Contextualizing Alternative Models of Secret Sharing

by

Shannon Veitch

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A secret sharing scheme is a means of distributing information to a set of players such that any authorized subset of players can recover a secret and any unauthorized subset does not learn any information about the secret. In over forty years of research in secret sharing, there has been an emergence of new models and extended capabilities of secret sharing schemes. In this thesis, we study various models of secret sharing and present them in a consistent manner to provide context for each definition. We discuss extended capabilities of secret sharing schemes, including a comparison of methods for updating secrets via local computations on shares and an analysis of approaches to reproducing/repairing shares. We present an analysis of alternative adversarial settings which have been considered in the area of secret sharing. In this work, we present a formalization of a deniability property which is inherent to some classical secret sharing schemes. We provide new, game-based definitions for different notions of verifiability and robustness. By using consistent terminology and similar game-based definitions, we are able to demystify the subtle differences in each notion raised in the literature.

Acknowledgements

I would like to thank my supervisor, Doug Stinson, for introducing me to research, for his support throughout my undergraduate and graduate degrees, and for his invaluable advice along the way. I am confident that as long as I continue to do research, I will be reminded of the values of clarity, creativity, and originality that he has instilled in me.

Thanks to Florian Kerschbaum, not only for taking the time to read this thesis, but also for his support throughout my degree. I am grateful to Alfred Menezes for his feedback on this thesis. I would also like to extend thanks to Douglas Stebila and Matt Borland for their guidance and support of my future endeavours.

Thank you to the members of the CrySP lab for the friendships and for always offering the most welcoming space. Special thanks to Bailey, Navid, Anna, Thomas, Rasoul, Lindsey, and Jason. I would also like to thank members of the Critical Media Lab and AAC for helping me maintain perspective of my work, and especially Alexi for her friendship.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A secret sharing scheme is a means of distributing information to a set of players such that any authorized subset of players can recover a secret and any unauthorized subset does not learn any information about the secret. Secret sharing was proposed independently by Blakley [19] and Shamir [133] in 1979. In over forty years of research on the topic of secret sharing, we have witnessed the emergence of new models and extended capabilities of secret sharing schemes.

Secret sharing has long been an important building block in the area of threshold cryptography. This includes cryptographic algorithms where a threshold number of participants are required to work together to perform some operation (such as decryption or creating digital signatures). Using threshold schemes enables a distribution of trust among a set of participants. With applications in multi-party computation and distributed data storage, models of secret sharing have been adapted over time to facilitate new goals.

The newly proposed models of secret sharing include additional capabilities beyond what was required in the original definitions of secret sharing. They also consider more complex adversarial settings. Some models were proposed to apply to specific applications whereas others were proposed as extensions of notions in different areas of work, such as coding theory or distributed computation. Motivations for new models originating from different areas of work has resulted in some overlapping and/or contradictory definitions. At the same time, some work has adopted unnecessarily complex and cumbersome notation which makes it difficult to compare new work with existing notions that achieve similar goals. To address these problems, this thesis aims to provide context for these new models by presenting and analyzing them in a cohesive manner.

This thesis includes an analysis of extended capabilities of secret sharing schemes and

alternative adversarial settings. We organize previous work and bring attention to some redundancies in the literature. We present new, readable, game-based definitions for many of the notions which we discuss. These new definitions enable easier comparison of the concepts which have been explored in the area of secret sharing. Our goal in this work is to present models of secret sharing in a consistent manner to highlight the benefits and drawbacks of each alternative model. This brings attention to potential directions of future work and demystifies the subtle differences in closely related work. Throughout this work, the key contributions are as follows:

- An analysis of extended capabilities of secret sharing schemes. This portion of the thesis includes but is not limited to:
  - A thorough review and comparison of methods of updating secrets without the presence of a dealer.
  - An analysis of methods for reproducing/repairing shares and a comparison of each approach.
- An analysis of alternative adversarial settings which have been considered in the literature. This portion of the thesis includes but is not limited to:
  - Formal definitions of four notions of verifiability and a comparison of each approach.
  - Formalization of a deniability property which is inherent to some classical secret sharing schemes.
  - Streamlined definitions of seven notions of robustness and a comparison of each approach.

Chapter 2

Background

A secret sharing scheme is a cryptographic primitive for splitting a secret into some shares and distributing the shares amongst a set of participants such that only an authorized subset of participants can reconstruct the secret. Constructions for secret sharing schemes were introduced independently in 1979 by Shamir [133] and Blakley [19]. Blakley's construction relies on finite geometries while Shamir's scheme uses polynomial interpolation to reconstruct secrets.

Secret sharing schemes were initially considered in an *information-theoretic* (or *unconditionally secure*) setting. This setting ensures that the security guarantees hold regardless of the computational capabilities of the adversary. The goals of secret sharing schemes in the information-theoretic setting are twofold: correctness and perfect privacy.

Correctness: Any authorized set of parties can reconstruct the secret.

Perfect Privacy: Any unauthorized set of parties learns *no information* about the secret from their shares.

A majority of this thesis will be focused on *threshold schemes*. In a (k, n)-threshold scheme, a dealer breaks a secret into n shares and distributes them amongst n participants such that any k of the participants can collaborate to reconstruct the secret. Any coalition of $k-1$ participants learns no information about the secret. The prototypical example of a threshold scheme in an information-theoretic setting is Shamir's scheme [133].

Construction 2.0.1. [Shamir's Secret Sharing Scheme [133]] We present a (k, n)-threshold scheme to share a secret $s \in \mathbb{F}_q$ where $q \geq n+1$. Here $\mathbb{F}_q$ is a finite field where q is a prime or a prime power.

Share: The dealer selects a random polynomial $r(x) \in \mathbb{F}_q[x]$ of degree $k-1$ such that $r(0) = s$. Each share is a coordinate $s_i = (x_i, y_i)$, where the x_i's are distinct and non-zero. The dealer gives share s_i to participant P_i for $i = 1, \ldots, n$.

Recover: Given k shares, the participants use polynomial interpolation to reconstruct $r(x)$ and then evaluate the polynomial at $x = 0$ to recover the secret s.

■

In Shamir's scheme, we make use of polynomial interpolation to reconstruct the secret. In particular, we can use the *Lagrange interpolation formula* to reconstruct the polynomial. Suppose we are working in a field $\mathbb{F}_q$ and are given k (not necessarily distinct) elements in $\mathbb{F}_q$, say $a_1, \ldots, a_k$. Let $x_1, \ldots, x_k$ be distinct elements in $\mathbb{F}_q$. Then there is a unique polynomial $A(x) \in \mathbb{F}_q[x]$ with degree at most $k-1$ such that $A(x_i) = a_i$ for $1 \leq i \leq k$. The *Langrange interpolation formula* states that

$$A(x) = \sum_{j=1}^{k} a_j \prod_{1 \leq h \leq k, h \neq j} \frac{x - x_h}{x_j - x_h}.$$

When reconstructing secrets with Shamir's scheme, we are typically concerned with the evaluation of this polynomial at 0. In this case, it is sufficient to compute

$$s = \sum_{j=1}^{k} a_j \prod_{1 \leq h \leq k, h \neq j} \frac{x_h}{x_h - x_j}.$$

Now, if we define

$$b_j = \prod_{1 \leq h \leq k, h \neq j} \frac{x_h}{x_h - x_j},$$

for $1 \leq j \leq k$, then we can write $s = \sum_{j=1}^{k} b_j a_j$. These terms b_j are called the *Lagrange coefficients* and they are publicly known values.

Another popular construction of a classical secret sharing scheme is the *additive secret sharing scheme*.

Construction 2.0.2. [Additive Secret Sharing Scheme] We present an (n, n)-threshold scheme to share a secret $s \in \mathbb{Z}_q$ amongst a set of n participants. Note that q does not have to be prime.

Share: The dealer selects $n-1$ shares, say $s_1, \ldots, s_{n-1}$, at random from $\mathbb{Z}_q$ and sets $s_n = s - \sum_{i=1}^{n-1} s_i$. The dealer gives s_i to participant P_i for $1 \leq i \leq n$.

Recover: To recover s, the participants take the sum of all the shares modulo q.

■

Typically, it is of interest to minimize the size of the shares relative to the size of the secret, to reduce the amount of storage required for a given scheme. For a secret sharing scheme in which $\mathcal{S}$ is the set of possible secrets and $\mathcal{T}$ is the set of possible shares, we define the *information rate* to be $\rho = \log|\mathcal{T}|/\log|\mathcal{S}|$. Here and in what follows, we use log to denote $\log_2$. We say that a secret sharing scheme is *ideal* if it achieves perfect privacy and has an information rate equal to one. That is, the size of the domain of shares is exactly the size of the domain of the secret. Brickell [36] elaborates on ideal secret sharing schemes.

Given that much of the work in the area of secret sharing has been done in an information-theoretic setting, it is useful to review some basic concepts from information theory. In the remainder of this section, we present some definitions from information theory that we will use later on. Let X and Y be two random variables.

- The Shannon *entropy* of X is

$$H(X) = -\sum_{x \in X} \Pr[X = x] \log \Pr[X = x].$$

- The *joint entropy* $H(X, Y)$ of X and Y is

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} \Pr[X = x, Y = y] \log \Pr[X = x, Y = y].$$

- The *conditional entropy* $H(X \mid Y)$ of X given Y is defined as

$$H(X \mid Y) = \sum_{y \in Y} \Pr[Y = y] H(X \mid Y = y),$$

  where

$$H(X \mid Y = y) = -\sum_{x \in X} \Pr[X = x \mid Y = y] \log \Pr[X = x \mid Y = y].$$

For random variables X, Y, we have the following identity: $H(Y \mid X) = H(X, Y) - H(X)$.

Let $\mathcal{A}$ be an access structure on the set of participants $\mathcal{P}$ and suppose $\{p_{\mathcal{S}}(s)\}_{s \in \mathcal{S}}$ is a probability distribution on the set of secrets $\mathcal{S}$. A secret sharing scheme is a sharing of a secret in $\mathcal{S}$ among participants in $\mathcal{P}$ such that for any qualified subset $A \in \mathcal{A}$, $H(\mathcal{S} \mid A) = 0$, and for any non-qualified subset $A \notin \mathcal{A}$, $H(\mathcal{S} \mid A) = H(\mathcal{S})$.

We also consider the dealer's randomness of a secret sharing scheme Σ, given that the probability distribution on the set of secrets $\mathcal{S}$ is $\Pi_{\mathcal{S}} = \{p_{\mathcal{S}}(s)\}_{s \in \mathcal{S}}$. The dealer's randomness was defined in [24] to be

$$\mu(\mathcal{A}, \Pi_{\mathcal{S}}, \Sigma) = H(P_1, \ldots, P_n \mid \mathcal{S}),$$

where P_i denotes participant i. That is, $H(P_1, \ldots, P_n)$ is the entropy of the probability space from which the shares to be given to the participants are taken.

The authors in [25] consider a secret sharing scheme with the minimum possible amount of randomness for a given access structure $\mathcal{A}$. This was defined to be

$$\mu(\mathcal{A}, q) = \inf_{\mathcal{Q}, \mathcal{T}} \mu(\mathcal{A}, \Pi_{\mathcal{S}}, \Sigma)$$

where $\mathcal{Q}$ is the space of all probability distributions on a set $\mathcal{S}$ of q secrets and $\mathcal{T}$ is the space of all secret sharing schemes Σ for the access structure $\mathcal{A}$. It was also shown in [25] that for an access structure $\mathcal{A}$ on a set $\mathcal{P}$, supposing the secret is chosen in $\mathcal{S}$, if there exists an independent sequence of participants of length m then

$$\mu(\mathcal{A}, |\mathcal{S}|) \geq m \cdot \log |\mathcal{S}|.$$

Chapter 3

Extended Capabilities of Secret Sharing Schemes

Since its introduction, there has been a large body of work in adapting secret sharing schemes to scenarios where there is a desire for some distribution of trust. Throughout this development, classical secret sharing schemes have been extended beyond the basic properties of correctness and privacy. In this chapter, we discuss some of the most prominent examples of extended capabilities of secret sharing schemes.

3.1 What's in a secret?

In the previous chapter, we described secret sharing schemes where the secret may be represented by a scalar value or coordinate. This format makes it possible to share items such as cryptographic keys, passwords, and numerical values. In general, as long as the data to be shared can be encoded to a value in a finite field, we can use the classical constructions. For example, to share a large text file, one could use an encoding function which divides the file into multiple pieces, each representable by a value in a reasonably-sized finite field. Secret sharing has been extended to enable the sharing of more complex structures and data types, including but not limited to multiple non-independent secrets and functions.

3.1.1 Multi-secret sharing

It is natural to consider a scenario in which one wishes to share multiple secrets, say $s_1, \ldots, s_m$. Suppose each secret corresponds to an access structure, $\mathcal{A}_1, \ldots, \mathcal{A}_m$, where each $\mathcal{A}_i \subset 2^{\mathcal{P}}$ and $2^{\mathcal{P}}$ denotes the set of all subsets of $\mathcal{P}$. This goal is addressed by *multi-secret sharing* [87, 22]. Trivially, sharing multiple secrets can be accomplished by instantiating m sharings of the secrets $s_1, \ldots, s_m$, where each secret sharing is setup to correspond to a particular access structure $\mathcal{A}_i$. In this case, the size of the share would be m times the size of a share corresponding to a single secret. The challenge in multi-secret sharing schemes is to develop more efficient constructions which reduce the size of the shares.

Example 3.1.1. [Sharing Three Secrets [22]] Consider a setting with 4 participants $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ and three secrets $\mathcal{S} = \{s_1, s_2, s_3\}$, chosen uniformly and independently, where $s_i \in \{0, 1\}$ for all i. The construction will satisfy the following access structures,

- $\mathcal{A}_1 = \{\{P_1, P_2\}, \{P_2, P_3\}, \{P_2, P_4\}\}$,
- $\mathcal{A}_2 = \{\{P_1, P_3\}, \{P_2, P_3\}, \{P_3, P_4\}\}$,
- $\mathcal{A}_3 = \{\{P_1, P_4\}, \{P_2, P_4\}, \{P_3, P_4\}\}$,

where $\mathcal{A}_1$ is the access structure for s_1, $\mathcal{A}_2$ is the access structure for s_2, and $\mathcal{A}_3$ is the access structure for s_3. Note that with these access structures, the set of participants $\{P_2, P_3, P_4\}$ can recover all three secrets. The dealer chooses three bits $a, b, c \in \{0, 1\}$, uniformly and independently at random, and distributes the shares as follows:

- P_1 gets $a \oplus b \oplus c$,
- P_2 gets $a \oplus s_1, b, c$,
- P_3 gets $a, b \oplus s_2, c$, and
- P_4 gets $a, b, c \oplus s_3$.

In this construction, all shares make up ten bits of information, as opposed to the twelve that would be used if three instantiations of a classical scheme were used. ■

3.1.2 Visual secret sharing

A visual secret sharing scheme for n participants encodes an image into n image shares such that an authorized set of shares can be combined to recover the original image and an unauthorized set reveals no information about the secret image. The basic idea consists of printing images on transparent sheets called "transparencies" that can be stacked upon one another to reveal the secret.

Naor and Shamir introduced the original concept of visual cryptography [113] and provided a construction for black and white images. The idea is to construct binary matrices (where ones represent black pixels and zeros represent white pixels) such that shares of a white pixel, when stacked upon one another, have a smaller Hamming weight than shares of a black pixel. A single share appears to be a random choice of a matrix. The pixels with a larger Hamming weight will appear darker as the transparencies are stacked on top of one another.

Example 3.1.2. [(2,2)-visual threshold scheme] For each pixel in the original image, we expand the pixel into a share which contains 4 pixels. The shares are represented by one of the following six matrices:

$$\left\{ \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}.$$

Notice that the first two matrices are complementary, as are the third and fourth, and the final two. A white pixel is shared into two identical matrices from the list and a black pixel is shared into two complementary matrices. A single share is then a random matrix containing two black and two white pixels. When stacked upon one another, the matrices either have a Hamming weight of two (representing white) or a Hamming weight of four (representing black). ■

Later work by Blundo, De Bonis, and De Santis [20] improved on the pixel expansion of original constructions and provided constructions for coloured images. To generalize the constructions to coloured images, we use matrices with subpixels in the set $\{0, 1, \ldots, c\}$, where each value corresponds to a different colour. When stacking subpixels on top of one another, if a sufficient number of the subpixels are of colour i, then it appears as colour i; otherwise, the stacked pixels appear to be black. A pixel reconstructs to the colour i if and only if there are sufficiently many subpixels of colour i.

3.1.3 Function secret sharing and DPFs

Function secret sharing [30, 32] considers the problem of secret sharing a function between multiple parties. This presents more complications than sharing a scalar value since it introduces certain requirements between the shares. The most basic form of function secret sharing is called *distributed point functions* (DPFs) [70].

Definition 3.1.1 (Distributed point function (DPF) [70])**.** *A* point function $f_{x,y}$ *for* $x, y \in \{0,1\}^n, n \in \mathbb{N}$*, evaluates to* y *on input* x *and to 0 on all other inputs. A* distributed point function *is a family of functions such that, given* $f_{x,y}$*, we can find functions* f_{k_0} *and* f_{k_1} *such that* $f_{k_0}(a) \oplus f_{k_1}(a) = f_{x,y}(a)$ *on every input* a*, and individually, each* f_{k_i} *reveals no information about* x *and* y*.*

Essentially, a DPF allows a dealer to split a point function into two function shares that reveal nothing about the original function themselves. We note that this is not as simple as secret sharing the values x and y via classical methods, since the corresponding shares would not necessarily satisfy the requirement that $f_{k_0} \oplus f_{k_1} = f_{x,y}$

Construction 3.1.1. [Naive DPF] Let k_0 be the truth table of a random function $f_{k_0} : \{0,1\}^{|x|} \to \{0,1\}^{|y|}$ and let k_1 be the truth table of the function $f_{k_1} = f_{k_0} \oplus f_{x,y}$. This is perfectly secure since k_0 and k_1 are random. The drawback of this construction is that the size of k_0 and the size of k_1 is exponential in the size of the input. ■

Gilboa and Ishai [70] provide a construction of a DPF with polynomial key size under the assumption that a one-way function exists. This work is done in a computational setting, since it requires the existence of a one-way function.

Function secret sharing is the natural generalization of DPFs to more than two shareholders. A function secret sharing (FSS) scheme splits a function $f : \{0,1\}^n \to \mathbb{G}$, for some group $\mathbb{G}$, into p functions, $f_1, \ldots, f_p$, such that $\sum_{i=1}^{p} f_i = f$ on every input, and any strict subset of the keys reveals no information about f. The natural extension of the naive DPF construction is to additively share the truth table of f. There exist more efficient constructions of function secret sharing in the computational setting based on pseudorandom functions [30]. Function secret sharing and DPFs are particularly useful in constructing private information retrieval schemes [47, 66].

3.2 Updating secrets

There has been ample previous work discussing constructions of secret sharing schemes in which the shares corresponding to some secret, s, can be updated such that reconstruction

using the new shares results in a related, but different secret s'. In other words, the shareholders are able to reconstruct some $s' = f(s)$ for some function f. This is the basis of multi-party computation. We demonstrate this property using Shamir's scheme.

Example 3.2.1. [Updating a Secret in Shamir's Scheme] The dealer selects a random polynomial $r(x) \in \mathbb{Z}_q[x]$ of degree $k-1$ such that $s = r(0)$ is the secret. Each share is a coordinate $v_i = (x_i, y_i)$, where the x_i's are distinct and non-zero. Suppose the dealer would like to update the secret to $r(0) + c$ for some $c \in \mathbb{Z}_q[x]$. The dealer broadcasts c to all shareholders. For each $i = 1, \ldots, n$, shareholder $\mathcal{P}_i$ adds c to their share v_i. Following reconstruction, the new secret will be $s' = s + c$. ■

There are several properties to note in the previous example. First, the dealer is involved in updating the shares and the message that they broadcast reveals some information about the original secret once the new secret is reconstructed. Additionally, the shareholders do not have to collaborate in order to update their shares. These properties allude to the various ways in which we can update secrets. In the following subsections, we describe different methods used to update secrets. We also discuss some homomorphic properties required to allow shareholders to compute functions on secrets via local updates to their shares.

3.2.1 Updates with a dealer

One method of updating secrets involves a dealer sending a broadcast message to all participants. Different broadcast messages can enable the reconstruction of different secrets. We will discuss two types of secret sharing schemes which allow a dealer to update secrets via a broadcast message: prepositioned schemes and fully dynamic schemes.

In a prepositioned scheme [134] the shareholders are unable to recover any secret after the initial sharing. This is because the dealer withholds some information until a later point in time. Once the dealer reveals additional information, the shareholders can then recover a secret. The concept is related to updating secrets because the dealer can initially share some information, decide which secret they would like to recover at a later point in time, and broadcast some small amount of information corresponding to the secret they have chosen. These updates may also correspond to different access structures. Consider the following example, which adapts Shamir's scheme to become a prepositioned scheme.

Example 3.2.2. [(k, n)-threshold prepositioned scheme [53]] The dealer selects a random polynomial $r(x) \in \mathbb{Z}_q[x]$ of degree $k-1$ such that $s = r(x_0)$ is the secret. Note that x_0

may not equal 0. As in Shamir's scheme, each share is a coordinate (x_i, y_i), where the x_i's are distinct, non-zero, and not equal to x_0. A threshold number of participants can recover $r(x)$ but do not know the value of the secret until they are given x_0, which the dealer can broadcast at a later time. ■

In the prepositioned scheme described above, the dealer can broadcast multiple x-coordinates corresponding to different secrets if they wish to do so, resulting in different reconstructed secrets each time. A similar strategy is used in *fully dynamic secret sharing schemes* [21]. In addition to allowing participants to reconstruct different secrets, fully dynamic schemes enable different access structures to be activated, which we discuss later in Section 3.4. Some fully dynamic schemes satisfy the same properties as prepositioned schemes. Consider the following example of a fully dynamic scheme, in which the secrets correspond to different access structures.

Example 3.2.3. [Fully dynamic scheme [21]] Let the set of participants be $\mathcal{P} = \{P_1, P_2, P_3\}$ and the set of possible secrets be a finite field $\mathbb{Z}_q$. Let $s_1, s_2 \in \mathbb{Z}_q$ be two independently chosen secrets, and $\mathcal{A}_1 = \{\{P_1, P_2\}\}$, $\mathcal{A}_2 = \{\{P_2, P_3\}\}$ be the two access structures corresponding to the secrets, respectively.

For $i = 1, 2, 3$, the dealer randomly selects $a_i \in \mathbb{Z}_q$ and sends a_i to P_i. These are the shares. To activate $\mathcal{A}_1$, the dealer computes $b_1 = a_1 + a_2 + s_1 \bmod q$ and broadcasts b_1 to the shareholders. Similarly, to activate $\mathcal{A}_2$, the dealer computes $b_2 = a_2 + a_3 + s_2 \bmod q$ and broadcasts b_2 to the shareholders. ■

In the previous construction, suppose b_1 is broadcast so that P_1 and P_2 reconstruct s_1. Later, the dealer may broadcast b_2. Then, P_1, who knows s_1, can learn a_2 and reconstruct s_2 with P_3, even though $\{P_1, P_3\} \notin \mathcal{A}_2$. Therefore, a dealer must be careful to note whether or not they are revealing additional information when broadcasting information.

3.2.2 Dealer-free updates

Now we consider a setting in which we no longer have a trusted dealer after the scheme is initialized. Certain secret sharing schemes have homomorphic properties which enable shareholders to modify their shares to obtain shares of an updated secret without reconstructing the original secret. We will review some types of secret sharing schemes which possess these properties, enabling shareholders to update the secret without the presence of a dealer. Trivially, a single shareholder could act as a "dealer" and broadcast messages

as previously discussed. In this section, we consider schemes where all shareholders have equal responsibility.

A secret sharing scheme is said to be *linear* if the secret is an element from a finite field and can be computed as a linear combination of any set of authorized shares (and some independent random field elements). For example, in Shamir's scheme the secret can be written as a linear combination of the shares where the constant terms are the publicly known Lagrange coefficients. Suppose a set of shareholders have shares corresponding to a secret, a, shared via a (k_a, n)-threshold scheme, and shares corresponding to a secret, b, shared via a (k_b, n)-threshold scheme. Let c be some publicly known value. With a linear secret sharing scheme, it is easy to update the secret to the following values via local computations on the shares:

- $a + b$: If every shareholder takes the sum of their two shares, the result is a sharing of $a + b$ where the new threshold is $\max\{k_a, k_b\}$.
- $c \cdot a$: If every shareholder multiplies their share of a by c, the result is a sharing of $c \cdot a$ and the threshold remains unchanged.

The two techniques described above for updating shares are easily accomplished with linear secret sharing schemes. The next function which naturally comes to mind is the ability for the shareholders to obtain a sharing of the product $a \cdot b$. Multiplication is slightly more complex and not necessarily afforded by the properties of a linear secret sharing scheme. For example, using Shamir's scheme, the shareholders could multiply their two shares together to obtain a share of $a \cdot b$; however, the threshold value would also increase and the resulting polynomial would not be random due to the fact that it can be factored into two smaller polynomials. This has been pointed out in several previous works [116, 13]. There are three main categories of approaches to obtaining the product of secrets:

1. An interactive approach in the information-theoretic model.
2. A non-interactive approach in the information-theoretic model. Using this approach, we either obtain a scheme in which multiplication or addition is simple, but not both.
3. A non-interactive approach in the computational model. Using this approach, we can achieve both multiplication and addition of secrets.

