ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Repairable Threshold Schemes with Malicious Security

Semester Project

Iana Peix

June 23, 2023

Advisors: Prof. Dr. Kenny Paterson, Shannon Veitch

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

Repairable threshold schemes are threshold schemes where, the participants can combine their shares to reconstruct a share for a participant who lost their share. Current protocols ensure share repairability under the assumption of shareholder honesty. Our focus is on the malicious security of repairable threshold schemes, as no constructions secure against malicious participants exist to date. We prove that existing schemes, which hinge on the honesty of shareholders, become vulnerable in presence of malicious players. Secondly, we introduce new schemes with different levels of malicious security, by leveraging various techniques.

Contents

Chapter 1

Introduction

In today's interconnected world, secure communication among thousands of devices in distributed systems is essential. Whether it's cloud computing, peer-to-peer networks, or Internet of Things (IoT) systems, the challenge lies in establishing secure communication even when only a threshold number of participants behaves honestly.

Threshold schemes allows to share a secret based on the assumption that a threshold number of participants act honestly. Repairable threshold schemes (RTS), incorporate the ability to repair or recover lost or corrupted shares, ensuring the long-term usability of the shared secret. At the moment existing protocols assume that shareholders are honest. However, these schemes often lack the ability to handle practical scenarios where the adversary is actively malicious.

We will conduct a survey on the existing constructions of repairable threshold schemes and survey if existing constructions are secure against malicious participants. Then, outline attacks on current constructions if the shareholders are assumed to be actively malicious.

Furthermore, the project will extend repairable threshold schemes to the malicious setting and evaluate the communication complexity associated with each technique.

Chapter 2

Background

2.1 Threshold schemes

Threshold schemes are cryptographic primitives used to protect sensitive information by distributing the secret among a group of participants. The fundamental idea is to split the secret into shares such that any authorized subset of participants can combine their shares to reconstruct the secret.

One famous real-life example of threshold schemes is the nuclear launch codes used in the Soviet Union, where three different keys from the first secretary, the defense minister, and the chief of the general staff were needed to launch a nuclear strike, designed to keep any one person from having ultimate authority.

More recently, threshold schemes have found applications in various domains, such as Ledger using Shamir's Secret Sharing to protect clients' cryptocurrency wallet seeds [1].

Here we briefly introduce threshold schemes using the same definitions as [13].

Definition 2.1 *Suppose t and n are positive integers such that $2 \leq t \leq n$. A (t, n)-threshold scheme is a method in which a dealer chooses a secret s and distributes a share to each of the n players $P_1, \ldots, P_n$ such that the following two properties are satisfied:*

- ***Correctness**: any authorized subset of t players can compute the secret from the shares they collectively hold.*
- ***Secrecy**: no subset of fewer than t players can determine any information about the secret.*

A threshold scheme consists of two algorithms: a share algorithm run by the dealer that receives as input the secret s and outputs n shares, and a reconstruct algorithm, *which receives as input at least t distinct, valid shares from the players and outputs the secret.*

2.1.1 Shamir's Secret Sharing Scheme

One of the most prominent threshold schemes is Shamir's Secret Sharing Scheme, introduced by Adi Shamir in 1979 [17].

The scheme represents a (t, n) - threshold scheme, where we represent the secret as a polynomial of degree $t-1$, where t is the threshold number of participants required to reconstruct the secret and n is the total number of participants. The polynomial's coefficients are chosen randomly, with the secret itself represented by the constant term (i.e. the term at $f(0) = s$).

Each participant is then given a share of the form $(i, f(i))$ which corresponds to a point on the polynomial.

To recover the secret, any subset of t participants can combine their shares and use polynomial interpolation on their shares to determine the polynomial's coefficients and, consequently, the secret.

Algorithm 1 Shamir's Secret Sharing Scheme

```
1: procedure Share Distribution($n, t, s$)
2:     Input: $n$ - the total number of participants, $t$ - the threshold number
   of participants and $s$ - the secret to be shared
3:     Choose a prime number $p > s$ to define the finite field $\mathbb{F}_p$
4:     Generate a random polynomial $f(x) = a_0 + a_1x + \ldots + a_{t-1}x^{t-1}$ of
   degree $t-1$ over $\mathbb{F}_p$, where $f(0) = s$
5:     for $1 \leq j \leq d$ do
6:         Send participant $i$ the share $(i, f(i))$
7:     end for
8: end procedure
9: procedure Reconstruct($(1, f(1)) \ldots (t, f(t))$)
10:    Input: At least $t$ different shares
11:    Let $x_1, x_2, \ldots, x_t$ be distinct $x$-values from the shares
12:    Let $y_1, y_2, \ldots, y_t$ be the corresponding $f(x_i)$-values from the shares
13:    Use Lagrange interpolation to compute the polynomial $f(x)$ that
   passes through the points $(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)$
14:    The secret $s$ is the constant term of the polynomial, i.e., $s = f(0)$
15:    Output: $s$
16: end procedure
```

This ensures that only authorized participants who have access to the required number of shares can reconstruct the secret, while maintaining the confidentiality of the secret information. Given a set of $t-1$ or fewer players, it is feasible to reconstruct a polynomial of degree at most $t-1$ that aligns with their shares. This means that there are as many polynomials passing through these $t-1$ points, as there are elements in the field, ensuring the confidentiality of the secret by not revealing any specific information about it.

