theory and Practice*. Number ISBN 0-8493-8521-0. CRC Press, 1995.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1982.

[Yao86] Andrew C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986.

Index