First, we present an interactive protocol between shareholders to obtain a sharing of $a \cdot b$ when using Shamir's scheme. Despite the fact that the following protocol is well-known

within cryptographic folklore, it appears to be difficult to find a self-contained description of the protocol in the literature. Therefore, we present a concise description of the protocol here, which is based on a combination of techniques that are described in [116]. We assume the following properties about the participants:

- All shareholders are honest but curious, i.e., they do not deviate from the protocol but may attempt to infer information from what they are given.
- At most $k-1$ shareholders can collude with one another and share information during the execution of the protocol.

Construction 3.2.1. [Computing the product of two secrets in Shamir's scheme] Let k, n be integers such that $n \geq 2k-1$. Assume there are n shareholders and that there exists a secure channel between any two players. Assume that the dealer is honest and only active during the initialization step.

1. **Initialization:** The dealer uses two polynomials of degree $k-1$, say $f(x)$ and $g(x)$, to share two secrets, a and b. The corresponding shares are $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$.
2. **Local multiplication:** Each shareholder P_i receives shares s_i and t_i from the dealer and computes $u_i = s_i \cdot t_i$. The old shares are now erased.

   The values $u_1, \ldots, u_n$ are shares of the secret $a \cdot b$ corresponding to the polynomial $h(x) = f(x) \cdot g(x)$. Since $h(x)$ has degree $2k-2$, we effectively have a $(2k-1, n)$-threshold scheme at this point; however, the sharing polynomial $h(x)$ is not random.
3. **Randomization:** We now randomize the sharing polynomial $h(x)$ without changing the secret $a \cdot b$. First, k of the players are chosen, say $P_1, \ldots, P_k$. Each of these players independently creates shares of 0 using a polynomial of degree $2k-2$ and distributes the shares to all the players (including themselves) using secure channels. Each player P_i then adds the k shares they received to their original share u_i to create a new share v_i. All "old" shares are then erased. Now we have a secure sharing of $a \cdot b$ for a $(2k-1, n)$-threshold scheme.
4. **Threshold reduction:** We now apply a degree reduction technique to decrease the threshold from $2k-1$ to k, using the "Lagrange method" described in Section 2.1 of [116]. A set, Δ, of $2k-1$ shareholders is chosen (at random). Each shareholder $P_i \in \Delta$ selects a random polynomial $w_i(x)$ of degree at most $k-1$ such that $w_i(0) = v_i$. They give $w_i(j)$ to participant P_j for $1 \leq j \leq n$.

The following public values are computed:

$$\gamma_i^{\Delta} = \prod_{j \in \Delta, j \neq i} \frac{j}{j-i} \text{ for all } i \in \Delta.$$

Each player P_j erases their old shares and computes the new shares as follows:

$$\phi_j = \sum_{i \in \Delta} (\gamma_i^{\Delta} \times w_i(j)).$$

■

In the randomization step, we require that k participants perform the resharing of new values since we assumed that $k-1$ participants may be sharing information with one another. An alternative method of threshold reduction uses the Vandermonde method [13], but it requires that all n participants generate and reshare values, as opposed to the $2k-1$ in the Lagrange method above. All known methods of multiplying secrets via computations on the shares for Shamir's scheme require interaction between participants.

Now we discuss an alternative approach to multiplication of secrets which can be more efficient for small parameters than the approach using Shamir's scheme. As with the previous approach, this protocol is interactive and unconditionally secure. Rather than using Shamir's scheme, this protocol uses *replicated secret sharing* [85]. The following approach to obtaining the product of secrets is often used in multi-party computation and applications in machine learning due to its improved efficiency [6, 144]. In particular, we discuss how to obtain the product of two secrets for $(2,3)$-threshold schemes. In a $(2,3)$ replicated secret sharing scheme, each participant receives two shares from a $(3,3)$ additive secret sharing scheme (recall the definition of additive secret sharing given in Chapter 2). Given shares corresponding to two secrets, x and y, say (x_1, x_2) and (y_1, y_2), a participant can locally compute the following combination of their shares: $x_1y_1+x_2y_1+x_1y_2$. This is a share of $x \cdot y$ under a $(3,3)$ additive secret sharing scheme. Participants can then interactively perform a threshold reduction by sending each other shares in such a way that all participants end up with shares of $x \cdot y$ under a replicated $(2,3)$-threshold scheme. This approach can be generalized to larger parameters; however, there is the drawback of having to store as many shares as the threshold number of shareholders. This also implies that when performing additions of secrets, the computation requires a factor of k times the local addition operations. Nonetheless, the multiplication procedure is faster than using Shamir's scheme because it does not require the polynomial interpolation computations. The appropriate choice of which protocol to use depends on the parameters

of the threshold scheme and the application's constraints with respect to storage and computational capabilities.

It is possible to obtain the product of secrets without interaction between participants when working in the information-theoretic model; however, this requires a modification to the secret sharing scheme which removes the ability to easily compute the sum of secrets. Benaloh [14] provides a construction using discrete logarithms. The main observation is that the sum of the shares of the discrete logarithms of the secrets are the shares of the discrete log of the product of the secrets. In sharing the discrete log of the secret, we obtain a protocol where the product of two secrets is easy computed by locally taking the sum of shares. In general, discrete logarithms are difficult to compute so we must work modulo some small prime p or by using a special form where discrete logs are easier to compute [125, 2]. This does not affect the security of the scheme and it is still unconditionally secure. The drawback of this approach is that the sum of secrets is no longer simple to compute, since the scheme is not linear.

Naturally, it would be nice to have a protocol in which it is possible to compute the sum and product of two secrets without interaction between participants. This is possible in the computational model and discussed in recent work on so-called *homomorphic secret sharing* (HSS) [31, 33]. Essentially, a secret sharing scheme is called *homomorphic* if it is possible to perform local computations on shares (i.e., non-interactive protocols) which result in shares that sum to a sum or product of secrets (or a more generic function of secrets). The core idea of these constructions is to construct shares which have both additive and multiplicative encodings. The sum of shares in the additive encoding results in shares of the sum of the secrets. The sum of shares in the multiplicative encoding results in shares of the product of the secrets. There also must exist operations which can be computed locally on the shares that convert the shares between the two types of encodings. The security of these schemes rely on computational assumptions [31, 34].

These four main approaches to obtaining the product of secrets without the presence of a trusted dealer after initialization are summarized in Table 3.1.

	Non-Interactive	Multiplicative & Additive	Unconditionally Secure
Shamir		✓	✓
Replicated SS		✓	✓
Benaloh	✓		✓
HSS	✓	✓	

Table 3.1: Comparison of methods for obtaining the product of two secrets without a dealer

3.3 Reproducibility

The next property we discuss is reproducibility: the ability to recompute lost or corrupted shares. The term *reproducibility* and *reproducible secret sharing* was introduced in [11]. Given enough information, the dealer and/or the shareholders can reproduce lost shares. When accomplished by the shareholders, the same property is often called *repairability*. In this section, we discuss various methods of repairing/reproducing shares and use the terms repairability and reproducibility interchangeably.

3.3.1 Reproducibility via the dealer

Classical constructions of secret sharing schemes are probabilistic, i.e., they generate fresh randomness for each sharing. The security of the scheme relies on this property. Therefore, to be able to reproduce shares in an information-theoretic setting, the dealer must retain the randomness used in share construction or at least some information of equivalent entropy. In *adept secret sharing* [11], Bellare, Dai, and Rogaway consider the problem of reproducibility in a computational setting. By leveraging the properties of pseudo-random functions, their construction requires that the dealer retain only a small amount of randomness.

To reproduce every share in an information-theoretic setting, the dealer must store all of the randomness and the secret. Intuitively, this is obvious because if a dealer could reconstruct every share, they could also reconstruct the secret and the randomness. Thus, the entropy values must be equivalent. More formally, recall the definition of dealer's randomness from Chapter 2: $\mu(\mathcal{A}, \Pi_{\mathcal{S}}, \Sigma) = H(P_1, \ldots, P_n \mid \mathcal{S})$. We denote the dealer's randomness by $H(R)$. We know that $H(R) = H(P_1, \ldots, P_n \mid \mathcal{S})$ by definition, so we just need to show that $H(\mathcal{S}) = H(P_1, \ldots, P_n \mid R)$ as well. As a reminder from Chapter 2, the equation $H(Y \mid X) = H(X, Y) - H(X)$ holds for random variables X, Y. Therefore,

$$
\begin{aligned}
&H(P_1, \ldots, P_n \mid R) \\
&= H(P_1, \ldots, P_n, R) - H(R) \\
&= H(P_1, \ldots, P_n, R) - H(P_1, \ldots, P_n \mid \mathcal{S}) \\
&= H(P_1, \ldots, P_n, R) - H(P_1, \ldots, P_n, \mathcal{S}) + H(\mathcal{S}) \\
&= H(P_1, \ldots, P_n) + H(R \mid P_1, \ldots, P_n) - H(P_1, \ldots, P_n) - H(\mathcal{S} \mid P_1, \ldots, P_n) + H(\mathcal{S}) \\
&= H(R \mid P_1, \ldots, P_n) - H(\mathcal{S} \mid P_1, \ldots, P_n) + H(\mathcal{S}) \\
&= 0 - 0 + H(\mathcal{S}) = H(\mathcal{S}).
\end{aligned}
$$

This result is not surprising, but it allows us to conclude that, in order to reproduce shares from some combination of the dealer's randomness and the secret, all of the randomness and the secret must be stored. As demonstrated in [11], we can reduce the amount of information that the dealer must store by working in the computational setting. This form of reproducibility comes at the cost of diminished privacy for low-entropy secrets.

For example, to reproduce shares in Shamir's scheme, the dealer can retain the polynomial used to generate shares. Alternatively, they could hold something of equivalent entropy such as the secret and the random values used to generate the polynomial. There is also the possibility that the dealer has some combination of the shares and the randomness, which constitute equivalent entropy to holding the secret and the randomness. This would equally enable them to restore all shares. Given any less information, they could not regenerate all of the shares.

A dealer could store less information while still being able to reproduce some, but not all, shares. Trivially, the dealer could store a subset of shares which would allow them to reproduce those shares but not others. In Shamir's scheme, they could construct a lower degree polynomial to reduce the storage required. Naturally, storing enough shares to reconstruct the secret (and the corresponding randomness) would allow for reconstruction of all shares while storing fewer shares than the designated threshold only enables the reproducibility of some shares.

3.3.2 Repairable threshold schemes

We now shift our focus to reproducibility by the shareholders. Consider a scenario in which the dealer is no longer active after initialization of the scheme and a shareholder has lost their share. Given a sufficient number of shares, a collection of shareholders can reproduce other shares in an information-theoretic setting by using a *repairable threshold scheme* (RTS) [105]. In an RTS, some threshold number of participants can combine the information from their shares to repair other shares. The amount of information required to repair a share must be equivalent to having the secret itself. If one could recover new shares using some number of shares less than the threshold number required to reconstruct the secret, then this would violate the security of the scheme.

Repairable threshold schemes, first considered in [76], allow a subset of shareholders to reconstruct a lost share for another participant, without the involvement of the dealer. This involves the introduction of a repair algorithm, in addition to the share and recover algorithms. A (k, n, d)-*repairable threshold scheme*, or (k, n, d)-RTS, is a (k, n)-threshold scheme which has a repair algorithm that allows a participant to reconstruct their share

with the help from a set of d participants. We say that the RTS has *universal repairability* if any subset of d players can repair another participant's share. Otherwise, we say that it has *restricted repairability* if some, but not all, subsets of d players can repair another participant's share. Combinatorial constructions of repairable threshold schemes were presented in [142] and later refined in [105]. We present an example based on combinatorial designs.

Definition 3.3.1. *An (m, d, λ)-balanced incomplete block design, or (m, d, λ)-BIBD, is a pair $(X, \mathcal{D})$ where X is a set of points, $\mathcal{D}$ is a collection of non-empty subsets, called blocks, of X, and:*

- $|X| = m$,
- *each block in $\mathcal{D}$ contains exactly d points,*
- *every pair of distinct points is contained in exactly λ blocks.*

Example 3.3.1. [A combinatorial repairable threshold scheme [142]] Construct a $(5, 7)$-threshold scheme with shares $v_1, v_2, \ldots, v_7$. These are the *subshares*.

The subshares that each participant receives will be determined by a *distribution design*. The distribution design will be a $(7, 3, 1)$-BIBD whose blocks are assigned to participants as follows:

$$\begin{array}{llll} P_1 = \{1, 2, 3\} & P_2 = \{1, 4, 5\} & P_3 = \{1, 6, 7\} & P_4 = \{2, 4, 6\} \\ & P_5 = \{2, 5, 7\} & P_6 = \{3, 4, 7\} & P_7 = \{3, 5, 6\} \end{array}$$

Each v_i is given to all participants having point i in their block from the distribution design.

Any pair of participants has enough points to reconstruct the secret. This follows from the properties of BIBDs. In particular, any two blocks intersect in at most one point, and so two blocks must contain five distinct points. Since the threshold number of subshares required to reconstruct the secret is five, it follows that any pair of participants can reconstruct the secret. If a participant loses their share, there exist two other participants who can provide each of their subshares. For example, if P_3 loses their share, they can recover v_1 from P_1 or P_2, v_6 from P_4 or P_7, and v_7 from P_5 or P_6. Note that this scheme satisfies a *restricted repairability* property since there exists some subset of three participants that can repair P_i's share, but not all subsets of size three are sufficient to repair P_i's share. ■

3.3.3 Repairability from evolving schemes

Another approach to reproducing or repairing shares is with the use of *evolving secret sharing schemes* [98]. An evolving secret sharing scheme considers a setting in which the access structure changes over time. A special case of evolving secret sharing schemes considers a dealer who knows the access structure of the secret sharing scheme but does not have an upper bound on the number of participants. In other words, they may be required to generate an infinite number of shares for a particular secret without changing the threshold number of participants required to reconstruct the secret. Evolving schemes were motivated by settings with an *online* dealer who remains active throughout the lifetime of the secret.

Here we present an example of an evolving 2-threshold scheme based on *prefix codes*. We use t to denote the time. At each increment of the time t, a new shareholder arrives and may request a share from the dealer. Any two shareholders can reconstruct the secret.

Definition 3.3.2 (Prefix code [65])**.** *For some set Q, a set of codewords $C \subseteq Q^*$ is a* prefix code *if no codeword in C is the prefix of another codeword in C.*

Construction 3.3.1. [Evolving 2-threshold scheme [98]] Let $s \in \{0,1\}$ be the secret and let $C : \mathbb{N} \to \{0,1\}^*$ be a prefix code for the integers, i.e., for any $t_1 \neq t_2 \in \mathbb{N}$, $C(t_1)$ is not a prefix of $C(t_2)$. We write $\sigma(t)$ to denote the length of $C(t)$ for $t \in \mathbb{N}$. Let w be an infinite random binary string, generated as needed by the dealer. At time $t \in \mathbb{N}$, assume the dealer holds the prefix of the string w of length $\sigma(t)$, denoted $w_{\sigma(t)}$. For simplicity, we assume that $\sigma(t)$ is monotonically increasing.

Share: The share of participant P_t (who arrives at time t) is the following:

- If $s = 0$ then P_t receives the share $u_t = w_{\sigma(t)}$.
- If $s = 1$ then P_t receives the share $u_t = w_{\sigma(t)} \oplus C(t)$.

Recover: To reconstruct the secret, any two participants, say P_{t_1} and P_{t_2}, holding shares u_{t_1} and u_{t_2}, respectively, where $|u_{t_1}| \leq |u_{t_2}|$, check if u_{t_1} is a prefix of u_{t_2}. If it is a prefix then they output $s = 0$. Otherwise, they output $s = 1$.

■

The concepts used in evolving secret sharing can be used in settings where reproducibility is desirable. For example, if a participant loses their share, they may request a new

one from the dealer. Rather than reproducing their original share, the dealer may use an evolving secret sharing scheme and simply construct a new share for the participant. Meanwhile, the threshold remains unchanged. It is important to note that using an evolving secret sharing scheme may be detrimental to the security of the scheme in the presence of malicious parties. Suppose there is a single malicious shareholder. This shareholder can pretend to have lost their share and request a new one from the dealer. The dealer complies and constructs a new share with an evolving secret sharing scheme. The malicious shareholder can repeatedly request a new share until they are able to single-handedly recover the secret. Similar to our discussion on reproducibility via the dealer in the information-theoretic setting, the evolving schemes described by Komargodski et al. [98] require that the dealer holds the secret and all of the randomness.

3.3.4 Comparison of approaches to reproducibility

When a trusted dealer remains active throughout the lifetime of the secret, the simplest approach to reproducibility is to have the dealer hold on to the secret and the randomness. Then, the dealer can use this information at a later point in time to reproduce shares, as needed. Although this is the most straightforward approach to reproducing shares, having a single, active party hold all of the secret information leaves an obvious target. This may defeat the purpose of distributing data in the first place. Such a strategy can appear to be counterintuitive to the goal of secret sharing schemes to begin with. The same holds true for reproducibility from evolving schemes, where we assume an online setting in which the dealer holds all of the information for an extended period of time.

Repairable secret sharing schemes allow for reproducibility of shares in a distributed manner, removing the requirement that a single party maintain all of the secret information. If the dealer is no longer present after initialization, then they are no longer a vulnerability of the system. Removing this risk can be preferable in settings where the goal is distribution of trust. Although repairability protocols require more work than simply having a dealer redistribute shares, there exist studies on efficient constructions of repairable threshold schemes [95].

3.4 Alternative access structures

Most of the work we have discussed has focused on threshold access structures. That is, the schemes require some threshold number of participants in order to reconstruct the

secret. In this section, we review work that focuses on more general access structures and modifications to classical access structures. This includes the following concepts:

General access structures: We may specify exactly which subsets of participants are authorized sets and which are not.

Weighted and hierarchical schemes: Some participants have more authority or information than others.

Dynamic access structures: The access structure may be updated over time.

Password-protected: In addition to an authorized set of participants, these schemes require a password to reconstruct the secret.

3.4.1 General access structures

Let $\mathcal{P}$ denote the set of participants in a secret sharing scheme and let $\mathcal{A} \subseteq 2^{\mathcal{P}}$ denote the designated access structure. That is, $\mathcal{A}$ is a set of subsets of $\mathcal{P}$. Ito, Saito, and Nishizeki [85] proved that there exists a perfect secret sharing scheme for any $\mathcal{A}$ as long as $\mathcal{A}$ is a *monotone* access structure. An access structure $\mathcal{A}$ is said to be *monotone* if, for every $B \in \mathcal{A}$ and $B \subseteq C \subseteq \mathcal{P}$, we have that $C \in \mathcal{A}$. In other words, if C is a set of participants containing an authorized set of participants B, then C must also be an authorized set. To construct schemes for general monotone access structures, Ito, Saito, and Nishizeki [85] use a method that involves giving multiple shares to each participant, which is sometimes called *replicated secret sharing*. In the remainder of this subsection, we will present a construction satisfying a general monotone access structure, by Benaloh and Leichter [15].

Suppose $\mathcal{A}$ is a monotone set of subsets of $\mathcal{P} = \{P_1, \ldots, P_n\}$. Let the set of possible secrets be $\mathcal{S} = \mathbb{Z}_m$ for some integer m. Denote the secret by $s \in \mathcal{S}$. First, we will construct a monotone circuit that represents the access structure and then build a secret sharing scheme based on the circuit. Let $\mathcal{O}$ be a monotone circuit (i.e., it has only *and* and *or* gates) with inputs $x_1, \ldots, x_n$ corresponding to each participant P_i. We can represent such a circuit with a Boolean formula containing $\wedge$ and $\vee$ operators. In particular, we can use the following Boolean formula: $\vee_{B \in \mathcal{A}} (\wedge_{x_i \in B} x_i)$.

We proceed by assigning a value, $f(w)$, to every wire w in the circuit. Assign the output wire w_{out} the value of the secret, s. We will iterate over gates in the circuit that have an output defined but no defined input until every wire has some value assigned to it. For some gate G with input wire w and output wire w_G, we define $f(w)$ as follows:

- If G is an *or* gate, $\vee$, then define $f(w) = f(w_G)$ for every input wire w of G.
- If G is an *and* gate, $\wedge$, then share the value $f(w_G)$ using the additive scheme described in Construction 2.0.2 among the t input wires. That is, choose $t-1$ elements of $\mathbb{Z}_m$ independently at random, say $y_1, \ldots, y_{t-1}$, compute $y_t = f(w_G) - \sum_{i=1}^{t-1} y_i \bmod m$, and let $f(w_i) = y_i$ for $i = 1, \ldots, t$.

This completely determines the monotone circuit. The corresponding secret sharing scheme simply consists of giving each participant P_i the list of values $f(w)$ where w is an input wire of the circuit which receives input x_i.

Example 3.4.1. Suppose the access structure is $\mathcal{A} = \{\{P_1, P_2, P_4\}, \{P_2, P_3\}, \{P_1, P_3\}\}$ and the secret is $7 \in \mathbb{Z}_{31}$. The corresponding Boolean formula is

$$(P_1 \wedge P_2 \wedge P_4) \vee (P_2 \wedge P_3) \vee (P_1 \wedge P_3).$$

After iterating through all gates and assigning values to wires, the circuit looks something like the following (where the random values chosen for input wires to $\wedge$ gates may vary).

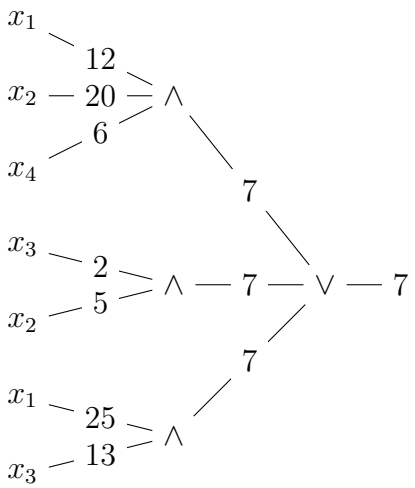

Figure 3.1: Example of a monotone circuit for a general access structure

Then, P_1 receives $(12, 25)$, P_2 receives $(20, 5)$, P_3 receives $(2, 13)$ and P_4 receives (6). Any authorized set can recover the secret and any unauthorized set learns no information about the secret. ■

This construction, although applicable to any monotone access structure, is not particularly efficient. In the case of a (k, n)-threshold scheme, using the circuit construction requires that we give each participant $\binom{n-1}{k-1}$ elements of the group, as opposed to the single element that is given in Shamir's scheme. There is a significant body of work focused on improving the efficiency of secret sharing schemes for different access structures. The efficiency of a secret sharing scheme can be measured by the *information rate*, described in Chapter 2, and there are many papers discussing bounds on the information rate of secret sharing schemes [40, 96, 27, 139, 26]. To improve the efficiency of general constructions, some work has shifted to a computational setting to achieve smaller shares [100] while others focus on constructing schemes with an optimal information rate for particular access structures [36, 86]. A bulk of this work has focused on *graph access structures*, where the access structure is described by a graph such that a subset of participants can reconstruct the secret only if they contain an edge in the graph [147, 23, 69, 9].

3.4.2 Weighted and hierarchical schemes

Consider a scenario in which certain participants have more authority than others. These settings can be represented by *weighted threshold schemes* or *hierarchical schemes*.

In a weighted threshold scheme, each shareholder is assigned a positive weight and a subset of shareholders can reconstruct the secret if and only if the sum of their weights is greater than or equal to a certain threshold. There is a natural way of constructing a weighted threshold scheme where the total weight required is some positive number k from (k, n)-threshold schemes. Each participant is simply given a number of shares equal to their assigned weight. This, however, is not the most efficient way of constructing weighted threshold schemes. Beimel, Tassa, and Weinreb [10] characterize which weighted threshold access structures are ideal, i.e., can be achieved with information rate equal to one, and discuss how to construct them from a linear ideal secret sharing scheme.

Weighted threshold schemes can be considered a specialization of hierarchical threshold schemes, which were introduced by Simmons [135]. In a hierarchical threshold scheme, the participants are partitioned into a hierarchy of disjoint levels $L_1, \ldots, L_m$ with corresponding thresholds $k_1 > k_2 > \cdots > k_m$. Then, a set of participants is an authorized set if and only if it contains at least k_i users from the ith level and above, for at least one $i \in 1, \ldots, m$. This may also be called a *disjunctive multilevel scheme*. Similarly, in a *conjunctive multilevel scheme* [145], a set of players is said to be authorized if and only if it contains at least k_i users from the ith level *for all* $i \in 1, \ldots, m$. Simmons [135] and Brickell [36] provide constructions for disjunctive multilevel schemes based on finite geometries while Tassa [145]

provides a construction for conjunctive multilevel schemes based on Birkhoff polynomial interpolation.

Later, Nojoumian and Stinson [117] introduced *sequential secret sharing*, in which players still belong to different levels of authority but can only recover a secret belonging to their own level. Also, there exists a master secret which can be revealed if all of the secrets in higher levels are first recovered.

3.4.3 Updating the access structure

Consider an access structure that changes over time. Some desired changes to the access structure may include adding/removing participants, adjusting thresholds, and adding/removing authorized sets of participants. In this subsection, we discuss existing methods for updating the access structure in secret sharing schemes.

Adding participants

It is not hard to imagine a scenario in which we would like to add a new player to a threshold scheme. This can trivially be accomplished if the dealer is still active after initialization by having the dealer generate a new share for the new participant. In the setting where the dealer is no longer active after initialization, the shareholders can cooperate to generate a share for a new participant without reconstructing the secret. This is called an *enrollment protocol* or an *admission protocol*. For a (k, n)-threshold scheme, at least k players must cooperate to generate a new share, resulting in a $(k, n+1)$-threshold scheme.

A number of enrollment/admission protocols have been introduced in past works which were very computationally expensive [80, 114, 132]. Saxena, Tsudik, and Yi [131] presented a more efficient non-interactive protocol for admitting new participants in a threshold scheme which is computationally secure and based on bivariate polynomials. Nojoumian, Stinson, and Grainger [118] presented an efficient, unconditionally secure enrollment protocol, assuming all participants correctly follow the protocol, to generate shares for new participants under Shamir's threshold scheme. This protocol is interactive and presented below. It is interesting to note that, again, we can see a trade-off between non-interactivity and unconditional security, akin to the trade-off discussed in Section 3.2.2.