For example, in a 4-out-of-7 threshold scheme, the secret would be split into seven shares, and any four of these shares would be required to reconstruct the original secret. This ensures that even if up to three participants are compromised, the secret information remains secure.

A more detailed example can be found in Appendix A.2.

Attacks on Shamir's Scheme

While Shamir's scheme is theoretically unconditionally secure in the passive adversary setting, there are several possible attacks in practical applications, where adversaries can be actively malicious.

The first attack in the actively malicious setting was presented by Tompa and Woll [19], where an adversary corrupts a participant and intentionally submits a false share during a reconstruction attempt.

Let us consider a (k, n) Shamir's secret sharing scheme where a secret is shared among participants. Suppose the adversary controls participant P_i and submits a false share $f(x_i)'$ instead of the correct share $f(x_i)$.

The secret can be expressed as:

$$s = \sum_{i=1}^{k} l_i f(x_i)$$

where l_i represents the Lagrange coefficients and $f(x_i)$ denotes the shares, for $i = 1, \ldots, k$.

If the adversary modifies the share and submits $f(x_i)' = f(x_i) + \delta$, the secret will be reconstructed as follows:

$$s' = \left(\sum_{j=1}^{i-1} l_j f(x_j) \right) + l_i f(x_i) + l_j \delta + \left(\sum_{j=i+1}^{k} l_j f(x_j) \right) = s + l_i \delta.$$

This attack has several consequences. It prevents honest participants from learning the correct secret, fails to alert other participants that the correct secret has not been reconstructed, and, most importantly, enables the adversary to learn the correct secret by exploiting their knowledge of both $f(x_i)$ and $f(x_i)'$ in the Lagrange interpolation.

2.2 Repairable threshold schemes

A natural question to ask is what happens if a share is corrupted in transit or lost and the dealer is not available anymore to resend the share. We would like to introduce a scheme where a player can repair their share by asking for help from other players. Such a scheme is called a repairable threshold scheme, which we abbreviate to RTS.

Here again we use the same definitions as used in [13].

Definition 2.2 *As before, let t and n be positive integers such that $2 \leq t \leq n$ and let $d \in N$ be such that $t \leq d \leq n-1$. Call d the repairing degree. A (t,n,d)-repairable threshold scheme, denoted (t,n,d)-RTS, is a (t,n)-threshold scheme which, in addition to the share and reconstruct algorithms, has a repair algorithm that allows a repairing player P_r to securely reconstruct their share with help from a set of d players, called the helping players.*

By definition, the threshold number needed to repair a share d must be bigger than the threshold for recovering the secret t.

Consider the scenario where d is smaller than t. In this case, a subset of d participants can always generate additional shares since d participants are sufficient for creating new shares. Therefore, the subset can simply generate more shares until they have enough to reconstruct the secret. Consequently, the threshold t to reconstruct must be smaller than or equal to d.

A (t,n,d)-repairable secret sharing scheme has *universal repairability* if any possible subset of at least d participants can repair another participant's share. If only some subsets can repair another participant's share we say that the scheme has *restricted repairability*.

One potential benefit of restricted repairability is the possibility of achieving more efficient schemes. Efficiency, in this context, means the communication complexity of the repairing algorithm. Where *communication complexity* is the sum of the sizes of all messages transmitted during an algorithm, divided by the size of the secret.

2.3 Enrollment Repairable Threshold Schemes

In this section, we present the enrolment RTS, initially proposed by Stinson and Wei in [18]. The purpose of presenting the enrollment RTS is to introduce a specific RTS schemes, which we will analyze in the future chapters.

The share distribution happens in the same way as in Shamir's scheme in Chapter 2.1.1. Starting with a (t,n)-Shamir threshold scheme in $\mathbb{F}_Q$, where Q is the choosen size of the secret space. The share for a player P_ℓ is denoted as $\phi_\ell = f(\ell)$, where $f(x) \in \mathbb{F}_Q[x]$ is a random polynomial of degree at most $t-1$ with the secret $f(0) = s$.