Example 3.4.2. [Unconditionally Secure Enrollment Protocol [118]] The following assumes that all participants follow the protocol honestly and that there exist pairwise secure channels between participants.

1. Each player P_i, for $1 \leq i \leq t$, computes their Lagrange interpolation coefficient:
$$\gamma_i = \prod_{1 \leq j \leq t, i \neq j} \frac{k-j}{i-j} \text{ where } i, j, k \text{ are players' ids.}$$

2. Participant P_i multiplies their share φ_i by their Langrange interpolation coefficient and splits the result into t portions:
$$\varphi_i \times \gamma_i = \partial_{1i} + \partial_{2i} + \cdots + \partial_{ti}.$$

3. Players exchange ∂_{ji}'s accordingly through pairwise channels. P_j adds the ∂_{ji}'s together and sends the result to P_k:
$$\sigma_j = \sum_{i=1}^{t} \partial_{ji}.$$

4. P_k adds each of the σ_j values together to get their share φ_k:
$$\varphi_k = \sum_{j=1}^{t} \sigma_k.$$

The proof of security of this protocol can be found in [118]. ■

An alternative approach to using an enrollment protocol is to use an *evolving secret sharing* scheme. We briefly discussed evolving schemes in Section 3.3.3. Evolving secret sharing, introduced by Komargodski et al. [98], considers the case where the set of shareholders is not known in advance and could possibly be infinite. Evolving schemes are motivated by an online setting in which the dealer remains active over a long period of time and participants may arrive at any point. These schemes require that there is no communication to the parties who have already received shares when a new participant arrives. The initial paper presents constructions for k-threshold evolving schemes where k is a fixed constant and the number of shareholders may be unbounded. This is not possible in Shamir's scheme, for example, since shares are an element of a finite field $GF(q)$, and so it is limited to at most $q-1$ shares. In [99], evolving schemes are extended to a dynamic threshold access structure, where the qualified sets are of increasing size as the number of participants increases. They also present an extension which is robust, where the secret can be recovered even with some invalid or corrupt shares. Later, [126] considered the problem of evolving secret sharing with t essential participants, namely where a fixed set of t participants are essential and may receive shares of constant size. Typically, the share sizes (and consequently, the information rate) in these constructions are relatively large to account for the potentially unbounded set of participants.

Removing participants

The ability to withstand the loss or removal of a participant in a threshold scheme has been termed *disenrollment capability* [18]. The removal or loss of a participant means that their share should become invalid. The remaining shares thus need to be modified so that the invalid share can no longer be used. Otherwise, in the case of a (k, n)-threshold scheme, the publication of a lost share would decrease the threshold to $k-1$. Blakley, Blakley, Chan, and Massey [18] introduced the term *disenrollment capability* and considered systems in which participants distribute information through insecure channels to modify the remaining shares.

Charnes, Pieprzyk, and Safavi-Naini [43] present a disenrollment scheme in the computational setting based on Shamir's scheme. Rather than having participants interact with one another to remove a participant, this construction requires the existence of a *combiner* participant. The combiner is responsible for maintaining random values which will be used to update shares when a participant is removed. The combiner is also responsible for reconstructing the secret using these random values.

Proactive secret sharing schemes [80] can also be used to remove participants. Proactive secret sharing schemes update shares over time to prevent an adversary from learning the secret from leaked shares over time. To remove a participant, the active participants can update their shares via a proactive scheme and exclude the inactive participant from the update so that the inactive participant does not receive a new, valid share. We discuss proactive schemes in more detail in Section 4.4.1.

Threshold adjustments

In a (k, n)-threshold scheme, it may be desirable for the threshold parameter, k, to not remain static over time. A decrease in mutual trust may result in a desire for a threshold increase and an increase in mutual trust may result in a desire for a threshold decrease. Schemes which allow the threshold parameter to be changed over time are sometimes called *threshold changeable secret sharing schemes*. The ability to increase or decrease the threshold parameter can be considered in several settings: with or without a dealer, and with or without secure channels between participants. Threshold adjustments can trivially be achieved in a setting where the dealer is present and there exist secure channels. This is easily done by having the dealer issue new shares to all participants and having participants discard their old shares. This solution, however, is inefficient and often impractical.

Consider a setting where shares have been distributed to participants under secure channels, but these secure channels are no longer in use. Under the presence of the original

dealer, threshold changes can be achieved by two general techniques, described in detail by Martin et al. [110]:

- During the original initialization, the dealer gives each participant a secret key in addition to their share. At a later point in time, the dealer can send new shares (corresponding to a different threshold) to participants, encrypted under their respective secret key.
- During the original initialization, the dealer gives each participant shares for the (k, n)-threshold scheme and for a $(n+1, 2n)$-threshold scheme. To change the threshold parameter, they broadcast a specific number of shares of the second scheme that effectively change the threshold parameter.

The dealer does not need to know upon initialization what the change in threshold will be, but they do need to know that a change may be needed in order to properly set up the scheme. Participants must be trusted to discard their old shares in the case of a threshold increase.

Most of the work in this topic has considered a setting in which the dealer is no longer present after initialization. In this case, participants can work together to change the threshold. We now describe several known techniques for accomplishing these changes. Martin summarizes the models that might be considered when dynamically changing thresholds [107].

- The *zero addition* technique [116, 80] can be used for threshold increase. This involves generating shares corresponding to a (k', n)-threshold scheme, for some $k' > k$, with secret equal to 0 and then adding these shares to the original shares.
- The *public evaluation* technique [116] can be used to decrease the threshold. This involves participants collectively generating a new share (as in enrollment protocols) and then publishing this share so that the threshold is decreased.
- The *resharing* or *redistribution* technique [13, 55, 111, 116] can be used to increase or decrease the threshold. This technique, demonstrated in the degree reduction step of Construction 3.2.1, involves participants treating their own share as a secret and resharing it among all other participants according to a new threshold value. This technique often relies on the linearity of the secret sharing scheme so that reshared values can be added together to obtain new shares.

- In a setting without secure channels after initialization, a *ramp scheme* can be used to allow for threshold increases, assuming that the participants honestly delete their original shares after updating them [110]. In this construction, it must be known ahead of time that a threshold change is desired.

- In an active adversary setting, techniques from *verifiable secret sharing* can be applied when updating shares to prevent shareholders from corrupting the updates [116].

- In a non-interactive setting, only threshold increase can be accomplished (otherwise, it would violate the privacy requirement of threshold schemes). Harn and Hsu [78] suggest using bivariate polynomials for this purpose, where initial shares are polynomials that are later evaluated to shares dependent on the desired threshold.

Updating general access structures

More generally, we might consider how changes can be dynamically made to general access structures. One approach is to setup a secret sharing scheme in such a way that a specific access structure can be activated at some point in time via a broadcast message to all participants. This is called a *prepositioned scheme*. We previously discussed prepositioned schemes in Section 3.2.1 as a way to enable multiple secrets to be recovered, but they can also be used to activate particular access structures. This is demonstrated in Example 3.2.3. Fully dynamic schemes [21] often satisfy the properties of prepositioned schemes. Additionally, the activated access structure may be changed over time. In other words, at any particular time i, if an access structure $\mathcal{A}^{(i)}$ has been enabled via some broadcast message, then any set of participants not within the access structure should have no information about the corresponding secret s_i, even knowing the $i-1$ previous broadcast messages. These two approaches to updating the access structure require the presence of the dealer.

Just like with threshold adjustments, we might consider how to update a general access structure without the presence of the dealer. Desmedt and Jajodia's [55] technique of *share redistribution* which we discussed for threshold adjustment also applies to general access structures. The main idea is the same: participants treat their shares as secrets and reshare them among the other participants under the new access structure. The redistribution technique of Desmedt and Jajodia requires that the underlying scheme is linear. This enables the redistributed values to be combined into a single share. In particular, they show how to perform share redistribution under Benaloh and Leichter's scheme for general access structures [15], a scheme which we discussed in Section 3.4.1.

3.4.4 Password-protected secret sharing

Password-protected secret sharing (PPSS) [8], also sometimes called threshold password-authenticated secret sharing (TPASS), introduces the additional requirement of a password to the access structure. When storing data, simply encrypting data using a key derived from some password is vulnerable to password leaks and dictionary attacks. By combining password authentication with secret sharing, we can address a scenario in which a user can secret share some data among n servers and protect it with a password p such that they can later recover the data using p and some threshold number of honest servers. Further, any coalition of servers less than the threshold number should learn nothing about the data.

The initial construction of PPSS [8] relies on the decisional Diffie-Hellman (DDH) assumption and non-interactive zero-knowledge proofs, and so it is in the computational model. Later work made improvements on the number of rounds required [90, 91], removing assumptions of secure channels and public keys [38], and improving robustness [1]. Each of these constructions is considered in the computational model.

Password-protected secret sharing is similar in theory to hierarchical secret sharing schemes, which we discussed in Section 3.4.2. We could conceive of a hierarchical scheme with two levels (one for the user and one for the servers) where the user's share and k out of the n server shares are required to reconstruct the secret. One difference here is that the password comes from some dictionary which is likely different than the set of possible shares, since a password should be memorizable by a human. In addition, the models considered in PPSS constructions are most similar to those considered in password-authenticated key exchange protocols [12], where a client holding a password interacts with one or more servers (in a public key setting) to establish shared keys.

3.5 Anonymity

Anonymous secret sharing schemes were first investigated in 1988 by Stinson and Vanstone [140] and then further investigated in 1997 by Blundo and Stinson [28]. In anonymous secret sharing, the secret can be reconstructed without knowledge of which participants hold which shares. The shares are given to a trustworthy machine that does not know the identities of the participants. The recover algorithm carried out by the trusted machine ensures anonymity of the participants.

3.6 Relaxations

Thus far, we have described variations of secret sharing that have strengthened or extended the classical definitions in some form or another. Going in the other direction, we can consider how to relax the requirements of secret sharing schemes. Given that we initially introduced secret sharing schemes as providing two properties, privacy and correctness, we can consider how to relax one or the other.

3.6.1 Relaxing correctness

Relaxing the correctness requirement is accomplished by *probabilistic secret sharing* [51]. Recall the correctness requirement of a classical secret sharing scheme states that any authorized set of participants can reconstruct the secret. In probabilistic secret sharing, we relax this requirement to a property named α-correctness. A probabilistic scheme ensures α-correctness if for any authorized subset of participants, they can reconstruct the secret with probability greater than or equal to α. In the classical case, $\alpha = 1$. The privacy requirement does not change. We present an example from [51] for a $(2, n)$-threshold scheme, although the construction actually applies to the case where the number of shareholders is not limited (i.e., a $(2, \infty)$-threshold scheme).

Example 3.6.1. [Probabilistic secret sharing scheme [51]] Let $s \in \{0, 1\}$ be the secret and P_i denote the ith shareholder.

Share: For the first participant, P_1, their share is a random bit b_1. For the remaining participants P_i, $i > 1$, the shares are as follows:

- If the secret is $s = 0$ then the share is b_1.
- If the secret is $s = 1$ then the share is a new random bit b_i.

Recover: The reconstruction algorithm takes as input two bits, b_i and b_j, and outputs $b_i \oplus b_j$.

The paper proves that this is a $(1+p)/2$-probabilistic $(2, n)$-threshold scheme where $(p, 1-p)$ is the distribution of the secret bit. That is, the scheme achieves $(1 + p)/2$-correctness. ■

3.6.2 Relaxing privacy

Relaxing the privacy requirement can be accomplished in several different ways. In the classical, information-theoretic definition of secret sharing schemes, an adversary with *infinite* computational resources learns *no information* about the secret given any unauthorized set of shares. We can either relax the requirement that they learn no information or the property that they have infinite computational resources.

Classical secret sharing schemes are said to be *perfect* since they satisfy the property of *perfect privacy*, as discussed in Chapter 2. A *non-perfect* scheme may leak partial information to any subset of participants. The following list outlines variations of non-perfect secret sharing schemes. Jafari and Khazaei [89] expand on the relationships between the following notions.

- **Quasi-perfect**: In *quasi-perfect* secret sharing [94], it is required that the *percentage of information* leaked/missed is negligible. That is, any unauthorized set may learn a negligible amount of information about the secret. Additionally, there is a negligible probability that an authorized set cannot reconstruct the secret. So, this definition relaxes both correctness and privacy requirements by some small amount, ϵ, in an information-theoretic sense. In terms of entropy definitions, for an authorized set A and a set of secrets $\mathcal{S}$, $H(\mathcal{S} \mid A)/H(\mathcal{S}) \leq \epsilon$ for some negligible ϵ.
- **Almost perfect**: In *almost perfect* secret sharing [93], some small amount of information (in terms of entropy) can be leaked/missed. That is, for an authorized set A and a set of secrets $\mathcal{S}$, the entropy is $H(\mathcal{S} \mid A) \leq \epsilon$. (Recall that in a perfect scheme, $H(\mathcal{S} \mid A) = 0$.)
- **Partial**: A *partial* secret sharing scheme [89] requires that the information gained about the secret by any authorized set is strictly greater than the information gained by any unauthorized set. That is, for an authorized set A, unauthorized set B, and set of secrets $\mathcal{S}$, we require that $H(\mathcal{S} \mid A) < H(\mathcal{S} \mid B)$.
- **Statistical privacy**: A *statistical* privacy guarantee ensures that an adversary only learns a limited amount of information from an unauthorized set of shares, as opposed to no information at all (in an information-theoretic sense). They require that any unauthorized set of parties can only distinguish between shares corresponding to two different secrets with negligible probability.

On the other hand, we can consider an adversary in the *computational* setting. In a computational setting, an adversary learns no information about the secret assuming

they are computationally bounded. This relaxation allows for shorter shares and/or more efficient constructions [100]. Most variations of secret sharing were initially considered in the information-theoretic setting.

3.7 Summary

Recall from Chapter 2 that in a classic secret sharing scheme in the information-theoretic setting, we have a dealer who distributes shares among a set of participants such that the following two properties hold:

Correctness: Any authorized set of parties can reconstruct the secret.

Perfect Privacy: Any unauthorized set of parties learns *no information* about the secret from their shares.

In this chapter, we analyzed how previous work has extended this basic primitive to achieve new capabilities. The content of the secret itself (Section 3.1) and the structure of authorized sets (Section 3.4) have been extended and generalized. The properties of privacy and correctness have been relaxed to increase efficiency (Section 3.6). Also, additional functionality has been included in the design to enable updating secrets (Section 3.2), to enable updating access structures (Section 3.4.3), and to enable reproducibility of shares (Section 3.3). In this chapter, we presented a protocol to compute the product of two secrets in Shamir's scheme, which appeared to be difficult to find in the literature, despite being well known. This chapter also included a comparison and analysis of approaches to updating secrets without the presence of the dealer and a comparison and analysis of approaches to reproducing shares. We also observed that a trade-off between non-interactivity and unconditional security appeared in protocols extending the functionality of secret sharing schemes. In the next chapter, we continue our investigation of extensions to the original definitions of secret sharing schemes as we explore alternative adversarial settings.

Chapter 4

Alternative Adversarial Settings

This chapter considers variations of secret sharing that address new adversarial settings. In classical secret sharing schemes, we assumed that the dealer and the participants involved all act honestly and follow the protocol. Such an assumption may not always hold in practice, so it is interesting to consider alternative adversaries. Martin [108] presented an interesting survey on the topic of adversarial settings entitled "Challenging the adversary model in secret sharing scheme."

In this chapter, we propose new game-theoretic definitions for a number of adversarial settings. By using consistent terminology and definitions, we unify the notions presented in previous works and clarify the differences between various concepts. We consider the setting where the dealer may not be trusted, settings where shareholders may misbehave, and settings where shares may be leaked. Previous studies on the topic of verifiability often presented informal definitions or did not clearly demonstrate the differences in the adversarial models considered. This resulted in some difficulty in distinguishing between the security properties provided by different works that all claim to provide some form of security against an untrusted dealer. These subtleties are clarified in this chapter. Additionally, we formalize a notion of deniability, which prior to this work had not been considered in the context of secret sharing.

4.1 Verifiability

In a setting where the dealer is not trusted, it is useful to be able to verify the validity or the consistency of shares. This is the goal of *verifiable secret sharing* (VSS) [45]. VSS

schemes enable honest parties to recover the secret even in the presence of a malicious dealer who has corrupted some shares. Somewhat more formally, a scheme is *verifiable* if upon receiving a share, a participant can test whether or not it is a valid share. Given a set of valid shares which make up an authorized set, there exists a unique secret which is output by the Recover algorithm. VSS is one of the most widely used modifications of classical secret sharing schemes and is often employed in multi-party computation schemes. There exist several definitions and models under which verifiability has been considered. In this section, we analyze and compare the most prominent definitions of verifiability. Martin [108] also provides a survey of various definitions of verifiable secret sharing. In our work, we analyze the definitions in more detail and contribute new definitions using adversarial games, to clarify the differences in approaches to verifiability. Throughout this section, we consider (k, n)-threshold schemes.

A verifiable secret sharing scheme introduces a Verify algorithm which allows participants to verify the validity of their shares before reconstruction. A VSS scheme is *interactive* if the Verify algorithm requires that the shareholders interact with one another. Otherwise, we say it is *non-interactive*. In what follows, we say that a set of shares is *consistent* if for all authorized subsets of participants, A, who have verified and accepted their shares, there exists a secret s' such that every such subset A reconstructs the secret s'. In other words, a set of shares is *inconsistent* if there exists two authorized subsets, A_1, A_2, of participants that have verified their shares such that A_1 reconstructs a secret s_1 and A_2 reconstructs a secret $s_2 \neq s_1$.

4.1.1 Non-interactive VSS

The first scenario we consider is a (k, n)-threshold scheme where the players are all honest but the dealer is malicious. In the following, we will use a non-interactive Verify algorithm, meaning that the verification process involves shareholders running the Verify algorithm on the information they have received from the dealer. This information can include shares and publicly broadcasted information. It does not allow for interaction between shareholders. This definition is useful in a scenario where participants cannot communicate with one another but trust that the others are acting honestly. The following definition is a new game-theoretic definition for non-interactive VSS which encompasses much of the previous work in non-interactive VSS [121, 68].

The Non-Interactive Verifiability Game. Consider a probabilistic polynomial-time adversary acting as the dealer. Let $\mathcal{P}$ denote the set of n (honest) shareholders and consider a (k, n)-threshold scheme.

Step 1. The adversary generates n (possibly invalid) shares and gives one share to each participant $P_1, \ldots, P_n \in \mathcal{P}$. The adversary designates two distinct sets of participants, say A_1, A_2, both of size k.

Step 2. All participants in A_1 and A_2 verify their shares using a non-interactive Verify algorithm.

Step 3. If all shares are verified in step 2 then A_1 reconstructs a secret s_1 and A_2 reconstructs a secret s_2.

The adversary wins if the set of shares is not *consistent*. That is, the following two properties must hold:

1. all parties in A_1 and A_2 have verified and accepted their shares, and
2. $s_1 \neq s_2$.

This definition of verifiability ensures that the shares distributed by the *dealer* are consistent with one another. Since the Verify algorithm is non-interactive, such a scheme does not address the potential of misbehaving shareholders (this will be discussed in Chapter 5). Therefore, it is possible that a malicious shareholder could corrupt their share before reconstruction. Feldman [68] and Pedersen [121] both provided constructions of non-interactive VSS schemes. Feldman [68] presented the first non-interactive VSS scheme, which relies on the existence of a hard-to-invert encryption scheme. The privacy guarantee is computational while the verifiability guarantee is information-theoretic. Pedersen's non-interactive protocol [121] has an information-theoretic guarantee of privacy (it adapts Shamir's scheme) and a computational guarantee of verifiability. We present Pedersen's scheme as an example next.

Example 4.1.1. [Pedersen's Non-Interactive VSS [121]] Let p, q be large primes such that q divides $p-1$. Let $\mathbb{G}_q$ be the finite subgroup of $\mathbb{Z}_p^*$ of order q. Let $g, h \in \mathbb{G}_q$. Suppose the dealer wants to share a secret $s \in \mathbb{Z}_q$ using a (k, n)-threshold scheme. In what follows, we denote a commitment to some $x \in \mathbb{Z}_q$ by $E(x, y) = g^x h^y$ for some random $y \in \mathbb{Z}_q$.

1. **Share:** The dealer randomly chooses $t \in \mathbb{Z}_q$ and publishes a commitment to s, given by
$$E_0 = E(s, t) = g^s h^t.$$
The dealer selects a random polynomial $u(x) \in \mathbb{Z}_q[x]$ of degree $k-1$ such that $s = u(0)$ is the secret. Denote this polynomial by $u(x) = s + u_1 x + u_2 x^2 + \cdots + u_{k-1} x^{k-1}$. Let $s_i = u(i)$, for $i = 1, \ldots, n$.

The dealer randomly selects another polynomial $v(x) \in \mathbb{Z}_q[x]$ of degree $k-1$, which we denote $v(x) = t + v_1 x + \cdots + v_{k-1}x^{k-1}$. Let $t_i = v(i)$ for $i = 1, 2, \ldots, n$.

The dealer publishes (i.e., broadcasts) a commitment $E_i = E(u_i, v_i)$ for each $i = 1, \ldots, k-1$ and each participant receives the share (s_i, t_i) via secure channel from the dealer.

2. **Verify:** For participant P_i to verify their own share, P_i checks that

$$E_i = E(s_i, t_i) = \prod_{j=0}^{k-1} E_j^{i^j}.$$

Given valid shares, both sides of the equation should evaluate to $g^{s_i}h^{t_i} = g^{u(i)}h^{v(i)}$.

3. **Recover:** The recovery process is the same as in Shamir's scheme. Given k shares, participants can recover s using polynomial interpolation.

The security of this VSS scheme depends on the security of the commitment scheme, E. It is a computational guarantee, given by the fact that $E(s,t)$ reveals no information about s unless an adversary can solve the discrete logarithm $\log_g(h)$. This is proven in [121]. ■

It is interesting to note that both Pedersen [121] and Feldman's [68] non-interactive constructions are in the computational setting. Pedersen [121] argued that it is impossible to achieve non-interactivity in the information-theoretic setting. The proof provided by Pedersen that this verifiability property cannot be achieved in a non-interactive manner in the information-theoretic setting is missing a necessary assumption that $n \geq k+1$. We provide a refined version of the proof next. This trade-off between non-interactivity and unconditional security is reminiscent of the trade-off we discussed in Section 3.2.2 and Section 3.4.3.

Theorem 4.1.1 ([121])**.** *There does not exist a non-interactive (k, n)-threshold secret sharing scheme in which no information about the secret is revealed and even an infinitely powerful dealer cannot compute inconsistent shares, for $n \geq k+1$.*

Proof. Let b denote the information broadcasted by the dealer in a non-interactive secret sharing scheme. Assume that $s_1, \ldots, s_k, s_{k+1}$ is a set of valid shares corresponding to a secret s. Let $V(i, b, s_i)$ denote the verification predicate computed by P_i to verify their share s_i. Given $s_1, \ldots, s_{k-1}$, define

$$S_k(b) = \{s'_k \mid V(k, b, s'_k)\}.$$

In words, $S_k(b)$ denotes the set of possible valid shares for a participant P_k. Then, $s_k \in S_k(b)$ and $s_1, \ldots, s_k$ will reconstruct the secret s.

Suppose that there exists some $s'_k \in S_k(b)$ such that $s_1, \ldots, s_{k-1}, s'_k$ reconstructs to a secret $s' \neq s$. Then, $s_1, \ldots, s_{k-1}, s'_k, s_{k+1}$ is a set of inconsistent shares since $s_1, \ldots, s_{k-1}, s'_k$ reconstructs s' while $s_1, \ldots, s_{k-1}, s_{k+1}$ reconstructs s.

This is impossible, since we required that even an all-powerful dealer cannot find inconsistent shares. Therefore, it holds that, for all $s'_k \in S_k(b)$, the k shares, $s_1, \ldots, s_{k-1}, s'_k$, reconstruct the secret s. But then $P_1, \ldots, P_{k-1}$ can find the secret s by guessing a secret share $s_k \in S_k(b)$ and combining their shares to obtain s. This contradicts the privacy requirement of the secret sharing scheme. □

4.1.2 Unconditionally secure VSS

A disadvantage of non-interactive VSS is that it cannot detect if malicious shareholders have corrupted their shares before reconstruction. If we allow interaction between participants, then honest participants can verify the validity of other participants' shares before reconstruction, thereby detecting any last-minute corruption of shares. Additionally, we can then consider constructions in an information-theoretic model as opposed to a computational setting. The schemes may be perfectly secure both in terms of the privacy of the secret sharing scheme and the verifiability. That is, any set of unauthorized shareholders obtains no information about the secret *and* if an authorized set of participants verifies their shares to be correct, then they all will be able to reconstruct the same secret value. In this model, we assume that there exist private channels between each pair of participants.

For our definitions, we consider (k, n)-threshold secret sharing schemes which follow the subsequent template. The two game-based definitions which follow are both new definitions of unconditionally secure VSS. Later, we discuss how these definitions encompass previous work in the unconditionally secure setting [141].

Unconditionally Secure Verifiable Secret Sharing Scheme Template

Step 1. A dealer generates n shares and gives one share to each participant $P_1, \ldots, P_n \in \mathcal{P}$.

Step 2. Participants participate in an interactive Verify algorithm which outputs either 0 or 1, denoting failure or success of verification of shares, respectively. In the case of success, the Verify algorithm also outputs a subset of players, say P_0, to be used in the next step.

Step 3. If Step 2 succeeds, the participants in the set P_0 perform a reconstruction, outputting a secret or $\perp$, which denotes failure of reconstruction.

We consider two different scenarios. In the first, the dealer *and* a subset of shareholders is malicious. In this scenario, the honest participants would like to verify that their shares are consistent, enabling them to recover a consistent secret despite the presence of malicious shareholders and a malicious dealer. The malicious participants attempt to thwart reconstruction.

The Unconditionally Secure Verifiability Game with Malicious Dealer. Let $\mathcal{P}$ denote the set of n shareholders and suppose there is an adversary that controls the dealer and $b-1$ of the participants. We assume that $b-1 < k$. In Step 1 of the scheme, the adversary generates n possibly invalid shares and distributes them among the participants.

The adversary wins the game if the verification in Step 2 (as described above) succeeds but the reconstruction in Step 3 fails.

In the second scenario we consider, the dealer is honest and a subset of shareholders is malicious. In this scenario, the honest participants would like to verify that their shares are consistent and that they can recover the intended secret which was shared by the honest dealer. The malicious participants try to ensure that an incorrect secret is reconstructed or that no secret is reconstructed at all.

The Unconditionally Secure Verifiability Game with Honest Dealer. Let $\mathcal{P}$ denote the set of n shareholders, and suppose there is an adversary that controls b of the participants. We assume that $b < k$. The honest dealer generates and distributes shares corresponding to a secret s in Step 1.

The adversary wins the game if one the following cases occurs:

1. *verification in Step 2 fails, or*
2. *verification in Step 2 succeeds and reconstruction in Step 3 fails, or*
3. *verification in Step 2 succeeds and Step 3 outputs a secret s_1, where $s_1 \neq s$.*

This definition differs from the non-interactive definition from Section 4.1.1 in a few ways. In the unconditionally secure setting, the verification procedure must be interactive, as we proved in Theorem 4.1.1. The adversary also has the ability to control some subset of

shareholders in this model. Therefore, an unconditionally secure VSS scheme can protect against malicious shareholders attempting to modify their shares before reconstruction.

Rabin and Ben-Or [127] present a construction of an unconditionally secure VSS scheme using the idea of *check vectors*. Stinson and Wei [141] present another construction of an unconditionally secure VSS scheme, based on symmetric polynomials, which we present next.

Example 4.1.2. [Unconditionally Secure VSS [141]] Let $\mathcal{P}$ denote the set of n participants, let $\mathcal{S} = GF(q)$ be the set of possible secrets, and let $\omega \in GF(q)$ be a primitive element. Let $s \in \mathcal{S}$ be the secret. An adversary can control up to b of the participants (including the dealer). The protocol is outlined in the following steps, where we assume that honest participants follow the protocol as described. An adversary cannot win the unconditionally secure verifiability game (with honest or malicious dealer) for this protocol, provided that $n \geq k + 3b$ and $k > b$, where k is the threshold number of participants required to reconstruct a secret.

1. **Share:** An honest dealer chooses a random symmetric polynomial in $GF(q)[x, y]$:

$$f(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{ij} x^i y^j,$$

   where $a_{00} = s$ and $a_{ij} = a_{ji}$ for all i, j. For each $m \in 1, \ldots, n$, the dealer sends $h_m(x) = f(x, \omega^m)$ to P_m.

2. **Verify:** Each P_m sends $h_m(\omega^\ell)$ to P_ℓ for $1 \leq \ell \leq n, \ell \neq m$. Each P_ℓ checks whether $h_m(\omega^\ell) = h_\ell(\omega^m)$ for $1 \leq m \leq n$, $\ell \neq m$. If P_ℓ finds that $h_m(\omega^\ell) \neq h_\ell(\omega^m)$ then P_ℓ broadcasts (ℓ, m). (This indicates that P_ℓ does not accept the value $h_m(\omega^\ell)$.)

   Each P_i computes a maximum subset $G_i \subseteq \{1, \ldots, n\}$ such that any ordered pair $(\ell, m) \in G \times G$ is not broadcasted. This set G_i is computed based on publicly broadcasted information and its size is uniquely determined. Different players may compute different subsets G_i (all G_i's will have the same size), so they should agree on a particular set G before the following step. For example, the players could decide ahead of time to choose whichever set is lexicographically least. If $|G| \geq n - b$, then P_i outputs $ver_i = 1$. Otherwise, P_i outputs $ver_i = 0$.

3. **Recover:** Each P_i sends $h_i(0)$ to each player P_m, where $i \in G$. P_m computes a polynomial $f_m(0, y)$ such that $f_m(0, \omega^i) = h_i(0)$ for at least $n - 2b$ of the data they received. This variation of error correction is to account for the fact that b malicious

players may follow the protocol correctly up until sending $h_i(0)$. P_m computes and outputs $s' = f_m(0,0)$. If P_m and the dealer are honest, then $s' = s$.

■

The above construction occurs in the information-theoretic setting which imposes some trade-offs on other aspects of the protocol. In particular, an information-theoretic verifiable scheme can only tolerate an adversary if no three unauthorized subsets span the entire participant set [13, 48].

Stinson and Wei [141] define unconditionally secure VSS to be those that satisfy the following set of properties:

1. If an honest player rejects the shares during the verification process then every honest player rejects the shares.
2. If the dealer is honest then all honest players verify and accept their shares.
3. If at least $n - b$ players accept their shares then there exists an $s' \in \mathcal{S}$ such that the event that all honest P_i output s' after reconstruction is fixed after sharing, and $s' = s$ if the dealer is honest.
4. If $|S| = q$, s is chosen randomly from $\mathcal{S}$, and the dealer is honest then any coalition of at most $k - 1$ participants cannot guess at the end of Share the value of s with probability greater than $1/q$.

It is not hard to see that these properties are sufficient to guarantee that an adversary cannot win either of the two unconditionally secure verifiability games. First, consider the unconditionally secure verifiability game with a malicious dealer and assume the above properties hold for a given scheme. An adversary who can win the game ensures that the participants verify their shares in Step 2, while reconstruction fails in Step 3. This would contradict property 3 in the list above which states that if at least $n - b$ players accept their shares then there exists a valid secret that the participants reconstruct.

Now consider the unconditionally secure verifiability game with an honest dealer. Again, assume the above properties hold for a given scheme. Each win condition for the adversary leads to a contradiction:

1. If verification in Step 2 fails, then this contradicts property 2 which states that if the dealer is honest then verification succeeds.

2. If verification succeeds but reconstruction fails, then this contradicts property 3 which states that there exists a secret s' which will be output after reconstruction.

3. If verification and reconstruction succeed but a secret not equal to s is reconstructed, then this also contradicts property 3 which states that the reconstructed secret $s' = s$ if the dealer is honest.

Therefore, these conditions imply that an adversary cannot win either the unconditionally secure verifiability game with a malicious dealer or the unconditionally secure verifiability game with an honest dealer.

4.1.3 Publicly verifiable VSS

Chor, Goldwasser, Micali, and Awerbush [45] were the first to introduce the notion of VSS and they presented a scheme in the computational setting, relying on the difficulty of integer factorization. In this construction, both the privacy and verifiability guarantees are computational. Further, the construction has the property that it is *publicly verifiable* [137]. That is, anyone can verify that shares have been distributed correctly. Essentially, the shares are encrypted via some public-key encryption scheme so that the consistency check can be performed on encrypted shares by entities not holding a share themselves. Such a scheme is necessarily in the computational setting since it relies on the security of the public-key encryption scheme.

A publicly verifiable scheme is relevant in a scenario where the dealer and some subset of participants may be malicious. There is no assumption of secure channels between the dealer and shareholders. We now present a new game-based definition of publicly verifiable VSS. This definition is more formal than the informal model given in previous work [137].

The Public Verifiability Game. Consider a probabilistic polynomial-time adversary. Let $\mathcal{P}$ denote the set of n shareholders and suppose each participant has a valid key pair for asymmetric encryption. Consider a (k, n)-threshold scheme.

Step 1. The adversary generates n (possibly invalid) shares, say $v_1, \ldots, v_n$. Each share is encrypted using the public key of the respective shareholder. The encrypted share, $E(v_i)$, is forwarded to P_i for $i = 1, \ldots, n$. The adversary designates two distinct sets of participants, say A_1, A_2, both of size k.

Step 2. Any honest party verifies the encrypted shares for participants in A_1 and A_2 using a public PubVerify algorithm.

Step 3. If all shares are verified in step 2 then A_1 reconstructs a secret s_1 and A_2 reconstructs a secret s_2.

The adversary wins the game if the shares are not *consistent*, i.e., the following properties hold:

1. all shares corresponding to participants in A_1 and A_2 are verified and accepted, and
2. $s_1 \neq s_2$.

Publicly verifiable schemes are *non-interactive* as we have previously defined the term, since they do not require interaction between shareholders. Since these schemes are non-interactive and require public key encryption, they are necessarily in the computational setting. The same argument as in Theorem 4.1.1 applies if we replace the shares that are input into the verification procedure with encrypted shares (even if these were encrypted using an unconditionally secure symmetric encryption scheme). The validity of shares can be checked before reconstruction because the encrypted shares are publicly verifiable by any party. This can prevent malicious shareholders from corrupting their shares before reconstruction.

Benaloh [14] provided a construction of a publicly verifiable scheme. The scheme was based on Shamir's scheme, so the privacy guarantee is information-theoretic. The verifiability guarantee is computational. Stadler [137] discusses publicly verifiable schemes in more detail, providing an informal definition and several constructions based on discrete logarithms. We present one of the constructions from Stadler [137] next.

Example 4.1.3. [Publicly Verifiable Secret Sharing [137]] Let $\mathcal{P}$ denote the set of n participants and let p be a large prime such that $q = (p-1)/2$ is also prime. Let G be a group of order p and g be a generator such that computing discrete logarithms to the base g is hard. Let $h \in \mathbb{Z}_p^*$ be an element of order q. Let $s \in \mathbb{Z}_p$ be the secret and let $S = g^s$ be publicly known. We will construct a (k, n)-threshold scheme.

Share: A publicly known element $x_i \in \mathbb{Z}_p$, $x_i \neq 0$, is assigned to each participant P_i for $i = 1, \ldots, n$. Each participant chooses a secret key $z_i \in \mathbb{Z}_q$ and publishes their public key $y_i = h^{z_i} \pmod p$. The dealer chooses random elements $f_j \in \mathbb{Z}_p$ for $j = 1, \ldots, k-1$ and publishes $F_j = g^{f_j}$ for each j.

The dealer computes a share for each P_i:

$$s_i = s + \sum_{j=1}^{k-1} f_j x_i^j \pmod{p}.$$

The dealer encrypts s_i using the public key of P_i by randomly choosing some $\alpha_i \in \mathbb{Z}_q$ and calculating the pair

$$(A_i, B_i) = (h^{\alpha_i}, s_i^{-1} \cdot y^{\alpha_i}) \pmod{p}.$$

The dealer sends (A_i, B_i) to participant P_i. The values $S_i = g^{s_i}$ are published so that they can be publicly verified. To decrypt these shares, P_i can compute $s_i = A_i^{z_i}/B_i \pmod{p}$.

Verify: To verify their own share, participant P_i can check that

$$g^{s_i} = S_i = S \cdot \sum_{j=1}^{k-1} F_j^{x_i^j}.$$

For any other participant to verify that (A_i, B_i) is valid, i.e., that it is the encryption of S_i, they proceed with the following interaction with a prover (the dealer). The prover will prove to the verifier that the discrete logarithm of A_i to the base h is identical to the double discrete logarithm of $S_i^{B_i}$ to the bases g and y.

Prover (Dealer)		**Verifier**
repeat K times:		
$w \leftarrow_\$ \mathbb{Z}_q$		
$t_h = h^w \pmod{p}$		
$t_g = g^{(y^w)}$		$c \leftarrow_\$ \{0, 1\}$
	$\xrightarrow{t_h, t_g}$	
	$\xleftarrow{c}$	
$r = w - c \cdot \alpha \pmod{q}$		
	$\xrightarrow{r}$	
		$t_h \stackrel{?}{=} h^r A_i^c \pmod{p}$
		$t_g \stackrel{?}{=} \begin{cases} g^{y^r} \text{ if } c = 0, \\ S_i^{(B_i \cdot y^r)} \text{ if } c = 1 \end{cases}$

Repeating this K times ensures that the prover can successfully cheat with probability at most 2^{-K}. The protocol is proven to be perfectly zero-knowledge in the paper.

Reconstruct: Any group of at least k participants can compute the secret s using Lagrange's interpolation formula.

■

4.1.4 Authenticity

The ability to verify the authenticity of shares is also included in *adept secret sharing* (ADSS) [11]. The authors call this property *authenticity* or *binding*. The definition requires that for a given share from an honest dealer, there is at most one secret s for which it might be a share. In other words, a share acts as a commitment to a message. This property also holds for verifiable schemes. A difference between VSS and authenticity in ADSS is that the verification is built in to the Recover function in ADSS. There is no explicit verification function in an ADSS scheme. In addition, there is no requirement for the dealer to publicly broadcast any values. The previous definitions of VSS do not require that the dealer broadcast any values, but initial constructions often used that strategy. The authors present two different definitions of authenticity, the stronger of which we present next. We present a simplified version of the authenticity game here, to enable comparison with the other definitions we discussed. Although the original definition applies to general access structures, we consider threshold schemes in our definition.

The Authenticity Game. [11] Consider a probabilistic polynomial-time adversary acting as the dealer in a (k, n)-threshold scheme. The adversary generates two sets of n shares $\mathcal{S}, \mathcal{S}'$. The adversary wins the game if there exists authorized sets $\mathcal{V} \subseteq \mathcal{S}$ and $\mathcal{V}' \subseteq \mathcal{S}'$ (i.e., $|\mathcal{V}| \geq k$ and $|\mathcal{V}'| \geq k$) of verified shares (here, the verification is built into reconstruction) such that $\mathcal{V} \cap \mathcal{V}' \neq \emptyset$ and the reconstructed secrets, say s, s' from $\mathcal{V}, \mathcal{V}'$ (respectively) are not equal.

The requirement that $\mathcal{V} \cap \mathcal{V}' \neq \emptyset$ ensures that the sets have some share in common and, if the game cannot be won, that this share *commits* to a specific secret. For this reason, a scheme which satisfies the authenticity property must be in the computational setting. Otherwise, in the information theoretic setting, a single share corresponding to a single secret would suggest that a single share is an authorized set (and this would violate the requirement that $k - 1$ parties learn no information about the secret). This

is because, given a single share that commits to a secret, an adversary with unbounded computational resources can repeatedly guess the remaining $k-1$ shares and submit them to the reconstruction algorithm until it outputs a valid secret. This is guaranteed to be the correct secret because the original share commits to the correct secret.

The verification procedure must be non-interactive since it is built into the reconstruction protocol. Therefore, it does not require interaction with the dealer or between participants. These protocols are not publicly verifiable since verification can only occur during reconstruction.

The paper which introduced authenticity [11] provides a construction of a transformation which takes a classical secret sharing scheme and outputs a scheme which also satisfies authenticity. The rough idea of the construction is that the secret is encrypted into a ciphertext C which is placed into a public portion of each share. Then the encryption key, K, which is derived from a hash of the input (here, the input includes the secret, random information, a string representing an access structure, and possibly a tag), is secret shared using the classical secret sharing scheme. The details involve the use of a pseudorandom function. The verification occurs during reconstruction. Once the input information is reconstructed, we check that the hash output of the reconstructed information is consistent with the given shares. If an adversary could win the authenticity game, it would imply that there exists a collision in the output of the hash function. This construction is said to be much more efficient than previous constructions of verifiable schemes and does not require additional infrastructure changes, such as broadcast channels or interaction between shareholders.

In addition to the authenticity game, the authors in [11] provide a game-based definition of VSS to enable comparison between their new notion of authenticity and verifiability. They define a deterministic algorithm Verify, which takes as input a share and outputs 0 or 1, denoting whether or not the share is valid. For a valid share, the algorithm should output 1. A simplified version of the game is presented below.

The VSS Game. [11] Consider a probabilistic polynomial-time adversary.

Step 1. The adversary generates two sets of shares $\mathcal{S}, \mathcal{S}'$.

Step 2. If $\text{Verify}(S) = 1$ for all $S \in \mathcal{S} \cup \mathcal{S}'$ and both sets of shares contain authorized sets then a secret s_1 is reconstructed using the shares in $\mathcal{S}$ and a secret s_2 is reconstructed using the shares in $\mathcal{S}'$.

The adversary wins the game if $s_1 \neq s_2$.

The authors show that if an adversary cannot win the VSS game with non-negligible probability then they cannot win the authenticity game with non-negligible probability, i.e., this definition of VSS implies authenticity. This general definition of VSS does not specify whether the Verify algorithm is interactive. Aside from this omission, it is essentially the same as the non-interactive verifiability game which we presented in Section 4.1.1. It does not account for potentially malicious shareholders modifying shares before reconstruction; however, this is intentional since that ability is encompassed by the error correction property that is also discussed in the paper [11].

Since this VSS Game is essentially the same as the non-interactive verifiability game that we defined in Section 4.1.1, Pedersen's non-interactive scheme (Example 4.1.1) satisfies this definition of verifiability. This implies that Pedersen's scheme also provides authenticity, as defined above. Although authenticity is a weaker notion than verifiability as it has been previously defined, the provided construction is more efficient than previously suggested VSS schemes and does not make use of broadcast channels (a strategy used in Pedersen's scheme). In adept secret sharing, the verification of shares is built into the reconstruction algorithm. Consequently, the reconstructed secret (valid or not) and the entire set of submitted shares can be used to check validity, as opposed to previous verifiable schemes which enable verification of single shares before reconstruction.

4.1.5 Summary of verifiable schemes

Table 4.1 summarizes different constructions of VSS schemes and their respective properties and guarantees. All of the included protocols provide security against a malicious dealer.

	Non-Interactive	Publicly Verifiable	Malicious Shareholders	Unconditionally Secure
CGMA [45]	✓	✓	✓	
Benaloh [14]	✓	✓	✓	
Feldman [68]	✓			
Pedersen [121]	✓			
Stadler [137]	✓	✓	✓	
ADSS [11]	✓		✓	
Stinson, Wei [141]			✓	✓
Rabin, Ben-Or [127]			✓	✓

Table 4.1: Properties of verifiable secret sharing schemes

4.2 Rational and social secret sharing

4.2.1 Rational secret sharing

Later, in Chapter 5, we will discuss settings with cheating shareholders, where participants are framed as good or bad actors. Rational secret sharing [77] considers a form of misbehaviour that differs from typical notions of cheating. Rather than some parties acting maliciously, they consider *rational* individuals who can be expected to follow the protocol if and only if doing so increases their expected utility. In this scenario, the reconstruction of the secret can be considered in a game-theoretic sense. In particular, players aim to maximize their utility, which is determined by the following two properties:

- Players prefer to learn the secret above all else.
- If they cannot learn the secret, players prefer that the fewest number of other players learn the secret.

Rational secret sharing protocols require that it is optimal for participants to follow the protocol honestly when following the two rules above. We present an example given in [73].

Construction 4.2.1. [Rational secret sharing [73]] We construct a (k, n)-rational secret sharing scheme. Let $\mathbb{F}$ be a finite field and $\mathcal{S} \subset \mathbb{F}$ be the set of valid secrets.

Setup: The dealer prepares shares for a secret $s \in \mathcal{S}$ using Shamir's (k, n)-threshold scheme. Denote the shares s_i for $i = 1, \ldots n$. The dealer also generates a signature σ_i on each share s_i using a publicly known verification key PK. The dealer sends (s_i, σ_i) to player P_i.

Reconstruction: The shareholders proceed as follows:

1. The k participating parties run a secure computation protocol [13] secure against one malicious player. The protocol computes the following probabilistic functionality:
   - Each party inputs the values (s_i, σ_i) received from the dealer. The functionality checks that each σ_i is a valid signature on s_i (with respect to the public key PK), and aborts if this is not the case.

- The k input shares reconstruct to a secret s. With probability β, the functionality generates a fresh (k, n) Shamir sharing s', and each player P_i receives corresponding shares s'_i.
- With probability $1 - \beta$, the functionality generates a fresh (k, n) Shamir sharing of an invalid secret $\tilde{s} \in \mathbb{F} \setminus \mathcal{S}$ and each player P_i receives corresponding shares s_i.

2. If cheating is detected in the secure computation protocol above (i.e., the secure computation protocol is aborted), then parties terminate the overall protocol without ever reconstructing the secret.
3. Next, each player P_i broadcasts the output s'_i they received from the secure computation protocol. To prevent players from broadcasting some modified value, these shares should be authenticated in some way by the functionality (one way to do this is by generating signatures of each share). If this enables reconstruction of a secret $s \in \mathcal{S}$, the protocol terminates and the true secret has been reconstructed. If some player refused to broadcast their output share, then parties terminate the protocol without reconstructing the secret. In any other case, players erase the shares s'_i and repeat the reconstruction process (using (s_i, σ_i) as before).

An appropriate choice of β (depending on k and n) ensures that it is in each player's best interest to correctly follow the protocol. This will maximize their utility, given that they prefer to learn the secret above all else, but otherwise prefer that the fewest number of other players learn the secret. This utility function is specified and a formal game-theoretic proof that the construction is a rational scheme is given in the paper. ■

We note that this construction is only computationally secure because it uses public key cryptography in the form of signatures. The original work by Halpern and Teague [77] worked in an information-theoretic model and assumed the existence of private channels between the players.

4.2.2 Social secret sharing

Under scenarios where players may misbehave it is interesting to introduce the concept of reputations. Social secret sharing schemes were introduced by Nojoumian, Stinson, and Grainger [118]. Informally, shares are allocated based on a player's reputation and how they interact with other participants. In social secret sharing, each participant has a *level*

of importance and a *reputation* which must be balanced. These factors are balanced in an additional stage which we call *tuning*. The tuning mechanism adjusts the weights of each participant's reputation based on their cooperation and past actions. Similar to proactive secret sharing (which we discuss in detail in Section 4.4.1), participants renew their shares in the tuning phase. The authors also provide a verifiable social secret sharing scheme, which is secure against active adversaries.

A social secret sharing scheme involves assigning weights to participants which are representative of their *trust/reputation*. These weights are accounted for in the following ways:

- Participants may have initial *trust* values which determine the *weight* of the initial share they receive. This is similar to weighted threshold schemes, which we discussed in Section 3.4.2.
- Non-cooperative players' reputation/trust values can be decreased if they do no cooperate. If their weights fall below a threshold, such as zero, they may be removed from the scheme.
- Cooperative players' reputation/trust values can be increased over time. Also, new players can be incorporated into the scheme with some initial trust value and given shares via an enrollment scheme (Section 3.4.3).
- Shares are updated using *proactive secret sharing* (discussed in Section 4.4.1) according to the new weights. Players update their shares and disenrolled players do not receive any more shares. After this renewal, old shares become useless.
- Secret recovery is similar to weighted threshold schemes (Section 3.4.2) where a coalition of players can recover the secret if their total reputation value is greater than some threshold.

4.3 Deniability

There are some scenarios in which it may be beneficial to allow participants to perform actions similar to cheating. We consider the concept of deniability: a privacy property of cryptographic protocols that is desirable in cases where a user may want to deny performing some action. Existing work on deniability has focused on deniable encryption [39] which enables a sender of an encrypted message to deny having sent a message and deniable

authentication [60, 57], in which a participant can verify the authenticity of a message at the time it is received, but it cannot be proven to a judge after the action is performed. In addition, deniable authenticated key exchange protocols have been considered, which allow participants to deny having participated in a key exchange protocol [29, 106].

Deniable encryption and deniable authentication do not immediately translate to the context of secret sharing, since we are not directly encrypting/decrypting message, exchanging keys, or authenticating messages. Nonetheless, we can conceive of some scenario in which a participant is being coerced to reveal some information. Suppose Alice and Bob participate in a secret sharing scheme. Say Alice is the dealer and Bob receives one of the shares. Later, an adversary (or a judge), Eve, demands to see all the private information: the secret, the random bits, and Bob's share. In a classical secret sharing scheme, Eve cannot necessarily prove that Bob participated in the secret sharing scheme since his share may correspond to any secret. For example, using Shamir's scheme, a single point does not necessarily tell you anything about the corresponding polynomial/secret. Thus, classical secret sharing schemes inherently provide some property of deniability. However, this property does not necessarily hold for more complex variations of secret sharing. We formalize a definition of deniable secret sharing and explore how it can be achieved.

The notion of deniability has been considered in other cryptographic contexts, but has not been addressed in the area of secret sharing. Therefore, everything that follows in this section is new work. To formalize the notion of deniability, we continue in our approach to using game-based definitions.

4.3.1 Formalizing deniable secret sharing

Assume a (k, n)-Shamir threshold scheme. The shares are $s_i = (x_i, y_i)$, where the x_i's are distinct and non-zero, for $i = 1, \ldots, n$. All the shares lie on a polynomial $r(x)$ of degree $\leq k - 1$ over a finite field F_q. The secret is $r(0)$.

The Deniability Game. Fix k and two non-negative integers e and f.

Step 1. $k + e$ "good shares" are given to the adversary. All these shares lie on the polynomial $r(x)$. Note that $r(x)$ can be computed by interpolating any k of these good shares.

Step 2. The adversary creates f new shares ("bad shares") along with a new polynomial $p(x)$ such that $p(0) \neq r(0)$. There are now a total of $k + e + f$ shares. All of these shares have distinct, non-zero x-coordinates.

The adversary wins the deniability game if the number of shares that lie on $p(x)$ is greater than or equal to the number of shares that lie on $r(x)$. (In this case, the adversary can plausibly claim to an adjudicator that the secret is $p(0)$ rather than $r(0)$.)

If the adversary can win this deniability game, we say that the threshold scheme is (k, e, f)-deniable.

Theorem 4.3.1. *If $f \leq e$, then a (k, n)-threshold scheme is not (k, e, f)-deniable.*

Proof. The proof follows from results in [143, 128]. Denote $G = \{i : y_i = r(x_i)\}$ (the good shares which lie on $r(x)$) and $B = \{i : y_i = p(x_i)\}$ (the bad shares which lie on $p(x)$). We know that $|G \cap B| \leq k - 1$ because $p(x) \neq r(x)$. Then, if $f \leq e$, it follows that $|G| = k + e > k + f - 1 \geq |G \cap B| + f \geq |B|$. Therefore, the number of shares that lie on $p(x)$ is less than the number of shares that lie on $r(x)$. □

Theorem 4.3.2. *If $f > e$, then a (k, n)-threshold scheme is (k, e, f)-deniable.*

Proof. We prove this by describing how the adversary can carry out a successful attack. To illustrate the idea, assume $f = e + 1$.

1. The adversary chooses $k-1$ of the $k+e$ good shares and constructs a bad share (the bad share does not lie on $r(x)$).
2. These k shares are interpolated to obtain the polynomial $p(x)$.
3. Then the adversary constructs e additional bad shares that all lie on $p(x)$, by evaluating $p(x)$ at e new points.
4. The adversary presents the $e + 1 = f$ bad shares along with the polynomial $p(x)$.

Note that $r(x)$ and $p(x)$ agree on $k-1$ points, namely on the $k-1$ good shares selected in step 1 of the attack. So they cannot agree on any more points, and in particular, $r(0) \neq p(0)$.

The total number of shares that lie on $r(x)$ is $k+e$ (none of the bad shares lie on $r(x)$). The total number of shares that lie on $p(x)$ is also $k+e$. Therefore the adversary wins the deniability game. □

In the above game, we only allowed the adversary to construct new shares. The $k+e$ "original" shares cannot be modified. An alternative approach would be to allow the adversary to modify shares as well as to construct new ones. We provide analysis of the extended game below.

The Extended Deniability Game. Fix k, two non-negative integers e and f, and a non-negative integer $g \leq k+e$.

Step 1. $k+e$ "good shares" are given to the adversary. All these shares lie on the polynomial $r(x)$. Note that $r(x)$ can be computed by interpolating any k of these good shares.

Step 2. The adversary modifies g of the $k+e$ original shares such that they lie on a new polynomial $p(x)$ where $p(0) \neq r(0)$.

Step 3. The adversary creates f new shares ("bad shares") along the new polynomial $p(x)$ such that $p(0) \neq r(0)$. There are now a total of $k+e+f$ shares. All of these shares have distinct, non-zero x-coordinates.

The adversary wins the deniability game if the number of shares that lie on $p(x)$ is greater than or equal to the number of shares that lie on $r(x)$ and the number of shares that lie on $p(x)$ is at least k. (In this case, the adversary can plausibly claim to an adjudicator that the secret is $p(0)$ rather than $r(0)$.)

If the adversary can win this deniability game, we say that the threshold scheme is (k, e, f, g)-deniable.

Theorem 4.3.3. *If $2g + f \leq e$, then a (k, n)-threshold scheme is not (k, e, f, g)-deniable.*

Proof. The proof follows from results in [143, 128] and Theorem 4.3.2. Denote $G = \{i : y_i = r(x_i)\}$ (the good shares which lie on $r(x)$) and $B = \{i : y_i = p(x_i)\}$ (the bad shares which lie on $p(x)$). We know that $|G \cap B| \leq k-1$ because $p(x) \neq r(x)$ are polynomials of degree at most $k-1$. If $2g + f \leq e$, then it follows that $|G| = k + e - g > k - 1 + g + f \geq |G \cap B| + g + f \geq |B|$. Therefore, the number of shares that lie on $p(x)$ is less than the number of shares that lie on $r(x)$. □

Theorem 4.3.4. *If $e \geq g$ and $2g + f > e$, then a (k, n)-threshold scheme is (k, e, f, g)-deniable.*

Proof. We prove this by describing how the adversary can carry out a successful attack. To illustrate the idea, assume $g + f - 1 = e - g$ and $e \geq g$.

1. The adversary chooses $k-1$ of the $k+e$ good shares and constructs a bad share (the bad share does not lie on $r(x)$).

2. These k shares are interpolated to obtain the polynomial $p(x)$.

3. Then the adversary constructs $g+f-1$ additional shares that all lie on $p(x)$, by evaluating $p(x)$ at e new points.

4. The adversary replaces g of the $k+e$ good shares with the bad shares and presents the remaining f along with the polynomial $p(x)$.

Note that $r(x)$ and $p(x)$ agree on $k-1$ points, namely on the $k-1$ good shares selected in step 1 of the attack. So they cannot agree on any more points, and in particular, $r(0) \neq p(0)$.

The total number of shares that lie on $r(x)$ is $k+e-g$. We have that $k+e-g \geq k$ since $e \geq g$. The total number of shares that lie on $p(x)$ is also $k+e-g$ since $k-1+f+g = k+e-g$. Therefore the adversary wins the deniability game. □

The definitions we have presented describe a deniability property for secret sharing that is analogous to deniable encryption. These arguments demonstrate that classical threshold schemes inherently enable participants with control over a sufficient number of shares to deny holding shares corresponding to a particular secret.

4.4 Leakage Protection

In this section, we consider a scenario in which an adversary may obtain leaked information on shares. This could include leaked shares or leaked parts of shares. *Proactive secret sharing* and *leakage resilient secret sharing* address these vulnerabilities, respectively. In what follows, we elaborate on the threat models, definitions, and some examples.

4.4.1 Proactive secret sharing

Classical secret sharing schemes are vulnerable to adversaries who can gradually compromise shares over time. If the sensitive information is long-lived then we may require more protection to prevent adversaries from learning shares over a period of time. Proactive secret sharing [80] allows us to periodically renew shares without changing the secret. Old

shares become obsolete after their respective time period, making them unusable to an adversary who may have compromised the share at some point in time.

In proactive secret sharing, the lifetime of a secret is split into periods of time and at the beginning of each period of time, the shareholders update their shares via an interactive protocol. The dealer is only required for the initialization phase and is not required for any updates. In the case of a (k, n)-threshold scheme, an adversary must compromise k shares in a short period of time in order to recover the secret. We present an example of a proactive secret sharing scheme from Herzberg et al. [80]. This work exists in the computational setting, but later work addressed proactive secret sharing in an information-theoretic setting [141].

Example 4.4.1. [Proactive secret sharing scheme [80]]

Share: A dealer shares a secret $s \in \mathbb{Z}_q$ using Shamir's (k, n)-threshold scheme. Let $x_i = f(i)$ denote the share belonging to participant P_i for $i = 1, \ldots, n$, where f is some polynomial of degree $k - 1$.

Renewal: We denote the shares for some time period t by $x_i^{(t)}$ and the corresponding polynomial by $f^{(t)}$. Each P_i picks $k - 1$ random numbers $\{\delta_{im}\}_{m \in \{1 \ldots k\}}$ from $\mathbb{Z}_q$. These numbers define a polynomial $\delta_i(z) = \delta_{i1}z + \cdots + \delta_{i(k-1)}z^{k-1}$ with constant term 0. P_i sends $u_{ij} = \delta_i(j) \pmod q$ to P_j across a secure channel for $j \in \{1, \ldots, n\} \setminus \{i\}$. Each P_i computes their new share $x_i^{(t)} = x_i^{(t-1)} + \sum_{j=1}^{n} u_{ji} \pmod q$ and erases all the variables used except the current share $x_i^{(t)}$. These new shares correspond to the sum of polynomials with constant term 0 and f which has constant term s.

Reconstruct: Reconstruction is accomplished using polynomial interpolation, as in Shamir's scheme.

This protocol is secure against a passive adversary who may learn secret information available to up to $k - 1$ corrupted servers but who follows the protocol. ■

The original work on proactive secret sharing provides a construction for share renewal in the presence of active adversaries. They also provide mechanisms to detect corrupted (or lost) shares and are able to restore the correct share if necessary. The authors describe a method for a share to be recovered, analogous to repairable secret sharing, which we discussed in Section 3.3.2.

4.4.2 Leakage resilient secret sharing

In addition to accounting for the possibility of shares being leaked over time, we can consider adversaries who obtain some bounded amount of leakage from each share. Rather than obtaining the entire share, the adversary may receive a portion of a share. A *leakage resilient secret sharing scheme* is one in which the adversary learns no information about the secret despite having some bounded amount of leakage from every share. The concept was introduced in two independent works [74, 16] and explored further in [136, 101].

We present a new game-based definition in an information-theoretic setting which simplifies the original definition of leakage resilient secret sharing given by Goyal and Kumar [74].

The $(\epsilon, \mathcal{F})$-Leakage Resilience Game. Assume a (k, n)-threshold scheme. Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a set of functions that take as input a share. $\mathcal{F}$ is the set of leakage functions.

Step 1. An honest dealer generates n shares for a secret s, denoted $s_1, \ldots, s_n$, and n shares for a secret $\tilde{s}$, denoted $\tilde{s_1}, \ldots, \tilde{s_n}$.

Step 2. The adversary obtains $f_i(s_i)$ and $f_i(\tilde{s_i})$ for each $i = 1, \ldots, n$.

The adversary wins the $\mathcal{F}$-leakage resilience game if they can distinguish between the shares of s and $\tilde{s}$ with probability greater than ϵ. If the adversary cannot win the leakage resilience game, except with negligible probability, we say that the scheme is *ϵ-leakage resilient with respect to $\mathcal{F}$*.

This is a very generic definition, but it encompasses the intuition behind leakage resilient schemes. Suppose, for example, that $\mathcal{F}$ contained a set of functions where $k-1$ functions were the identity function and the remaining functions returned 0 for every input. Then the game would reduce to the privacy property of a (k, n)-threshold scheme. On the other hand, we can consider more complex leakage scenarios. For example, each f_i could output the least significant μ bits of a share, where the size of the share is larger than μ. An adversary in this setting may obtain limited information on each share and ideally, learn no additional information on the secret.

A reasonable goal in leakage resilient schemes is to achieve what is called *local leakage resilience*.

Definition 4.4.1 (Local Leakage Function Family [136]). *Let $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ be the domain of shares for a secret sharing scheme and suppose we consider (k, n)-threshold schemes. The corresponding local leakage function family is*

$$\mathcal{F}_{k,\mu} = \{f_{K,\tau} : K \subset [n], |K| \leq k-1, \tau_i : \mathcal{S}_i \mapsto \{0,1\}^\mu\}$$

where $f_{K,\tau}$ on input $(s_1, \ldots, s_n)$ outputs s_i for every $i \in K$ and outputs $\tau_i(s_i)$ otherwise. That is, an adversary obtaining the output of this function receives at most $k-1$ shares in plaintext and for every other share, they receive μ bits of information derived from the share.

Example 4.4.2. Recall the additive secret sharing scheme from Construction 2.0.2. An attack is described in [16] demonstrating that over finite fields of small characteristic, such as $\mathbb{F}_{2^\kappa}$, the additive scheme does not provide local leakage resilience. Suppose shares $s_1, \ldots, s_n$ are generated using the additive scheme for a secret $s \in \mathbb{F}_{2^\kappa}$. Consider the case where the adversary receives the least significant bit of each share s_i for $i = 1, \ldots, n$. Adding them up, the adversary can reconstruct the least significant bit of the secret s. A similar attack for Shamir's scheme over a field of small characteristic is given in [16]. ■

The authors of [16] also demonstrate that when the underlying field is of a large prime order and the number of parties is sufficiently large, both the additive secret sharing scheme and Shamir's scheme are secure against local leakage attacks. Informally, they show that for a prime p, there exists a constant $c_p < 1$ such that, for sufficiently large n, the additive secret sharing scheme over $\mathbb{F}_p$ is locally leakage resilient when $\lfloor (\log p)/4 \rfloor$ bits are leaked from every share. In fact, they prove that the local leakage resilience property holds for any linear secret sharing scheme constructed from a linear code with appropriately chosen parameters.

4.5 Summary

In this chapter, we began our investigation of alternative adversarial settings for secret sharing schemes. In a setting where the dealer is not trusted, it is beneficial to use schemes which provide some form of *verifiability*. We found in our analysis that much of the work in verifiable secret sharing used slightly different adversarial models without clearly illustrating their differences. At the same time, some adversarial models were never formally defined. We corrected these omissions in the literature by presenting game-based definitions of verifiability in a unified manner. We also discussed protections against leakage

in secret sharing schemes and provided a new game-based definition for leakage resilient schemes.

In contrast to verifying that participants are correctly participating in a protocol, we noted that it is sometimes of interest to allow participants to deny having participated in a secret sharing. This spurred the formalization of deniability in the context of secret sharing. We provided formal definitions of deniability and analyzed how they can be achieved.

We briefly addressed the potential of misbehaving shareholders in the concepts of rational and social secret sharing. In the following chapter, we discuss this setting in more depth. In particular, we analyze notions of robustness which aim to protect against different forms of cheating participants.

Chapter 5

Notions of Robustness

Classical secret sharing assumed a threat model in which the shareholders were honest. These assumptions were first challenged by Tompa and Woll [146]. They consider the presence of malicious shareholders who submit potentially corrupted shares to the reconstruction mechanism. Consider, for example, a Shamir threshold scheme and suppose some participant submits a value different than the share they were given. This attack simultaneously prevents the honest participants from learning the secret and allows the adversary to obtain the correct secret without being detected. Following the introduction of this attack, there have been several works studying potential notions of robustness that can be used to prevent such attacks.

In this chapter, we review several notions related to robustness in secret sharing. In particular, we investigate settings in which the shareholders may be dishonest, but the dealer is honest. Thus, our review does not encompass verifiable secret sharing, which considers the presence of a dishonest/malicious dealer. The goal is to present a unified framework for various notions of robustness in secret sharing schemes. The literature in the area of secret sharing which assumes the presence of dishonest shareholders has grown to encompass several different notions, some of which overlap in motivations and constructions. We aim to clarify the commonalities and differences of each notion in this section. Further, we make use of a unified manner of defining these notions so that they are easier to compare to one another.

In summary, in the setting of unconditionally secure threshold schemes where the dealer is assumed to be honest, we explore the many methods of thwarting cheating participants:

Making cheating impossible. Invalid shares do not inhibit the reconstruction of the

correct secret. This is the goal of *robust secret sharing*, *error decodable secret sharing*, *error correction*, and *non-malleability*.

Share verification. Invalid shares are identified and discarded. Similar strategies are used in *cheater identification* and *cheater detection*.

Discouraging cheaters from sending invalid shares. This strategy is effective if cheating participants gain no advantage over honest participants, i.e., sending invalid shares results in the recovery of an invalid secret. This is explored in *cheating-immune secret sharing* and *fairness schemes*.

5.1 Robust secret sharing

We review the notion of robust secret sharing suggested by Tompa and Woll [146]. This was the first paper to challenge the adversarial model of classical secret sharing. The work was motivated by the potential of malicious shareholders in a secret sharing scheme. In the following scenario, we assume that the dealer is an honest participant, but the participants who receive shares may act maliciously. The definition which follows is a new game-based definition, although it is fairly obvious to derive from the description given by Tompa and Woll [146].

The Robustness Game. Assume a (k, n)-threshold scheme. The secret is s. Fix a non-negative integer $1 \leq t < k$. We assume all secrets are equally probable.

Step 1. t of the n shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. A secret s' is reconstructed using the t "bad shares" and $k - t$ of the original "good shares". The adversary may choose which of the "good shares" are used in reconstruction. The adversary wins the robustness game if the reconstructed secret s' is a valid secret and $s' \neq s$.

Typically, we let $t = k - 1$. Then, if the adversary can only win this game with some small probability $\epsilon > 0$ (this is the cheating probability), we say that the threshold scheme is ϵ-*robust*.

Remark 5.1.1. *When we say the adversary "may choose which of the 'good shares' are used in reconstruction," this does not mean that the adversary gets to see the value of the share. The intention is that the adversary is able to select some identifiers denoting which shares to use in reconstruction. For example, in Shamir's scheme, it is sufficient to allow the adversary to select certain x-coordinates corresponding to shares. Knowing these x-coordinates allows the adversary to use their knowledge of publicly known Lagrange coefficients to perform attacks which would otherwise be impossible if they had no control over which shares were used during reconstruction. This is demonstrated in Theorem 5.1.1.*

The game-based definition of robustness which we have proposed is equivalent to the concept of robustness defined by Tompa and Woll [146]. They also demonstrate that Shamir's scheme is not robust.

Theorem 5.1.1. *A (k, n)-Shamir threshold scheme is not robust.*

Proof. We prove this by describing how the adversary can carry out a successful attack by modifying only a single share. Suppose the shares $v_1, \ldots, v_k$ are used to reconstruct a polynomial.

The true secret, s, can be written as $s = \sum_{i=1}^{k} a_i v_i$ where the a_i's denote the Lagrange coefficients and the v_i's are the good shares, for $i = 1, \ldots, k$.

Assume that the adversary modifies a single share v_1 by adding some δ. That is, upon reconstruction, the adversary submits $v_1' = v_1 + \delta$.

The reconstructed secret will be

$$s' = a_1 v_1' + \sum_{i=2}^{k} a_i v_i = s + a_1 \delta.$$

The reconstructed secret is valid since there are no restrictions on the secrets in Shamir's original scheme. As long as $\delta \neq 0$, we have $s' \neq s$, so the adversary wins the game. □

There exist many constructions of robust secret sharing schemes. One approach, suggested by Ogata and Kurosawa [120] is to use *difference sets*.

Definition 5.1.1 ([138])**.** *Suppose $(G, +)$ is a finite group of order v with identity element 0. Let ℓ and λ be positive integers such that $2 \leq \ell < v$. A* (v, ℓ, λ)-difference set *in $(G, +)$ is a subset $D \subset G$ such that*

- $|D| = \ell$,
- *the multiset* $[x - y : x, y \in D, x \neq y]$ *contains every element in* $G \setminus \{0\}$ *exactly* λ *times.*

Example 5.1.1. A $(15, 7, 3)$-difference set in $(\mathbb{Z}_{15}, +)$ is $\{0, 1, 2, 4, 5, 8, 10\}$. ■

It is known that a $(\frac{q^{i+2}-1}{q-1}, \frac{q^{i+1}-1}{q-1}, \frac{q^i-1}{q-1})$-difference set exists whenever q is a prime power and i is a positive integer [92].

Construction 5.1.1. Let $\mathcal{P}$ be a set of n players. Let D be a $(v, \ell, 1)$-difference set in $(\mathbb{Z}_v, +)$ where $v = \ell(\ell - 1) + 1$ and $\ell - 1$ is a prime or a prime power. Let the secret space be D. For a given secret $s \in D$, the share and recover algorithms are as follows.

- **Share:** Select $k - 1$ values $r_1, r_2, \ldots, r_{k-1}$ uniformly at random from $\mathbb{Z}_v$ and let $f \in \mathbb{Z}_v[x]$ be the polynomial defined by

  $$f(x) = r_{k-1}x^{k-1} + r_{k-2}x^{k-1} + \cdots + r_1 x + s.$$

  Give player i the share $v_i = f(i)$ for all $i \in \{1, \ldots, n\}$.
- **Recover:** A collection of k players perform polynomial interpolation to recover a value $\tilde{s}$. If $\tilde{s} \in D$, they accept $\tilde{s}$ as the secret. Otherwise, they output $\perp$.

■

In this construction, the set of possible secrets is the difference set D. Thus, given a secret space of size $\ell = |D|$, one can define a public bijection between the secret messages and the elements in the difference set. This bijection can then be used before sharing the secret to ensure that it is an element of the set D and after reconstruction to obtain the corresponding secret.

Theorem 5.1.2. *The proposed construction is a* $1/\ell$*-robust* (k, n)*-threshold scheme.*

Proof. We just show that the above construction satisfies $1/\ell$-robustness. The fact it is a (k, n)-threshold scheme follows immediately from recognizing that it is a straightforward modification of Shamir's scheme.

Recall that we can write the true secret s as $s = \sum_{i=1}^{k} a_i v_i$ where the a_i's denote the Lagrange coefficients and the v_i's denote the good shares used in reconstruction.

Suppose the adversary modifies a single share by adding some value δ. That is, they modify v_1 and submit $v_1' = v_1 + \delta$ instead. Then, the reconstructed secret would be

$$s' = a_1 v_1' + \sum_{i=2}^{k} a_i v_i = s + a_1 \delta.$$

Note that it is possible for the adversary to modify more than one share. They may choose to add values $\delta_1, \ldots, \delta_{k-1}$ to some set of $k-1$ shares under their control. This does not, however, increase their probability of winning the game, as the result is essentially the same. The reconstructed secret would simply be the sum of the secret and a linear combination of the δ_i's. Therefore, for simplicity in this proof, we assume the adversary only modifies a single share.

Now, by the definition of a difference set, there exists a unique pair $x_1, x_2 \in D$ such that $x_2 - x_1 = a_1\delta$. Therefore, the adversary wins if and only if $s = x_1$, for some $s' = x_2$. Since we assume that all secrets are equally as probably, the probability that this occurs is $1/|D| = 1/\ell$. So the proposed construction is $1/\ell$-robust. □

Ogata and Kurosawa [120] discuss bounds on the probability of an adversary successfully cheating in robust schemes. In doing so, they observe that the adversary gains no advantage by modifying more than a single share. Suppose the set of secrets is $\mathcal{S}$, the true secret is s, and the adversary controls $k-1$ shares denoted by $v_1, \ldots, v_{k-1}$. For every $s' \in \mathcal{S}$ where $s' \neq s$, there exists at least one possible share v_1' such that replacing v_1 with v_1' in the reconstruction algorithm results in s' being reconstructed. This follows from the privacy property that requires that the $k-1$ shares yield no information on the value of the secret. Therefore, an adversary gains no advantage from modifying the remaining $k-2$ shares, because they can achieve any desired change by modifying a single share. This property holds for perfect secret sharing schemes. It is interesting to consider whether or not this holds when we consider other notions of robustness.

From the attack by Tompa and Woll, we can also observe that in a classical threshold scheme, it is sometimes possible for adversaries to determine the original secret after reconstruction in addition to corrupting the original secret. We can extend the definition of robustness to consider this possibility. This requires two simple changes to the robustness game.

- The adversary is given the *identities of the good shares* which will be used in reconstruction (in addition to the information provided in the original game).

- The adversary wins the game if, in addition to the requirements of the original robustness game, they can uniquely determine the original secret s.

If the adversary can only win this robustness game with some small probability $\epsilon > 0$, the threshold scheme may also be called ϵ-*robust*.

Robust secret sharing was first suggested in the computational setting by Krawcyzk [100]. The primary goal of this paper was to reduce the size of shares via relaxing the information-theoretic requirements to computational requirements. No formal definitions or proofs were provided on the topic of robustness. Follow-up work by Bellare and Rogaway [129] examined robust computational secret sharing (RCSS) more formally. The requirements of RCSS are similar to those that we have presented above, i.e., the ideas suggested by Tompa and Woll. The main difference is that Tompa and Woll work in an information-theoretic setting while Krawcyzk, and later Bellare and Rogaway, consider the computational setting.

5.2 Error decodability

Error decodable secret sharing was proposed by Kurosawa [102]. A secret sharing scheme is error decodable if it is possible to recover the correct secret from the set of *all shares* even if some of the shares are corrupted by an adversary.

As before, we are considering a scenario in which the dealer is honest. The motivation for this notion is the possibility that some shares may be corrupted during transmission/communication. Thus, we want to be able to recover the secret correctly from a set of potentially noisy shares. This can be modelled in a similar way to the robustness definition, where some shareholders are dishonest. Whether a share is intentionally or unintentionally corrupted does not have much of an effect on the definitions. Here, however, the technique is slightly different than that of robust secret sharing. Error decodable schemes typically require additional shares, above what is strictly required of the threshold scheme, in order to perform error correction. So, the definition of error decodability takes into account all n shares when reconstructing, using the redundancy to make it possible to recover the original secret. Next, we propose a new game-based definition for error decodable secret sharing, based on the ideas from Kurosawa [102].

The Error Decodability Game. Assume a (k, n)-threshold scheme. The secret is s. Fix a non-negative integer $1 \leq t < k$.

Step 1. t of the n total shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. An error decoding algorithm takes as input the n shares and outputs a secret s' or outputs $\perp$ if it fails.

The adversary wins the error decodability game if $s' \neq s$ or the error decoding algorithm fails.

Typically, we let $t = k - 1$. If the adversary cannot win this game, we say that the threshold scheme is *error decodable*.

Remark 5.2.1. *This definition can be generalized to any access structure (other than threshold schemes). In any case, we give the adversary an unauthorized set of shares which they may corrupt. The win condition does not change.*

The definitions of robustness and error decodability are equivalent aside from Step 3. In Step 3, n shares are used to reconstruct a secret in the error decodability game whereas k shares are used in the robustness game, defined in Section 5.1. Also, the adversary can win if the error decoding algorithm fails, which is not a possibility in the robustness game since the reconstruction algorithm only takes as input k shares. Using our game-based definitions of robustness and error decodability, we can observe that if a (k, n)-threshold scheme is robust, it must also be error decodable.

Certain threshold schemes inherently achieve the property of error decodability. This can be observed by recognizing the relationship between a (k, n)-threshold scheme and Reed-Solomon codes [112].

Definition 5.2.1 (Reed-Solomon Codes)**.** *Let q be a prime power and n a positive integer with $n \leq q + 1$. Let $\alpha_1, \ldots, \alpha_n$ be distinct elements in the finite field $GF(q) \cup \{\infty\}$. Let $\mathcal{C}$ be the length n code over $GF(q)$ defined by setting*

$$\mathcal{C} = \{f(\alpha_1), \ldots, f(\alpha_n) \mid f \in GF(q)[x], \deg f < k\}$$

where $f(\infty)$ is the coefficient of x^{k-1} in f. Then $\mathcal{C}$ is an $[n, k, n - k + 1]$ Reed-Solomon code.

Theorem 5.2.1. *[112] A (k, n)-threshold scheme is error decodable if and only if $n - k \geq 2(k - 1)$.*

Proof. Suppose that a secret vector is in the set $GF(q)^k$. Then the possible shares make up a $[n, k, n-k+1]$ Reed-Solomon code. The minimum distance of a Reed-Solomon code is $n-k+1$, so if $n-k \geq 2e$ then the code can be used to correct e errors. Therefore, given a vector of shares corresponding to participants of a (k, n)-threshold scheme, we can recover the corresponding secret if up to e shares have been corrupted. Thus, a $[n, k, n-k+1]$ Reed-Solomon code can correct $k-1$ errors if and only if $n-k \geq 2(k-1)$. This implies that a (k, n)-threshold is error decodable, i.e., we can recover from $k-1$ corrupted shares, if and only if $n-k \geq 2(k-1)$, that is, $n \geq 3k-2$. □

We present Kurosawa's construction [102] of an error decodable secret sharing scheme. First, we review some preliminary definitions. Let $\mathcal{P}$ be a set of n participants and let $2^{\mathcal{P}}$ denote the set of all subsets of $\mathcal{P}$. A collection $\Sigma \subseteq 2^{\mathcal{P}}$ is said to be a *monotone access structure* if Σ contains all sets $A' \in 2^{\mathcal{P}}$ such that $A \subseteq A'$ for some $A \in \Sigma$. A secret sharing scheme *realizing* an access structure Σ is a secret sharing algorithm where the subsets in Σ are the authorized sets and subsets in the complement of Σ are the unauthorized sets. We denote the complement of Σ as Σ^c.

Kurosawa's scheme is based on linear secret sharing schemes. For any monotone access structure Σ, we can construct a corresponding linear secret sharing scheme. Let $M = \{\vec{m}_1, \ldots, \vec{m}_\ell\}$ be an $\ell \times d$ matrix over a finite field $GF(q)^d$ and $\phi : \{1, \ldots, \ell\} \to \{1, \ldots, n\}$, a labelling function where $\ell \geq d, n$. We say that (M, ϕ) is a *linear secret sharing scheme realizing* Σ if the vector $(1, 0, \ldots, 0) \in \text{span}\{\vec{m}_i \mid \phi(i) \in B\}$ if and only if $B \in \Sigma$.

Construction 5.2.1. [Linear Secret Sharing Scheme [84]] Let (M, ϕ) be as defined above. Let $\mathcal{S} = GF(q)$ denote the secret space.

- **Share:** For a secret $s \in \mathcal{S}$, the dealer chooses a random vector $\vec{r} \in \mathcal{S}^{d-1}$ and computes

$$v = M \times \begin{pmatrix} s \\ \vec{r} \end{pmatrix} = (v_1, \ldots, v_\ell)^T. \tag{5.1}$$

  Then, the dealer computes a vector of shares $(u_1, \ldots, u_n)$ where $u_i = \{v_j \mid \phi(j) = i\}$. The dealer gives u_i to participant P_i for $i = 1, \ldots, n$.

- **Recover:** For a set of participants A,if A is an authorized set then there exists $\alpha_j \in GF(q)$ such that $\sum_{\{j|\phi(j)\in A\}} \alpha_j \vec{m}_j = (1, 0, \ldots, 0)$. Then, $\sum_{\{j|\phi(j)\in A\}} \alpha_j v_j = s$, so the participants of A can recover the secret s.

■

Kurosawa uses this linear secret sharing scheme as a building block in constructing an error decodable scheme. He demonstrates that a perfect secret sharing scheme is error decodable if and only if the access structure, Σ, satisfies the condition Q^3 [102].

Definition 5.2.2. *A monotone access structure Σ for a set of participants $\mathcal{P}$ satisfies Q^3 if $B_1 \cup B_2 \cup B_3 \neq \mathcal{P}$ for any $B_1, B_2, B_3 \in \Sigma^c$.*

Example 5.2.1. Consider a $(2, 6)$-threshold scheme corresponding to a set of six participants $\mathcal{P}$. The unauthorized sets, Σ^c, consist of all subsets of one or fewer participants in $\mathcal{P}$. Thus, for any $B_1, B_2, B_3 \in \Sigma^c$, $B_1 \cup B_2 \cup B_3$ contains at most three participants in $\mathcal{P}$, so the threshold scheme satisfies Q^3.

On the other hand, if we take a $(3, 6)$-threshold scheme, the Q^3 property does not hold since there exist three subsets of two participants such that their union is the set of all participants.

In fact, a (k, n)-threshold scheme satisfies property Q^3 if and only if $n > 3(k-1)$. This is the same property that is required to construct an error decodable scheme from a threshold scheme in Theorem 5.2.1. ■

Construction 5.2.2. [Kurosawa's Error Decodable Scheme [102]] Let $\mathcal{P}$ be a set of n players. Let the secret space be denoted by $\mathcal{S} = GF(q)$. Let (M, ϕ) be the linear secret sharing scheme realizing an access structure Σ which satisfies Q^3.

For a secret $s \in \mathcal{S}$, the share, recover, and error decoding algorithms are as follows.

- **Share:** The linear secret sharing scheme (M, ϕ) is used to generate a share vector $(v_1, \ldots, v_\ell)$ for the secret s according to Equation (5.1). For $i = 1, \ldots, \ell$, the dealer treats v_i as a secret and generates a new share vector $\vec{u}^i$ using the scheme (M, ϕ). Then v_i is allocated to participant $\phi(i)$ and $\vec{u}^i_j$ is allocated to participant $\phi(j)$ for each $j = 1, \ldots, \ell$. That is, participant m receives the share $\bigcup_{i=1}^{\ell} \{\vec{u}^i_j \mid \phi(j) = m\} \cup \{v_j \mid \phi(j) = m\}$.
- **Recover:** Recover is easily accomplished by running the reconstruction algorithm of Construction 5.2.1 to recover each v_k and then repeating to recover s.
- **Error Decoding:** For $i = 1, \ldots, \ell$, the participant holding v_i generates the share vector of v_i using (M, ϕ) and compares with the share vectors (that is, $\vec{u}^i$) held by the other participants. If the vectors differ in positions corresponding to some authorized set, then the share v_i is deemed to be corrupt. If the set of users with corrupted shares is in some unauthorized set in Σ^c, then the remaining participants use their shares to reconstruct s. Otherwise, the output is $\perp$.

■

Essentially, the second sharing step ensures that the validity of the initial sharing step can be checked. This construction requires larger share sizes due to the additional values that each participant must hold. Later work suggested a more efficient error decodable scheme with smaller share sizes [109]. More recent work considered how to construct error decodable schemes in a setting with an asynchronous communication model where messages may be arbitrarily delayed [46].

Error decodable secret sharing is closely related to one-round *perfectly-secure message transmission* (PSMT). Perfectly-secure message transmission [58] considers a model in which there are two users, Alice and Bob, connected by some number of distinct communication channels, some of which may be controlled by an adversary. Alice sends information to Bob over some communication channels which may be controlled by the adversary. A one-round PSMT scheme allows Alice to send a message to Bob in such a way that the adversary learns no information about the message by eavesdropping on the channels that they control and the adversary cannot prevent Bob from recovering the message. PSMT schemes were first introduced for threshold adversarial structures where the adversary may control a specified number of channels [58].

Definition 5.2.3. *A* one-round (n, k)-PSMT *scheme is an algorithm permitting a sender to transmit a message s to a receiver over n channels such that:*

- *the receiver can recover s even if an adversary modifies the information from up to t channels;*
- *an adversary eavesdropping on up to t channels learns no information about s.*

The introductory work demonstrated that a (n, k)-PSMT scheme exists if and only if $n \geq 3k + 1$ [58]. Desmedt, Wand, and Burmester investigated PSMT schemes secure against arbitrary monotone adversary structures [56]. Martin, Paterson, and Stinson studied the relationship between one-round PSMT schemes and error decodable secret sharing schemes [109]. They show that a Γ-error decodable secret sharing scheme realizing an access structure Σ (the adversary structure in this case is the set Γ) implies a (Γ, Σ^c)-PSMT scheme, where Γ denotes the subsets of channels an adversary may actively control and Σ^c denotes the subsets of channels an adversary may passively observe. However, the opposite direction does not hold. A (Γ, Σ^c)-PSMT scheme cannot be transformed into a Γ-error decodable secret sharing scheme. Kurosawa provides a construction of a one-round PSMT protocol based on error-decodable secret sharing [102] and later work by Choudhury provides a construction of one-round PSMT in an asynchronous setting [46].

5.3 Cheating immunity

Cheating-immune secret sharing was suggested by Pieprzyk and Zhang [124] as an alternative method of handling cheating participants. The dealer is assumed to be honest and participants may cheat during reconstruction by submitting an incorrect share. A scheme is said to be cheating-immune if the cheating participants have no advantage over the honest participants in determining the true secret.

The Cheating Immunity Game. Assume an (n, n)-threshold scheme. The secret is s, a value in the finite field $GF(p^r)$ where p is prime and r is a positive integer. Also, assume secrets are distributed uniformly at random. Fix $1 \leq t < n$.

Step 1. t of the n total shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. A secret s' is reconstructed using all n shares and all parties learn the reconstructed secret s'.

We say that the *probability of successful cheating* is the probability that the adversary can guess the true secret, s, given the t bad shares used in reconstruction, the t original shares given to the adversary before they were modified, and the reconstructed secret s'. If $s' \neq s$ and the probability of successful cheating is greater than p^r (which is the probability an honest participant guesses the correct secret), then the adversary wins.

If the adversary cannot win this game, we say that the threshold scheme is *t-cheating-immune*. If $t = 1$, we say that the scheme is cheating-immune. Most examples in the literature examine the case where $t = 1$.

The goal of cheating-immune secret sharing is different than that of robust secret sharing because rather than attempting to *prevent* cheating, we aim to *discourage* cheating. In this setting, participants may cheat; however, the resulting reconstructed share should not reveal more information to the cheaters than it does to the honest participants. Consider, for example, Shamir's threshold scheme. We demonstrated in Theorem 5.1.1 that Shamir's threshold scheme is not robust. It is easy to see from the same proof that a (k, n)-Shamir threshold scheme is also not cheating-immune. Given that a cheating adversary submits a corrupt share $v_1' = v_1 + \delta$, the reconstructed secret becomes $s' = s + a_1\delta$ where a_1 is some publicly known Lagrange coefficient. The cheating participant in this case learns the

original secret $s = s' - a_1\delta$ while the honest participants only learn the incorrect secret. Thus, we have observed the following result.

Theorem 5.3.1. *An (n, n)-Shamir threshold scheme is not cheating-immune.*

In fact, D'Arco, Kishimoto, and Stinson show that any perfect secret sharing scheme cannot be cheating-immune [50].

Now we present a secret sharing scheme from [50] that is cheating-immune. Recall that we are working in the finite field $GF(p^r)$. Let $b_p^+ a$ denote the sum of $\lceil p/2 \rceil$ elements equal to a and $b_p^- a$ denote the sum of $\lfloor p/2 \rfloor$ elements equal to a for some $a \in GF(p^t)$. For $a = 1$ (the identity), we just write b_p^+ or b_p^-. For example, if $p = 3$ and $a = 1$ then $b_p^+ = 2$ and $b_p^- = 1$. To present our cheating-immune scheme, we will use a *defining function*. That is, a function $f : GF(p^r)^n \to GF(p^r)$ which associates each n-tuple of shares with a secret in $GF(p^r)$.

Definition 5.3.1. *A function $f : GF(p^r)^n \to GF(p^r)$ is* balanced *if, for each $K \in GF(p^r)$, it holds that*

$$|\{x \in GF(p^r)^n \mid f(x) = K\}| = p^{r(n-1)}.$$

That is, each value $f(x) \in GF(p^r)$ has the same number of pre-images x.

Before continuing, we introduce some notation. Let $\delta \in GF(p^r)^n$ be a vector representing cheaters where the non-zero elements represent the change in the true shares. Let HW denote the Hamming weight of a vector. Given two vectors, x and δ, let x_δ^+ denote the vector such that $x_j^+ = x_j$ if $\delta_j \neq 0$ and $x_j^+ = 0$ otherwise. Conversely, let x_j^- denote the vector such that $x_j^- = x_j$ if $\delta_j = 0$ and $x_j^- = 0$ otherwise. That is, if δ represents the cheaters then x_j^- is the vector containing the shares of honest participants not modified by cheaters and zeros in the place of modified shares. Similarly, x_j^+ is the vector containing the shares of participants which are modified by cheaters *before modification* and zeros in the place of the unmodified shares. Finally, we write $\tau \preccurlyeq \delta$ if $\tau_j \neq 0$ implies $\delta_j \neq 0$.

Definition 5.3.2 ([122])**.** *A function h of degree two is said to have the property $B(k)$ if, for any $\delta \in GF(p^r)^n$ with $1 \leq HW(\delta) \leq k$ and for any $\tau \preccurlyeq \delta$, the function $h(x_\delta^- + \delta + \tau) - h(x_\delta^- + \tau)$ is a non-constant affine function.*

Example 5.3.1. Let $p = n = 3$ and $k = 1$. Let $\mu_{(3,3)} : GF(3)^3 \to GF(3)$ be defined as follows:

$$\mu_{(3,3)}(x_1, x_2, x_3) = x_1 + x_1x_2 + 2x_2x_3 + x_3x_1 + 2x_1^2.$$

Then, the property $B(1)$ requires that for any $\delta \in GF(3)^3$ with $HW(\delta) = 1$ and for any $\tau \preccurlyeq \delta$, the function $\mu_{(3,3)}(x_\delta^- + \delta + \tau) - \mu_{(3,3)}(x_\delta^- + \tau)$ is a non-constant affine function. We can see that this holds:

δ	τ	$\mu_{(3,3)}(x_\delta^- + \delta + \tau) - \mu_{(3,3)}(x_\delta^- + \tau)$
$(1,0,0)$	$(0,0,0)$	$x_2 + x_3$
$(1,0,0)$	$(1,0,0)$	$1 + x_2 + x_3$
$(1,0,0)$	$(2,0,0)$	$2 + x_2 + x_3$
$(0,1,0)$	$(0,0,0)$	$x_1 + 2x_3$
$(0,1,0)$	$(0,1,0)$	$x_1 + 2x_3$
$(0,1,0)$	$(0,2,0)$	$x_1 + 2x_3$
$(0,0,1)$	$(0,0,0)$	$x_1 + 2x_2$
$(0,0,1)$	$(0,0,1)$	$x_1 + 2x_2$
$(0,0,1)$	$(0,0,2)$	$x_1 + 2x_2$

■

This function can be generalized as follows.

Definition 5.3.3. *Let $n \geq 2k+1$. We define $\mu_{n,p} : GF(p^r)^n \to GF(p^r)$ to be the function*

$$\mu_{n,p} = x_1 + \sum_{i=1}^{\lfloor n/2 \rfloor} \left(b_p^- x_{[2i-1]_{(n)}} x_{[2i]_{(n)}} + b_p^+ x_{[2i]_{(n)}} x_{[2i+1]_{(n)}} \right) + \begin{cases} b_p^- x_n x_1 + b_p^+ x_1^2 & \text{if } n \text{ is odd,} \\ 0 & \text{otherwise,} \end{cases}$$

where $[i]_{(n)}$ denotes the integer j such that $1 \leq j \leq n$ and $j \equiv i \mod n$.

Lemma 5.3.2 ([122])**.** *Let f_1, f_2 be functions over $GF(p^r)^{n_1}$ and $GF(p^r)^{n_2}$, respectively. Let $f(x) = f_1(y) + f_2(z)$ where $x = (y, z)$, $y \in GF(p^r)^{n_1}$, and $z \in GF(p^t)^{n_2}$. Then:*

1. *f is balanced if f_1 or f_2 is balanced.*
2. *f satisfies $B(k)$ if both f_1 and f_2 satisfy $B(k)$.*

It has been shown that $\mu_{n,p}$ is balanced and satisfies the property $B(k)$ [50]. Let $\chi_{2k+1} = \mu_{2k+1,p}$. Using the lemma above, the following results hold.

Lemma 5.3.3 ([122])**.** *Let $\chi_{4k+2}(x_1, \ldots, x_{4k+2}) = \chi_{2k+1}(x_1, \ldots, x_{2k+1}) + \chi_{2k+1}(x_{2k+2}, \ldots, x_{4k+2})$. Then χ_{4k+2} satisfies $B(k)$ and is balanced.*

Theorem 5.3.4 (Cheating-Immune Scheme [122])**.** *Let k, m be positive integers with $m \geq k+1$ and let $n_1, \ldots, n_m \in \{4k+1, 4k+2\}$ such that $n = n_1 + \cdots + n_m$. Let $f(x) = \chi_{n_1}(x_1) + \cdots \chi_{n_m}(x_m)$, for $x = (x_1, \ldots, x_m)$ and for $i = 1, \ldots, m$, $x_i \in GF(p^t)^{n_i}$. If each χ_{n_i} is constructed according to Definition 5.3.3 or Lemma 5.3.3 and $\chi_{n_1}, \ldots, \chi_{n_m}$ have mutually disjoint variables, then the secret sharing scheme with defining function $f(x)$ is k-cheating-immune.*

This construction is evidently complex so we provide some intuition for how this conclusion was reached. From a cheating participant's perspective, determining the original secret using the defining function, their knowledge of their share, and the reconstructed secret, is equivalent to solving a system of equations. The property $B(k)$, accompanied by the balanced property, ensures that this system of equations has as many solutions as possible, giving the cheating participant no advantage in guessing the secret. Thus, constructing a cheating-immune scheme comes down to determining a defining function which satisfies said properties.

Cheating immunity differs from robustness and error decodability in that it is not attempting to prevent cheating so much as discourage cheating. Consequently, the focus is no longer on whether or not the reconstructed secret is not equal to the original secret. In fact, we expect that the reconstructed secret will be different from the original. We are not concerned with the scenario where the reconstructed secret is equal to the original secret, because both honest and cheating participants learn the same amount of information. The notion of cheating immunity shifts the focus to the remaining threat that exists if a different value is reconstructed: whether or not the cheating participants can still learn the original secret. Error decodability and robustness imply cheating immunity because if an adversary cannot modify the reconstructed secret by cheating, then there is no advantage to be gained from the reconstructed secret.

5.4 Cheater identification

The notions of robustness which we have considered thus far prevent malicious shareholders from corrupting the secret; however, they do not necessarily enable honest shareholders to establish which participants are "cheating". This concept is called *cheater identification* and was first suggested by McEliece and Sarwate [112]. We present a game-based definition similar to the game defined by Obana for cheater-identifiable threshold schemes [119].

The Cheater Identification Game. Assume a (k, n)-threshold scheme. The secret is s. Fix non-negative integers $1 \leq t < k$ and $1 \leq m \leq n$.

Step 1. t of the n total shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. A cheater identification algorithm is run on the t bad shares and $m - t$ good shares. The adversary may choose which good shares are used but cannot modify

the shares. The cheater identification algorithm outputs a set L consisting of the set of shares identified as cheaters.

The adversary wins the cheater identification game if none of the t bad shares are included in the set L.

If the adversary cannot win the cheater identification game, except with negligible probability, we say that the scheme is a t-cheater identifiable (k, n)-threshold scheme.

The first secret sharing schemes capable of identifying cheaters were presented independently by Rabin and Ben-Or [127] and Brickell and Stinson [37]. Later work by Carpentieri [41] improved upon the Rabin and Ben-Or construction by decreasing the size of the shares. It has been show that in order to achieve cheater identification, we must have $t < k/2$ [103]. The typical strategy for cheater identifiable schemes is to include additional information with each share distributed among the participants, which can be used to verify the validity of the participant's share. Brickell and Stinson [37] provide a construction of a cheater identifiable scheme which is based on Blakley's threshold scheme [19]. We review Blakley's scheme now.

Construction 5.4.1. [Blakley's Scheme [19]] Let $\mathcal{P}$ be the set of n participants. Let V be a k-dimensional vector space over $GF(q)$. Fix a publicly known line $\ell \in V$. Then the secret space, $\mathcal{S}$, is the set of q points on the line ℓ.

- **Share:** For a secret $s \in \mathcal{S}$, the dealer constructs a random $(k-1)$-dimensional subspace H that meets ℓ at a point. Let $H_s = H + s$. The dealer selects n distinct, random points, $\{s_1, \ldots, s_n\}$, on H_s such that no j of them lie in a $(j-2)$-dimensional Euclidean subspace or translate of a subspace (in other words, the no j of the points lie on a flat) for $j \leq k$. The dealer gives s_i to participant P_i for each $i = 1, \ldots, n$.
- **Recover:** Any k participants can use their points to uniquely determine H_s. Then $H_s \cap \ell = s$, so they can recover the secret.

■

To make Blakley's scheme cheater identifiable, the new construction has the dealer distribute additional information to the participants.

Construction 5.4.2. [Cheater Identifiable Scheme [37]] Let $\mathcal{P}$ be the set of n participants. Let V be a k-dimensional vector space over $GF(q)$. Fix a publicly known line $\ell \in V$. Then the secret space, $\mathcal{S}$, is the set of q points on the line ℓ.

- **Share:** For a secret $s \in \mathcal{S}$, the dealer follows the share algorithm from Blakley's scheme. Also, the dealer constructs n random $(k-1)$-dimensional subspaces, denoted H_i, such that the intersection of H with $j-1$ of the H_i's is a subspace of dimension $k-j$ for $j \leq k$. In addition to the point s_i, the dealer also gives to participant P_i the $n-1$ parallel hyperplanes $H_{ij} = H_i + s_j$, $j = 1, \ldots, n$, $j \neq i$.
- **Recover:** The recover algorithm is the same as in Blakley's scheme.
- **Cheater Identification:** A single honest participant, P_i, can check the validity of the share belonging to another participant, P_j. To do so, P_i simply verifies that the share s_j lies on H_{ij}.

■

The additional information given to participants allows them to rule out some possible secrets. In particular, $k-1$ participants can rule out up to $1 + \binom{n-k+1}{k-1}$ points as the secret; however, assuming $q >> n$, this causes no issues in practice. The remaining possible secrets are equally likely. Suppose that participant P_1 is honest and participant P_2 tries to cheat by suggesting that their share is $s_2' \neq s_2$. P_2 cannot choose s_2' to be a point on ℓ or a point on the hyperplane through s_1 parallel to ℓ, since it would be obvious that they were lying. Also, they will not choose a point on H_s, since that would not affect the secret. P_2 only succeeds in cheating if there is a hyperplane H_{12}' containing s_2 and s_2' equal to H_{12}. There are $q-1$ possibilities for H_{12}, all equally likely to contain s_2', thus the probability that P_2 succeeds in cheating is $1/(q-1)$. Otherwise, the honest participant P_1 can identify P_2 as a cheating participant. Suppose $n-1$ participants collude and cheat. They are most likely to be successful if they leave $k-2$ shares unchanged and lie about the remaining $n-k+1$ shares. Then the probability that the honest participant, P_1, cannot identify at least one cheater (that is, the cheating parties succeed) is at most $(n-k+1)/(q-1)$.

Example 5.4.1. Consider a $(2,3)$-threshold scheme over $GF(q)$ for some large prime q. Let ℓ be the x-axis containing the points $(b,0)$, $b \in GF(q)$, and let $s_1 = (1,5)$. Suppose P_1 also receives the lines $H_{12} = (1+a, -1-a)$ and $H_{13} = (4+a, 8-a)$, $a \in GF(q)$. Note that these are parallel lines. Then P_1 can rule out the points where H_{12} and H_{13} intersect ℓ, i.e. $(0,0)$ and $(12,0)$. Also, P_1 knows that the secret cannot be $H_{11} \cap \ell$, so they can rule out $(6,0)$. The key could be any of the remaining $q-3$ points on the x-axis. Suppose that P_2 receives the share $s_2 = (-2,2)$. Then P_1 and P_2 can determine that the secret must be $(-4,0)$. ■

The construction above enables participants to independently verify whether or not another participant's share is valid. Additionally, the cheater identification can occur

Construction	# honest shares required for cheater identification	Time of cheater identification
Brickell and Stinson [37]	1	Before reconstruction
Rabin and Ben-Or [127]	1	Before reconstruction
Carpentieri [41]	1	Before reconstruction
Kurosawa, Obana, Ogata [103]	k	Before reconstruction
Obana [119]	$k - \lfloor \frac{k-1}{3} \rfloor$	During reconstruction
Bellare, Dai, Rogaway [11]	k	During reconstruction

Table 5.1: Properties of cheater identifiable secret sharing schemes

separately from reconstruction. This differs from other cheater identifiable schemes which may require honest participants to combine their information to verify the validity of other shares or may perform cheater identification in conjunction with reconstruction. The general game-based definition of cheater identification that we have suggested encompasses each of these cases. In the case where a single participant may verify a single share corresponding to another participant, we can set $t = 1$ and $m = 2$. In the case where cheater identification occurs during reconstruction, we can require $m \geq k$. The flexibility of the definition makes it possible to apply to each of the cheater-identifiable schemes in Table 5.1 with varying properties.

Table 5.1 also includes a construction from Bellare, Dai, and Rogaway [11]. Later, in Section 5.6, we provide more details on the property of error correction provided by this construction. Here, it is relevant to note that this error correction property encompasses cheater identification. Essentially, the reconstruction algorithm outputs valid shares in addition to the recovered secret. This approach inherently provides cheater identification. Since this functionality is built into reconstruction, an authorized set of participants is required in order to perform reconstruction. In the case of a (k, n)-threshold scheme, this is k honest shares.

Cheater identification differs from the previous notions of robustness we have discussed since it includes an additional step to check for cheating participants. This notion is stronger than the definition of robustness since it prevents an invalid secret from being reconstructed and it identifies participants holding incorrect shares. Although the constructions of cheater identifiable schemes are typically less space efficient than standard robust schemes (due to the additional information that is distributed to participants), it enables honest participants to remove potentially malicious participants from the protocol. This can be useful in cases where one is concerned about the presence of cheating participants and has the ability to remove participants and restart a secret sharing protocol.

Cheater identification schemes generally do not overlap with error decodable schemes, since the goal is not to reconstruct in the presence of corrupted shares, rather it is to identify and remove the corrupted shares. Given this difference, cheater identifiable schemes can often tolerate more cheating participants than an error decodable scheme.

5.4.1 Identification vs detection

A notion similar to cheater identification is cheater detection. Where the concept of cheater identification is clearly defined as the requirement to be able to identify which participants submitted modified shares, cheater detection is a somewhat ambiguous term. Some papers use the term cheater detection to refer to a scenario where we want to detect that *some* share has been modified from its original state [79, 120]. This is essentially the same as the definition of robust secret sharing. If an adversary can "cheat" by submitting a modified share that results in a different secret being reconstructed, without being detected, then the scheme is not robust. Other papers which discuss cheater detection also enable honest participants to identify which shares belong to cheaters [37, 82]. The term "cheater detection" is used in both scenarios, so it is important to note which papers intend to satisfy robustness and which intend to satisfy cheater identification.

5.5 Fairness

The notion of *fairness* originated from multi-party computation schemes [71], where the property means that no player receives the output of a multi-party computation unless all players receive the output. This idea can be extended to secret sharing schemes, where the property suggests that no player participating in reconstruction receives the correct reconstructed secret unless all players do. This can be considered a notion of robustness since it requires that an adversary only receives the secret if the honest players also receive the correct secret. The following game-based definition is aligned with the definition given by Laih and Lee [104].

The Fairness Game. Assume a (k, n)-threshold scheme. The secret is s. Fix a non-negative integer $1 \leq t < k/2$.

Step 1. t of the n total shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. A secret s' is reconstructed using k shares, including the adversary's t shares. The $k-t$ "good shares" to be used in reconstruction may be chosen by the adversary.

If the probability that the adversary learns the correct secret, s, is greater than the probability that an honest participant learns the correct secret, then the adversary wins the fairness game.

If the adversary cannot win the fairness game (except with negligible probability), we say that the scheme is a t-fair (k, n)-threshold scheme.

Constructions of fairness schemes from Laih and Lee [104] and later Hwang and Chang [81] made use of cheater identifiable schemes, which we discussed in Section 5.4.

Construction 5.5.1. [Threshold Scheme with Fairness [81]] We assume the existence of a t-cheater identifiable threshold scheme where cheater identification occurs before reconstruction and only requires 1 honest share. For example, any of [37, 127, 41]. The following construction is based on Shamir's secret sharing scheme, so it makes sense to apply a cheater identification protocol like the one defined by Carpentieri [41].

We present a construction of a (k, n)-threshold scheme with t-fairness. Let $s \in \mathbb{Z}_p$ be the secret, for some large prime p.

Share:

1. The dealer randomly chooses an integer $a_0 \in \mathbb{Z}_p$ and computes $b_0 = a_0 \oplus s$.
2. The dealer generates a random polynomial $f(x) = a_0 + a_1 x + \cdots a_{k-1}x^{k-1}$ of degree $k-1$ and a random polynomial $g(x) = b_0 + b_1 x + \cdots + b_{t-k-1}x^{t-k-1}$ of degree $t-k-1$, both in $\mathbb{Z}_p[x]$.
3. The dealer gives participant P_i the pair $(f(i), g(i))$.

Recover: Denote each participant's share by (d_{i1}, d_{i2}), for $i = 1, \ldots, n$.

1. k participants combine the first half of their shares d_{i1}.
2. The honest participants run a cheater identification protocol on the initial shares to identify any invalid shares. If all shares are valid, continue. Otherwise, abort the reconstruction process.
3. The k participants provide the second half of their shares d_{i2}.
4. The honest participants run a cheater identification protocol on the second set of shares to identify any invalid shares. The participants identify $t-k$ valid shares to be used in the next step.

5. Using the k valid shares from Step 1, the participants recover $f(x)$ using Lagrange interpolation. Using the $t-k$ valid shares identified in Step 4, the participants recover $g(x)$ using Lagrange interpolation.
6. Participants compute the secret $s = f(0) \oplus g(0)$.

The paper [81] proves that this is a t-fair (k, n)-threshold scheme, i.e., that an adversary cannot win the fairness game. ■

It is not hard to see that Shamir's scheme does not ensure the *fairness* property. This is one of the points made by Tompa and Woll [146] when they introduced the concept of robustness: the idea that dishonest participants can learn the secret even if an invalid secret is recovered. The justification that Shamir's threshold scheme is not fair is the same as the justification that it is not cheating-immune, given in Section 5.3. So, we have the following result.

Theorem 5.5.1. *A (k, n)-Shamir threshold scheme is not fair.*

The concept of fairness is very similar to that of cheating immunity, in the sense that the goal is to construct schemes where cheating participants do not have advantages over honest participants. The differences are that cheating-immune schemes apply to (n, n)-threshold schemes with up to $n-1$ cheaters and fairness schemes apply to more general (k, n)-threshold schemes with only $t < k/2$ cheaters. Cheating-immune schemes require more shares in reconstruction (to account for the possibility of more cheating participants).

Additionally, we can observe that if a scheme is robust, then it must also be fair. Suppose we have a robust scheme in which an adversary can win the fairness game. This suggests that the adversary is able to modify t shares in such a way that the honest participants do not learn the correct secret. This implies that the reconstructed secret $s' \neq s$, contradicting the fact that the scheme is robust. Therefore, robustness implies fairness.

5.6 Error correction

Error correction was suggested as a property of the recently proposed *adept secret sharing* [11], alongside privacy and authenticity properties. In addition to guaranteeing that the recovery of the secret succeeds, it also identifies valid shares when there is a unique explanation as to which shares imply the secret. Essentially, it combines the concepts of

robustness and cheater identification. The original work presents the definitions in a computational setting, but it is fairly straightforward to consider the idea of error correction in an information theoretic setting.

In defining the error correction property, the authors formalize which information is included in the Recover algorithm. Rather than simply taking as input some set of shares, the algorithm also requires some "known" information. That is, the message-recovery algorithm is of the form: Recover: Known $\times$ Shares $\rightarrow$ Msg $\times$ Rand $\times$ Shares $\cup \{\perp\}$. The Known information includes either an access structure that the recovering party knows to be operative or it is the subset of the shares given to Recover that are known to be valid. Then, the algorithm outputs a message, the shares used to reconstruct the message, and the randomness. If the algorithm cannot recover the message, $\perp$ is output. The following definition is an adapted version of the game-based definition from the original paper [11], where we have restricted to the threshold case.

The Error Correction Game. Assume a (k, n)-threshold scheme. The secret is s. Fix some t, such that $1 \leq t \leq n$.

Step 1. The adversary creates/modifies t shares and chooses some corresponding Known information.

Step 2. The t shares and the Known information are input into the Recover algorithm, which outputs a triple $(\mathcal{M}, \mathcal{R}, \mathcal{V})$ or $\perp$, where $\mathcal{M}$ is a message, $\mathcal{R}$ the randomness, and $\mathcal{V}$ the set of shares used to recover $\mathcal{M}$.

Step 3. An Explanation algorithm is run on the Known information and the input shares. For all possible subsets of the input shares, the algorithm determines all triples $(\mathcal{M}, \mathcal{R}, \mathcal{V})$ that may be returned from running Recover on the shares and the corresponding access structure. If all message and randomness pairs, $(\mathcal{M}, \mathcal{R})$, are equal and there is a valid set of shares $\mathcal{V}$ containing all others, the algorithm returns this unique triple $(\mathcal{M}, \mathcal{R}, \mathcal{V})$. Otherwise, it outputs $\perp$ if no such valid set exists.

The adversary wins the game if the outputs of Explanation and Recover algorithms are different.

If the adversary can never win the error correction game, this is called *perfect error correction*. The goal of the adversary is to force the Recover algorithm to recover something wrong, i.e., something that does not align with the unique explanation.

The authors prove that error correction implies robustness [11]. Let us consider an example to demonstrate how this definition is stronger than robustness. The main difference between error correction and robustness is that in addition to preventing recovery when there is no authorized set of shares, error correction also prevents recovery if there is more than one possible explanation for which shares were corrupted. Additionally, the expanded capabilities of the Recover algorithm as defined in adept secret sharing allow for more possibilities. Take, for example, a $(2, 4)$-threshold scheme constructed according to Construction 5.1.1. We already know that this is robust. However, consider an adversary attacking this scheme under the error correction game. Suppose the adversary modifies two shares, $a \to \tilde{a}$ and $b \to \tilde{b}$, such that they correspond to a different, valid secret. Then, if the Recover algorithm was given all four shares, including $\tilde{a}$ and $\tilde{b}$, there would be no way to determine which secret is the correct value to recover. The Recover algorithm may return one of the two valid secrets, while the Explanation algorithm would return $\perp$. So, the construction does not provide error correction.

To ensure error correction, the following modification to the Recover algorithm is suggested by [11].

Construction 5.6.1. [Recover with Error Correction [11]]

- Given a set $\mathcal{S}$ of shares and some Known information (an access structure or some set of valid shares), determine all possible subsets of $\mathcal{S}$ that contains the valid shares or satisfy the access structure defined by Known. Denote these by $\mathcal{S}_1, \ldots, \mathcal{S}_w$.
- For each $\mathcal{S}_i$, for $i = 1, \ldots, w$, run the Recover algorithm on $\mathcal{S}_i$ and Known.
- If Recover always fails to find a message, return $\perp$. If Recover returns two different sets of triples $(\mathcal{M}, \mathcal{R}, \mathcal{V})$, $(\mathcal{M}', \mathcal{R}', \mathcal{V}')$ of a message, randomness, and valid set of shares, where $\mathcal{V}' \not\subset \mathcal{V}$ (or vice versa), then return $\perp$. In this case, there are two different plausible scenarios. Otherwise, return the unique message, randomness pair, and the corresponding largest set of valid shares, $(\mathcal{M}, \mathcal{R}, \mathcal{V})$.

■

This construction can be applied to any recover algorithm which returns not only the secret, but also the randomness and corresponding valid shares.

5.7 Non-malleability

Non-malleability in the context of encryption refers to the property that given a ciphertext, it is impossible to generate a different ciphertext such that the respective plaintexts are related [57]. This concept has been extended to other cryptographic protocols, such as commitment schemes and zero-knowledge proofs. While the literature has agreed upon a definition of non-malleability in the context of encryption protocols, the same cannot be said for non-malleability of secret sharing schemes.

5.7.1 Existing definitions of non-malleable secret sharing

The first mention of non-malleable secret sharing was in a 2006 PhD thesis by Kenthapadi [97] and in a corresponding paper on distributed noise generation [59]. The authors refer to *non-malleable verifiable secret sharing* as an extension of verifiable secret sharing (VSS). Although they do not provide a formal definition or construction of non-malleable VSS, the papers provide a brief, high-level definition. The authors state that:

> "A non-malleable VSS scheme ensures that the values shared by a non-faulty processor are completely independent of the values shared by the other processors; even exact copying is prevented." [97, 59]

Here, *values* refers to shares corresponding to some secret. When the authors write that "exact copying is prevented," they intend to prevent two shares from being equal to one another. This is to address a scenario in which a malicious party views a valid share and replaces a share intended for another participant with the valid share that they had previously seen. If the shares were independent in the usual sense of the word, then it should be possible that two shares be equal to one another. Thus, this definition is somewhat ambiguous. Furthermore, constructions of verifiable secret sharing, discussed in Section 4.1, already ensure that a valid share intended for one participant cannot be sent to another and still be verified. This is because the *Verify* algorithm typically takes as input the identity of the shareholder. It is unclear whether this requirement that copying be prevented provides any additional security above that which is already provided in VSS schemes. One difference between this definition of non-malleable VSS and some VSS schemes is that the definition of non-malleable VSS does not assume private channels between participants. This property is not inherent to all verifiable schemes, as some models of verifiable secret sharing also assume that there do not exist private channels between participants [45].

The suggested construction in [97] to achieve non-malleable VSS from a VSS scheme appends public keys to shares and then encrypts the share and public key using a non-malleable encryption scheme [57]. For example, suppose participant P_i is sending a share to participant P_j. Then, P_i appends to the beginning of their share the public keys corresponding to P_i and P_j, and encrypts this message using P_j's public key.

The work by Kenthapadi, in particular the followup paper on distributed noise generation [59], is well-known. However, this is not the only line of work that discusses non-malleability in the context of secret sharing. In 2008, a paper in the area of secure multi-party computation [83] provided a new definition of non-malleable secret sharing. They consider the setting of a $(2, 2)$-threshold scheme and say that the scheme is *non-malleable* if an adversary cannot win the following game.

The IPS Malleability Game.

Step 1. In a $(2, 2)$-threshold scheme, two shares, a and b, are generated for some secret s.

Step 2. The adversary modifies a single share $a \to \tilde{a}$.

Step 3. The reconstruction algorithm takes as input $\tilde{a}$ and b. The adversary wins if $\tilde{a} \neq a$ and the reconstruction algorithm outputs some valid secret s'.

If the adversary can win the above game with at most probability ϵ, they say that the scheme is ϵ-non-malleable. Later work studying fairness in secure computation used the same definition of non-malleable secret sharing [72, 71, 17], as did a paper on universal composability [130]. It has been noted that this definition of non-malleability can be achieved using AMD codes in the same way that they had previously been used to provide robustness [71, 49].

Definition 5.7.1 ((Strong) AMD code [49])**.** *Let $\mathcal{G}$ be an additive abelian group of size n and $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$ be m pairwise disjoint k-subsets of $\mathcal{G}$. Then $(\mathcal{G}, \mathcal{A})$ is an (n, m, k)-*AMD code *if an adversary cannot win the following game with non-negligible probability.*

1. *A* source $i \in \{1, \ldots, m\}$ *is chosen and given to the adversary.*
2. *The adversary chooses a value $\Delta \in \mathcal{G} \setminus \{0\}$.*
3. *The source is encoded by choosing g uniformly at random from $\mathcal{A}_i$.*
4. *The adversary wins if and only if $g + \Delta \in A_j$ for some $j \neq i$.*

By secret sharing the encoding of a secret instead of the secret itself, we can achieve robustness [49] and IPS non-malleability [83, 71]. This definition of non-malleability is very similar to the definition we provided for robust secret sharing. Aside from the fact that it is restricted to the case of $(2, 2)$-threshold schemes (although this is easily generalized), the only remaining difference is that the IPS Malleability game checks if the inputted share $\tilde{a} \neq a$ rather than checking whether the reconstructed secret $s' \neq s$. This is a slightly stronger definition, given that if the reconstructed secret $s' \neq s$, this implies that the inputted shares must differ. On the other hand, different shares do not necessarily imply a different reconstructed secret. The motivation for this second definition of non-malleable secret sharing is unclear. The original paper discusses how such a scheme prevents cheating participants from modifying their shares without being detected, but it is unclear if there is an advantage to be had by an adversary who modifies their shares in a way that does not change the reconstructed secret.

The above two definitions of non-malleable secret sharing do not agree with one another. To make matters more complicated, there is a third line of work which refers to non-malleable secret sharing that provides another definition, different from the first two. This work began in 2010 with the introduction of non-malleable codes [63, 64]. In coding theory, non-malleable codes can be considered as a relaxation of error-correcting and error-detecting codes. The previously discussed AMD codes are an example of error-detecting codes. At a high level, non-malleable codes are codes that when decoded, either return the original message or a message *independent of and unrelated to* the original message. To precisely define non-malleability, it must be done with respect to some family of *tampering functions*, which determine what it means for messages to be independent or unrelated.

Follow-up work noted that non-malleable codes in the *split-state model* could be considered a secret sharing scheme [61, 5, 4]. The split-state model [62, 52] describes a setting where a codeword consists of two parts which are stored on two separate memory parts that can be tampered with independently. Any such code is equivalent to a $(2, 2)$-threshold scheme where the encoding mechanism is the sharing function, the decoding mechanism is the reconstruction function, and the two parts of the codeword are the shares. These can be called 2-out-of-2 non-malleable secret sharing schemes. A main difference between these non-malleable schemes and previous notions of robustness is the concept of *independent tampering*. For example, in a 2-out-of-2 scheme, it is equivalent to assuming that two different adversaries each receive a share that they can modify, but they cannot communicate with one another.

Dziembowski, Kazana, and Obremski [61] constructed a non-malleable code for single bit messages in the split-state model (equivalent to a non-malleable $(2, 2)$-threshold scheme for single bit messages). In their construction, both shares lie in a sufficiently large n-

dimensional vector space $\mathbb{F}^n$. To encode (or secret share) 0, we choose a random pair of orthogonal vectors L and R, i.e., $\langle L, R\rangle = 0$. To encode (or secret share) 1, we choose a random pair of non-orthogonal vectors L and R, i.e., $\langle L, R\rangle \neq 0$. To achieve non-malleability in this setting, the adversary must not be able to independently tamper with the shares L and R to produce a different output message than the original with probability greater than 1/2. The proof of non-malleability is quite involved and so we refer readers to the original paper for more details [61].

Later, Aggarwal, Dodis, and Lovett presented the first non-malleable code in the split-state model for multi-bit messages [5]. To do so, they construct a non-malleable code with respect to the family of affine functions in a finite field of prime order p, $\mathbb{F}_p$. Then, the scheme consists of first encoding the secret using this non-malleable code, and then splitting the encoded secret into two shares, L and R, such that the inner product of L and R is equal to the encoding.

Extending this line of work, Goyal and Kumar initiated a more formal study of this notion of non-malleable secret sharing [74]. In this work, they presented (t, n)-threshold schemes satisfying this notion of non-malleability based on non-malleable codes. They provide formal definitions for non-malleable secret sharing that are generalized versions of non-malleable codes in the split-state model. Most, but not all, of the recent work which refers to non-malleable secret sharing is based on these definitions. The initial study focused on threshold schemes while later work provided constructions for general access structures [75, 7, 3]. These constructions worked in an information-theoretic model and aimed to satisfy statistical privacy and statistical non-malleability. This definition of non-malleable secret sharing has also been extended to the computational setting [67, 35] and it has been combined with the idea of leakage resilience [67, 62].

Using Goyal and Kumar's definition of non-malleability, it is easy to see that a secret sharing scheme which is non-malleable cannot be linear. In a linear secret sharing scheme, briefly discussed in Section 3.2.2, local computations on the shares have meaningful effects on the reconstructed secret. This directly contradicts the intentions of non-malleable secret sharing schemes.

5.7.2 Redefining non-malleable secret sharing

We propose a novel game-based definition of non-malleability. The goal of this definition is to encompass the idea of non-malleability, as it is typically used in cryptographic literature [57], in the most straightforward manner. This definition is motivated by the definitions presented by Goyal and Kumar [74], but rather than deriving definitions from

coding theory, we present them from a more cryptographic perspective. Goyal and Kumar discuss non-malleability with respect to "families of tampering functions." In contrast, we define non-malleability with respect to a symmetric relation $\sim$. With this notation, we can provide more fine-grained constructions of non-malleable schemes, secure with respect to specific relations. The schemes we discuss satisfy perfect privacy and non-malleability as opposed to statistical privacy and statistical non-malleability.

The $\sim$-Malleability Game. Assume a (k, n)-threshold scheme. The secret is s. Fix some t, $1 \leq t < k$, and a symmetric relation $\sim$ over the set of valid secrets.

Step 1. t of the n total shares are given to the adversary.

Step 2. The adversary modifies the t shares to create new "bad shares".

Step 3. A secret s' is reconstructed using k shares, t of which are the "bad shares". The good shares used during reconstruction may be chosen (but not modified) by the adversary.

The adversary wins the $\sim$-malleability game if the reconstructed secret s' is a valid secret such that $s' \neq s$ and $s' \sim s$.

If the adversary cannot win the non-malleability game, except with negligible probability, we say that the scheme is t-non-malleable with respect to $\sim$.

We note that if we let $\sim$ denote any relationship between two different values in the set of secrets, i.e., we let $\sim$ be $\neq$, then the only requirement for the adversary to win is that $s' \neq s$. Then, this definition is equivalent to the definition of robust secret sharing. It follows from this observation that if a (k, n)-threshold scheme is robust, it is also non-malleable.

Consider, for example, a Shamir threshold scheme. We showed in Section 5.1 that a (k, n)-Shamir threshold scheme is not robust. Although non-malleability is a weaker requirement, we can demonstrate that a (k, n)-Shamir threshold scheme is also not non-malleable, i.e., it is malleable, for a given relation $\sim$.

Theorem 5.7.1. *A (k, n)-Shamir threshold scheme is malleable with respect to the relation $\sim$ where $a \sim b$ if $a = b + c$ for some pre-specified c.*

Proof. We prove this by describing how the adversary can carry out a successful attack by modifying only a single share. Suppose the shares $v_1, \ldots, v_k$ are used to reconstruct a polynomial.

The true secret, s, can be written as $s = \sum_{i=1}^{k} a_i v_i$ where a_i denotes the Lagrange coefficients and v_i are the good shares for $i = 1, \ldots, k$.

Assume that the adversary modifies a single share v_1 by adding some δ. That is, upon reconstruction, the adversary submits $v_1' = v_1 + \delta$.

The reconstructed secret will be

$$s' = a_1 v_1' + \sum_{i=2}^{k} a_i v_i = s + a_1 \delta.$$

Since the Lagrange coefficient a_1 is publicly known, the adversary can choose δ such that $a_1 \delta = c$ for a given c. Then, the reconstructed secret will satisfy $s' = s + c$, i.e., $s' \sim s$, and the adversary wins the game. □

The above malleability game considers the case where an adversary can modify multiple shares. Previous work distinguished this from the case where an adversary can *independently* tamper with up to n shares. That is, they cannot use information from one share to inform how they tamper with another share. Note that in the original game, $t < k$, whereas in the following game where the adversary is restricted to independent tampering, we can have $t \leq n$.

The $\sim$-Malleability Game with Independent Tampering. Assume a (k, n)-threshold scheme. The secret is s. Fix $1 \leq t \leq n$ and a symmetric relation $\sim$ over the set of valid secrets.

Step 1. t of the n total shares are given to t non-communicating adversaries, i.e., each receives one share.

Step 2. t non-communicating adversaries each receive a share and modify their share to create new "bad shares".

Step 3. A secret s' is reconstructed using k shares. If $k \geq t$ then choose t of the "bad shares" to reconstruct. If $k < t$ then reconstruct using the t "bad shares" and $k - t$ "good shares". The good shares used during reconstruction may be chosen (but not modified) by an adversary.

The adversary wins the $\sim$-malleability game if the reconstructed secret s' is a valid secret such that $s' \neq s$ and $s' \sim s$.

The construction of a non-malleable code in the split-state model which we discussed in the previous subsection, from Aggarwal, Dodis, and Lovett [5], fulfills this definition of non-malleability with independent tampering for $(2, 2)$-threshold schemes. Here, the scheme is non-malleable with independent tampering with respect to the relation $\sim$, where $\sim$ includes any affine relationship in the finite field. More general constructions of t-out-of$n$ non-malleable secret sharing schemes with independent tampering are given in the work by Goyal and Kumar [74]. These constructions are somewhat complex and rely on several building blocks, such as leakage resilient schemes (discussed in Section 4.4.2), so we do not include them here.

5.7.3 Comparison with related notions

Non-malleability can be considered a notion of robustness because it concerns itself with the presence of cheating participants. It is a weaker notion than the concept of robustness considered by Krawcyzk [100] and Tompa and Woll [146]. Robust secret sharing schemes are able to detect that an error has occurred and potentially correct the error. In a non-malleable scheme, an error may or may not be detected and/or corrected. Instead, it only ensures that if an error occurs that is not able to be corrected, then the output is independent of the true secret. The idea is also similar to cheating immunity, which is the topic of Section 5.3. Suppose we choose $\sim$ in such a way that it contains all relationships which reveal any additional information about the true secret. Then, a scheme which is non-malleable with respect to $\sim$ would also be cheating-immune, since the adversaries gain no advantage from their knowledge of the relationship $\sim$.

Non-malleability differs from the previous notions we have considered in that it considers independent tampering: a setting in which an adversary can independently modify up to n shares but cannot combine the information gathered from each share. This threat is unique to the literature on non-malleable schemes. A similar, but different setting is considered in leakage-resilient schemes (Section 4.4.2), where the adversary may obtain information from all n shares. This setting is different because, although the adversary obtains some leaked information, they do not have the ability to modify the shares.

Chapter 6

Conclusion and Discussion

In addition to surveying work which has extended models of secret sharing, here is a summary of the most important conclusions and new contributions of the previous chapters:

- (Chapter 3: Reproducibility) Section 3.3 includes a comprehensive presentation of the approaches to reproducibility/repairability of shares which may be achieved with or without a dealer. We compare each approach and observe that repairability on the part of the shareholders removes the potential attack vector of a dealer holding all of the information.

- (Chapter 3: Relaxations) Most alternative models of secret sharing were initially considered in the information-theoretic setting. Table 6.1 summarizes some variations of secret sharing schemes that have been covered in this survey and examples of where they have been studied in the computational model and/or the information-theoretic model. Constructions that provide statistical privacy are listed under the information-theoretic model because they still consider an adversary with unbounded computational power.

- (Chapters 3 and 4: Trade-offs) In Section 3.2.2, Section 3.4.3, and Section 4.1.1 we noticed a similar trade-off between non-interactivity and unconditional security. It appears that in several scenarios, relaxing the privacy requirement to the computational setting enables us to construct protocols for additional functionalities that are *non-interactive*. This is an interesting relationship to explore in other areas of work.

- (Chapter 4: Verifiability) There exist several different models of verifiability that have been used in previous work. We present a comprehensive comparison of each

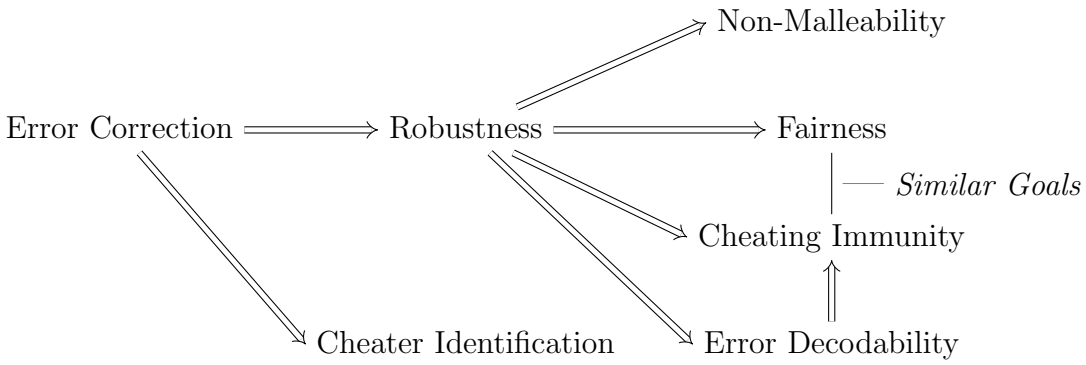

Figure 6.1: Summary of relationships between notions of robustness

notion of verifiability and the differences in the models used. This includes more formal definitions of some models than have been given in previous works.

- (Chapter 4: Deniability) There is an inherent deniability property afforded by classical secret sharing schemes which allows shareholders to deny having shares corresponding to a particular secret. We have formalized this property and demonstrated how it applies to classical threshold schemes.
- (Chapter 5) There exist many notions of robustness which only differ slightly in their goals and/or definitions. Chapter 5 presents comparable game-based definitions of each of these notions. Figure 6.1 summarizes the relationship between each notion.
- (Chapter 5: Non-Malleability) Non-malleable secret sharing has been defined in three different lines of work and the definitions in the literature do not agree with one another. We present a critique of some of the definitions in previous work and a new game-based definition motivated by one of the three lines of work.

Comparing notions of robustness. To elaborate on Figure 6.1, we observe that the notion of error correction [11] is the strongest of the notions of robustness we have considered. It encompasses both robustness and cheater identification and formalizes the intuition that the reconstruction process does the best job possible with the information that is presented to it. Error correction is also achievable for general access structures whereas robustness is difficult to achieve without assuming an honest majority of participants.

Variation	Section	Information-Theoretic	Computational
Adept SS	Sections 4.1 and 5.6		✓[11]
Anonymous SS	Section 3.5	✓[140, 28]	
Cheating-immune SS	Section 5.3	✓[123, 124]	
Error-decodable SS	Section 5.2	✓[102, 109]	
Evolving SS	Section 3.3.3	✓[98, 99]	
Fully dynamic SS	Section 3.2.1	✓[21]	✓[148]
Function SS	Section 3.1.3		✓[30]
Leakage-resilient SS	Section 4.4.2	✓[74, 16]	
Multi-secret sharing	Section 3.1.1	✓[22, 88]	✓[54]
Non-malleable SS	Section 5.7	✓[74]	✓[67]
Password-protected SS	Section 3.4.4		✓[8, 90, 38]
Proactive SS	Section 4.4.1	✓[141, 115]	✓[80]
Probabilistic SS	Section 3.6.1	✓[51]	
Rational SS	Section 4.2.1	✓[77]	✓[73]
Repairable SS	Section 3.3.2	✓[142, 105]	
Robust SS	Section 5.1	✓[44, 42]	✓[100, 129]
Social SS	Section 4.2.2	✓[118]	
Verifiable SS	Section 4.1	✓[127, 141]	✓[14, 68, 121]
Visual SS	Section 3.1.2	✓[20, 113]	

Table 6.1: Variations of secret sharing schemes and examples of relevant studies in an information-theoretic and/or computational setting.

We also observed that robustness implies error decodability, non-malleability, cheating immunity, and fairness. Given these relationships and the fact that there exist efficient constructions of robust secret sharing schemes, the benefits of robust secret sharing schemes become even more clear. On the other hand, the benefits of constructions specific to error decodability, non-malleability, cheating immunity, and fairness are less obvious since the goals of these notions are already accomplished by robust schemes.

Comparing notions of verifiability. In contrast to the notions of robustness, which are largely interchangeable with one another, the notions of verifiability offer significant differences from one another. Each definition of verifiability which we discussed in Section 4.1 protected against malicious dealers; however, aside from this similarity, they achieved different goals and required different constructions. The exception to this is the property of

authenticity, which is very closely related to the original definition of non-interactive verifiability. The non-interactive verifiability property implies authenticity. Also, authenticity is slightly more restrictive in the sense that it requires that each share commit to a unique secret. Although this allows for some efficient constructions, it also excludes any constructions in an information-theoretic setting. The paper proposing authenticity [11] claims that the authenticity definition is more widely applicable since it does not require the use of broadcast channels; however, the original definition of non-interactive verifiability did not in and of itself *require* the use of broadcast channels (despite the fact that some early constructions made use of them).

On overlapping terms. Another observation of note is the overuse of terms defining models of secret sharing. For example, aside from the collection of notions of robustness, there are two different definitions of robustness itself (one slightly stronger than the other). We have also reviewed four different concepts of verifiability and observed that non-malleability was defined in three different, contradictory ways. This demonstrates the importance of thorough literature reviews and accessible writing. We have laid the groundwork to make it possible to notice the differences between each of the notions we have covered. That being said, the original publications of these notions are more often than not defined in very different ways. This makes it difficult to know for certain whether one is defining a redundant notion. Ideally, future work in this area will consider the breadth of available models of secret sharing before introducing supposedly novel definitions.

Combining capabilities and models. One advantage of adept secret sharing [11] is its ability to easily combine several different properties (error correction, authenticity, and reproducibility). This has also been achieved in multi-party computation protocols which are secure against malicious participants. These protocols typically combine the ability to update secrets with verifiability or robustness. Additionally, as we pointed out in Section 3.6.2, many models of secret sharing have been considered in a computational setting which typically allows for more efficient schemes or smaller share sizes. Taking a combination of properties can be trivial in some cases and more complex in others. For example, the deniability property inherent to some classical schemes is contradictory to many notions of robustness and verifiability. It is interesting to consider which combinations of extended capabilities and alternative adversarial models have not been previously considered in the literature and which combinations may be useful in practice.

On definitions. With this thesis, we have aimed to define models of secret sharing in such a way that they are more readable and easy to understand. We have stripped away some superfluous notation of previous work to highlight the main goals and differences of each model. Ideally, these more accessible definitions, presented in a unified way, will prevent the consideration of overlapping, contradictory, and/or unnecessarily complex models of secret sharing in future work. We use consistent terminology throughout this thesis and present most of the variations of secret sharing schemes using similar game-based definitions, for consistency. We hope that this thesis highlights the advantages of existing models and helps bring context to some of the more recent contributions in the field.

References

[1] Michel Abdalla, Mario Cornejo, Anca Nitulescu, and David Pointcheval. Robust password-protected secret sharing. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 61–79. Springer, Heidelberg, September 2016.

[2] Leonard Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science*, pages 55–60. IEEE Computer Society, 1979.

[3] Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 510–539. Springer, Heidelberg, August 2019.

[4] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 459–468. ACM Press, June 2015.

[5] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th ACM STOC*, pages 774–783. ACM Press, May / June 2014.

[6] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 805–817. ACM Press, October 2016.

[7] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 593–622. Springer, Heidelberg, May 2019.

[8] Ali Bagherzandi, Stanislaw Jarecki, Nitesh Saxena, and Yanbin Lu. Password-protected secret sharing. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 433–444. ACM Press, October 2011.

[9] Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 499–529. Springer, Heidelberg, November 2020.

[10] Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 600–619. Springer, Heidelberg, February 2005.

[11] Mihir Bellare, Wei Dai, and Phillip Rogaway. Reimagining secret sharing: Creating a safer and more versatile primitive by adding authenticity, correcting errors, and reducing randomness requirements. *PoPETs*, 2020(4):461–490, October 2020.

[12] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[13] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[14] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret sharing. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 251–260. Springer, Heidelberg, August 1987.

[15] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, Heidelberg, August 1990.

[16] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 531–561. Springer, Heidelberg, August 2018.

[17] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 421–439. Springer, Heidelberg, August 2014.

[18] Bob Blakley, G. R. Blakley, Agnes Hui Chan, and James L. Massey. Threshold schemes with disenrollment. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 540–548. Springer, Heidelberg, August 1993.

[19] George Robert Blakley. Safeguarding cryptographic keys. In *International Workshop on Managing Requirements Knowledge (MARK) 1979*, pages 313–318. IEEE, 1979.

[20] Carlo Blundo, Annalisa De Bonis, and Alfredo De Santis. Improved schemes for visual cryptography. *Designs, Codes, and Cryptography*, 24(3):255–278, 2001.

[21] Carlo Blundo, Antonella Cresti, Alfredo De Santis, and Ugo Vaccaro. Fully dynamic secret sharing schemes. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 110–125. Springer, Heidelberg, August 1994.

[22] Carlo Blundo, Alfredo De Santis, Giovanni Di Crescenzo, Antonio Giorgio Gaggia, and Ugo Vaccaro. Multi-secret sharing schemes. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 150–163. Springer, Heidelberg, August 1994.

[23] Carlo Blundo, Alfredo De Santis, Douglas R. Stinson, and Ugo Vaccaro. Graph decompositions and secret sharing schemes. *Journal of Cryptology*, 8(1):39–64, December 1995.

[24] Carlo Blundo, Alfredo De Santis, and Ugo Vaccaro. Randomness in distribution protocols. In Serge Abiteboul and Eli Shamir, editors, *Automata, Languages and Programming*, pages 568–579, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[25] Carlo Blundo, Antonio Giorgio Gaggia, and Douglas R. Stinson. On the dealer's randomness required in secret sharing schemes. *Designs, Codes, and Cryptography*, 11:235–259, 1997.

[26] Carlo Blundo, Alfredo De Santis, Antonio Giorgio Gaggia, and Ugo Vaccaro. New bounds on the information rate of secret sharing schemes. *IEEE Transactions on Information Theory*, 41(2):549–554, 1995.

[27] Carlo Blundo, Alfredo De Santis, Roberto De Simone, and Ugo Vaccaro. Tight bounds on the information rate of secret sharing schemes. *Designs, Codes, and Cryptography*, 11:107–110, 1997.

[28] Carlo Blundo and Douglas R. Stinson. Anonymous secret sharing schemes. *Discrete Applied Mathematics*, 77(1):13–28, 1997.

[29] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use PGP. In *Workshop on Privacy in the Electronic Society (WPES)*, pages 77–84. ACM, 2004.

[30] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, Heidelberg, April 2015.

[31] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

[32] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[33] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193. Springer, Heidelberg, April / May 2017.

[34] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.

[35] Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, and Daniele Venturi. Non-malleable secret sharing against bounded joint-tampering attacks in the plain model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 127–155. Springer, Heidelberg, August 2020.

[36] Ernest F. Brickell. Some ideal secret sharing schemes. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 468–475. Springer, Heidelberg, April 1990.

[37] Ernest F. Brickell and Douglas R. Stinson. The detection of cheaters in threshold schemes (rump session). In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 564–577. Springer, Heidelberg, August 1990.

[38] Jan Camenisch, Anja Lehmann, Anna Lysyanskaya, and Gregory Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 256–275. Springer, Heidelberg, August 2014.

[39] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Heidelberg, August 1997.

[40] Renato M. Capocelli, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the size of shares for secret sharing schemes. *Journal of Cryptology*, 6(3):157–167, March 1993.

[41] Marco Carpentieri. A perfect threshold secret sharing scheme to identify cheaters. *Designs, Codes, and Cryptography*, 5:185–187, 1995.

[42] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 195–208. Springer, Heidelberg, April 2012.

[43] Chris Charnes, Josef Pieprzyk, and Reihaneh Safavi-Naini. Conditionally secure secret sharing schemes with disenrollment capability. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, and Ravi S. Sandhu, editors, *ACM CCS 94*, pages 89–95. ACM Press, November 1994.

[44] Mahdi Cheraghchi. Nearly optimal robust secret sharing. *Designs, Codes, and Cryptography*, 87:1777–1796, 2019.

[45] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th FOCS*, pages 383–395. IEEE Computer Society Press, October 1985.

[46] Ashish Choudhury. Asynchronous error-decodable secret-sharing and its application. In S. Jajoda and C. Mazumdar, editors, *Information Systems Security (ICISS)*. Springer, 2015.

[47] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE Computer Society Press, May 2015.

[48] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 316–334. Springer, Heidelberg, May 2000.

[49] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 471–488. Springer, Heidelberg, April 2008.

[50] Paolo D'Arco, Wataru Kishimoto, and Douglas R. Stinson. Properties and constraints of cheating-immune secret sharing schemes. *Discrete Applied Mathematics*, 154(2):219–233, 2006.

[51] Paolo D'Arco, Roberto De Prisco, Alfredo De Santis, Angel Pérez del Pozo, and Ugo Vaccaro. Probabilistic Secret Sharing. In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117, pages 1–16, Dagstuhl, Germany, 2018.

[52] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 121–137. Springer, Heidelberg, September 2010.

[53] Ed Dawson and Diane Donovan. The breadth of Shamir's secret-sharing scheme. *Computers & Security*, 13(1):69–78, 1994.

[54] Massoud Hadian Dehkordi and Samaneh Mashhadi. New efficient and practical verifiable multi-secret sharing schemes. *Information Sciences*, 178(9):2262–2274, 2008.

[55] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and its applications. Technical report, George Mason University, 1997.

[56] Yvo Desmedt, Yongge Wang, and Mike Burmester. A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 277–287. Springer, 2005.

[57] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[58] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, 1993.

[59] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 486–503. Springer, Heidelberg, May / June 2006.

[60] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, May 1998.

[61] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 239–257. Springer, Heidelberg, August 2013.

[62] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th FOCS*, pages 293–302. IEEE Computer Society Press, October 2008.

[63] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science (ICS)*, 2010.

[64] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *Journal of the ACM*, 65(4):1–32, 2018.

[65] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[66] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 1775–1792. USENIX Association, August 2021.

[67] Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 448–479. Springer, Heidelberg, August 2019.

[68] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987.

[69] Motahhareh Gharahi and Massoud Hadian Dehkordi. Perfect secret sharing schemes for graph access structures on six participants. *Journal of Mathematical Cryptology*, 7:143–146, 2013.

[70] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, Heidelberg, May 2014.

[71] S. Dov Gordon. *On Fairness in Secure Computation*. PhD thesis, University of Maryland, College Park, 2010.

[72] S. Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 91–108. Springer, Heidelberg, February 2010.

[73] S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 229–241. Springer, Heidelberg, September 2006.

[74] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 685–698. ACM Press, June 2018.

[75] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 501–530. Springer, Heidelberg, August 2018.

[76] Xuan Guang, Jiyong Lu, and Fang-Wei Fu. Repairable threshold secret sharing schemes. arXiv preprint arXiv: 1410.7190, 2014.

[77] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. In László Babai, editor, *36th ACM STOC*, pages 623–632. ACM Press, June 2004.

[78] Lein Harn and Ching-Fang Hsu. Dynamic threshold secret reconstruction and its application to the threshold cryptography. In *Information Processing Letters*, volume 115, pages 851–857, 2015.

[79] Lein Harn and Changlu Lin. Detection and identification of cheaters in (t, n) secret sharing scheme. *Designs, Codes, and Cryptography*, 52:15–24, 2009.

[80] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 339–352. Springer, Heidelberg, August 1995.

[81] Ren-Junn Hwang and Chin-Chen Chang. Enhancing the efficiency of a (v, r, n)-fairness secret sharing scheme. In *18th International Conference on Advanced Information Networking and Applications (AINA)*, pages 208–211, 2004.

[82] Shin-Jia Hwang and Chin-Chen Chang. A dynamic secret sharing scheme with cheater detection. In Josef Pieprzyk and Jennifer Seberry, editors, *ACISP 96*, volume 1172 of *LNCS*, pages 48–55. Springer, Heidelberg, June 1996.

[83] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[84] M. Ito, A. Saio, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *Journal of Cryptology*, 6(1):15–20, March 1993.

[85] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[86] Wen-Ai Jackson and Keith M Martin. Perfect secret sharing schemes on five participants. *Designs, Codes, and Cryptography*, 9(3):267–286, 1996.

[87] Wen-Ai Jackson, Keith M. Martin, and Christine M. O'Keefe. Multisecret threshold schemes. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 126–135. Springer, Heidelberg, August 1994.

[88] Wen-Ai Jackson, Keith M. Martin, and Christine M. O'Keefe. On sharing many secrets (extended abstract). In Josef Pieprzyk and Reihaneh Safavi-Naini, editors, *ASIACRYPT'94*, volume 917 of *LNCS*, pages 42–54. Springer, Heidelberg, November / December 1995.

[89] Amir Jafari and Shahram Khazaei. Partial secret sharing schemes. Cryptology ePrint Archive, Report 2020/448, 2020. https://eprint.iacr.org/2020/448.

[90] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253. Springer, Heidelberg, December 2014.

[91] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 276–291, 2016.

[92] Dieter Jungnickel and Alexander Pott. Difference sets: An introduction. In *Difference Sets, Sequences and their Correlation Properties*, pages 259–295. Springer, 1999.

[93] Tarik Kaced. Almost-perfect secret sharing. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 1603–1607. IEEE, 2011.

[94] Tarik Kaced. *Secret Sharing and Algorithmic Information Theory.* PhD thesis, Universite de Montpellier, 2012.

[95] Bailey Kacsmar. Designing efficient algorithms for combinatorial repairable threshold schemes. Master's thesis, University of Waterloo, 2018.

[96] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the 8th Annual Structure in Complexity Theory Conference*, pages 102–111. IEEE, 1993.

[97] Krishnaram Kenthapadi. *Models and Algorithms for Data Privacy.* PhD thesis, Stanford University, September 2006.

[98] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 485–514. Springer, Heidelberg, October / November 2016.

[99] Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: Dynamic thresholds and robustness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 379–393. Springer, Heidelberg, November 2017.

[100] Hugo Krawczyk. Secret sharing made short. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 136–146. Springer, Heidelberg, August 1994.

[101] Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing. Cryptology ePrint Archive, Report 2018/1138, 2018. https://eprint.iacr.org/2018/1138.

[102] Kaoru Kurosawa. General error decodable secret sharing scheme and its application. *IEEE Transactions on Information Theory*, 57(9):6304–6309, 2011.

[103] Kaoru Kurosawa, Satoshi Obana, and Wakaha Ogata. t-Cheater identifiable (k, n) threshold secret sharing schemes. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 410–423. Springer, Heidelberg, August 1995.

[104] C. S. Laih and Y. C. Lee. A v-fairness (t, n) secret sharing scheme. In *IEEE Computers and Digital Techniques*, pages 245–248, 1997.

[105] Thalia M. Laing and Douglas R. Stinson. A survey and refinement of repairable threshold schemes. *Journal of Mathematical Cryptology*, 12(1):57–81, 2018.

[106] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol. https://signal.org/docs/specifications/x3dh/, 2016.

[107] Keith M. Martin. Dynamic access policies for unconditionally secure secret sharing schemes. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, pages 61–66, 2005.

[108] Keith M Martin. Challenging the adversary model in secret sharing schemes. *Coding and Cryptography II, Proceedings of the Royal Flemish Academy of Belgium for Science and the Arts*, pages 45–63, 2008.

[109] Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Error decodable secret sharing and one-round perfectly secure message transmission for general adversary structures. *Cryptography and Communications*, 3(2):65–86, 2011.

[110] Keith M. Martin, Josef Pieprzyk, Reihaneh Safavi-Naini, and Huaxiong Wang. Changing thresholds in the absence of secure channels. In Josef Pieprzyk, Reihaneh Safavi-Naini, and Jennifer Seberry, editors, *ACISP 99*, volume 1587 of *LNCS*, pages 177–191. Springer, Heidelberg, April 1999.

[111] Keith M Martin, Rey Safavi-Naini, and Huaxiong Wang. Bounds and techniques for efficient redistribution of secret shares to new access structures. *The Computer Journal*, 42(8):638–649, 1999.

[112] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[113] Moni Naor and Adi Shamir. Visual cryptography. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 1–12, 1994.

[114] Maithili Narashima, Gene Tsudik, and Jeong Hyun Yi. On the utility of distributed cryptography in P2P and MANETs: the case of membership control. In *11th IEEE International Conference on Network Protocols (ICNP)*, pages 336–345, 2003.

[115] Mehrdad Nojoumian. Unconditionally secure proactive verifiable secret sharing using new detection and recovery techniques. In *14th Annual Conference on Privacy, Security and Trust (PST)*, pages 269–274. IEEE, 2016.

[116] Mehrdad Nojoumian and Douglas R. Stinson. On dealer-free dynamic threshold schemes. *Advances in Mathematics of Communications*, 7(1):39–56, 2013.

[117] Mehrdad Nojoumian and Douglas R. Stinson. Sequential secret sharing as a new hierarchical access structure. Cryptology ePrint Archive, Report 2015/403, 2015. https://eprint.iacr.org/2015/403.

[118] Mehrdad Nojoumian, Douglas R. Stinson, and Morgan Grainger. Unconditionally secure social secret sharing scheme. *IET Information Security*, 4(4):202–211, 2010.

[119] Satoshi Obana. Almost optimum t-cheater identifiable secret sharing schemes. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 284–302. Springer, Heidelberg, May 2011.

[120] Wakaha Ogata and Kaoru Kurosawa. Optimum secret sharing scheme secure against cheating. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 200–211. Springer, Heidelberg, May 1996.

[121] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[122] Josef Pieprzyk and Xian-Mo Zhang. Cheating prevention in secret sharing over GF(p^t). In C. Pandu Rangan and Cunsheng Ding, editors, *INDOCRYPT 2001*, volume 2247 of *LNCS*, pages 79–90. Springer, Heidelberg, December 2001.

[123] Josef Pieprzyk and Xian-Mo Zhang. Nonlinear secret sharing immune against cheating. In *DMS 2001: Seventh International Conference on Distributed Multimedia Systems*, pages 154–161, Taipei, Taiwan, 2001. Tamkang University.

[124] Josef Pieprzyk and Xian-Mo Zhang. On cheating immune secret sharing. *Discrete Mathematics and Theoretical Computer Science*, 6:253–264, 2004.

[125] Stephen Pohllig and Martin Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[126] Jyotirmoy Pramanik and Avishek Adhikari. Evolving secret sharing with essential participants. Cryptology ePrint Archive, Report 2020/1035, 2020. https://eprint.iacr.org/2020/1035.

[127] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

[128] R. S. Rees, Douglas R. Stinson, Ruizhong Wei, and G. H. J. van Rees. An application of covering designs: determining the maximum consistent set of shares in a threshold scheme. *Ars Combinatoria*, 53:225–237, 1999.

[129] Phillip Rogaway and Mihir Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 172–184. ACM Press, October 2007.

[130] Mike Rosulek. Universal composability from essentially any trusted setup. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 406–423. Springer, Heidelberg, August 2012.

[131] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Efficient node admission for short-lived mobile ad hoc networks. In *13th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2005.

[132] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Identity-based access control for ad hoc groups. In Choonsik Park and Seongtaek Chee, editors, *ICISC 04*, volume 3506 of *LNCS*, pages 362–379. Springer, Heidelberg, December 2005.

[133] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[134] Gustavus J. Simmons. Prepositioned shared secret and/or shared control schemes. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 436–467, 1989.

[135] Gustavus J. Simmons. How to (really) share a secret. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 390–448. Springer, Heidelberg, August 1990.

[136] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 480–509. Springer, Heidelberg, August 2019.

[137] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer, Heidelberg, May 1996.

[138] Douglas Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer-Verlag New York, 2004.

[139] Douglas R. Stinson. New general lower bounds on the information rate of secret sharing schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 168–182. Springer, Heidelberg, August 1993.

[140] Douglas R. Stinson and Scott A. Vanstone. A combinatorial approach to threshold schemes. *SIAM Journal on Discrete Mathematics*, 1(2):230–236, 1988.

[141] Douglas R. Stinson and Ruizhong Wei. Unconditionally secure proactive secret sharing scheme with combinatorial structures. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 200–214. Springer, Heidelberg, August 1999.

[142] Douglas R. Stinson and Ruizhong Wei. Combinatorial repairability for threshold schemes. *Designs, Codes, and Cryptography*, 86(1):195–210, 2018.

[143] Douglas R. Stinson and Sheng Zhang. Algorithms for detecting cheaters in threshold schemes. *Journal of Combinatorial Mathemtics and Combinatorial Computing*, 61:169–191, 2007.

[144] Sijun Tan, Brian Knott, Yuan Tian, and David J. Wu. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *2021 IEEE Symposium on Security and Privacy*, pages 1021–1038. IEEE Computer Society Press, May 2021.

[145] Tamir Tassa. Hierarchical threshold secret sharing. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 473–490. Springer, Heidelberg, February 2004.

[146] Martin Tompa and Heather Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1(2):133–138, June 1988.

[147] Marten Van Dijk. On the information rate of perfect secret sharing schemes. *Designs, Codes, and Cryptography*, 6(2):143–169, 1995.

[148] Zhifang Zhang, Yeow Meng Chee, San Ling, Mulan Liu, and Huaxiong Wang. Threshold changeable secret sharing schemes revisited. *Theoretical Computer Science*, 418:106–115, 2012.