If our aim is to repair the share for a player P_r and assume that this share is being repaired by players $P_1, \ldots, P_d$ where $d > t$. We can describe the repair algorithm as follows:

Algorithm 2 Enrollment RTS

```
1: procedure REPAIR ALGORITHM($P_1$,$P_2$....$P_d$)
Require: $d \geq t$
2:     for $1 \leq i \leq t$ do
3:         $P_i \leftarrow$ compute random $\delta_{ji}$ $1 \leq j \leq d$ so that $\zeta_i\phi_i = \sum_{i=1}^{d} \delta_{ji}$
4:         for $1 \leq j \leq d$ do
5:             $P_i$ sends $\delta_{ji}$ to $P_j$
6:         end for
7:     end for
8:     for $1 \leq j \leq d$ do
9:         $P_j$ computes $\sigma_j = \sum_{j=1}^{d} \delta_{ji}$
10:    end for
11:    for $1 \leq i \leq t$ do
12:        $P_i$ sends $\sigma_i$ to $P_r$
13:    end for
14:    $P_r$ computes $\sum_{i=1}^{t} \sigma_i$ finding their share.
15: end procedure
```

The communication complexity of the repair algorithm is the sum of the sizes of all the messages transmitted during the protocol divided by the bit-length of the secret. As every message is an element of $\mathbb{F}_Q[x]$ as is the secret, the communication complexity is equal to the number of sent messages. As $t(t-1)$ messages are sent in line 5 and t messages on line 12 the total complexity is $O(t^2)$. In [13] this total complexity is lowered to $O(\frac{1}{2}(t(t+1)))$ by optimizing the number of messages sent in the repair algorithm.

2.4 Regenerating codes Repairable Threshold Schemes

2.4.1 Regenerating Codes

In a distributed storage system, data is stored across multiple nodes or servers. If one of these nodes fails, the lost data can be reconstructed using the remaining data on the other nodes. However, this process can require a large amount of data to be transmitted between the nodes. Regenerating codes are a type of error-correcting code used in distributed storage systems. These codes are designed to reduce the amount of data that needs to be transmitted when repairing lost or corrupted data, while retaining the storage efficiency of traditional codes [11].

Re-using definition from [8], we define regenerating code as follows.

Definition 2.3 *An (n, k, d, α, β) regenerating code distributes a file, represented as a polynomial f in $F[x] \leq k - 1$, by encoding it and sending elements of the encoding to n nodes where each node stores α bits of data. A failed node can recover (repair) its share by accessing size β data from d surviving nodes, and we denote the repair bandwidth as $\gamma = d\beta$. Any k nodes are able to reconstruct the original file f when using their collective stored data.*

2.5 Ramp Schemes

We will briefly introduce the concept of *ramp schemes*, which are often used in combinatorial RTS designs. Using the same definition as [12]:

Definition 2.4 (Ramp Scheme) *Let n be the number of participants in the scheme, and let τ_1 and τ_2 be the lower and upper thresholds, respectively, such that* $1 \leq \tau_1 < \tau_2 \leq n$. *A* (τ_1, τ_2, n)-ramp scheme *is defined as follows:*

- ***Reconstruction**: Any subset of the n participants of size τ_2 can determine the secret from the shares they hold.*
- ***Secrecy**: No subset of the n participants consisting of at most τ_1 participants is able to gain any knowledge about the secret.*

A (τ_1, τ_2, n)-ramp scheme *is equivalent to a (τ, n)-threshold scheme when $\tau_2 = \tau_1 + 1 = \tau$.*

2.6 Combinatorial Repairable Threshold Schemes

Combinatorial RTS, as demonstrated in [18], offers a method for constructing repairable threshold schemes. The intuition behind this construction is to give each player a subset of shares from a different threshold or ramp scheme called a base scheme. As each participant now has a subset of shares, these subsets can then be used to repair shares of other players.

If we use a (σ, m) Shamir scheme, implemented over a finite field F_Q as a base scheme, then to construct the repairable threshold scheme, we assign each player a subset of d out of the m shares. To do this we need a set system or design consisting of n blocks of size d, defined on a set of m points, which we refer to as the distribution design.

In the resulting (τ, n)-threshold scheme, each share is composed of d subshares. Let $s_1, \ldots, s_m$ denote the shares in the base scheme, and let $1, \ldots, m$ represent the points in the distribution design. Each player P_i corresponds to a block B_i in the distribution design. For each point $b \in B_i$, player P_i is given the corresponding sub-share s_b. To satisfy the threshold property we need the following conditions to hold:

- The union of any τ blocks contains at least σ points.
- The union of any $\tau - 1$ blocks contains at most $\sigma - 1$ points

This explanation provides a simplified understanding of the explanation given in [12] for combinatorial repairable threshold schemes.

A common choice for the distribution design is a balanced incomplete block design (BIBD). We use the definition from [13]:

Definition 2.5 *A (m,k,λ)-balanced incomplete block design,(m,k,λ)-BIBD,is a design such that*

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

Consider the following example from [13]: a $(2,12,3)$-RTS, this share algorithm uses a $(9,3,1)$-BIBD (balanced incomplete block design) as a basis, which is a repairable $(2,3,5)$-distribution design. For convenience, we will label the nine shares output from the ramp scheme $1,2,...,9$. We then construct the $(2,12,3)$-RTS by allocating sub-shares from the ramp scheme to the twelve players.

$$\begin{array}{lll}
P1 \leftarrow \{1,2,3\} & P5 \leftarrow \{2,5,8\} & P9 \leftarrow \{3,4,8\} \\
P2 \leftarrow \{4,5,6\} & P6 \leftarrow \{3,6,9\} & P10 \leftarrow \{1,6,8\} \\
P3 \leftarrow \{7,8,9\} & P7 \leftarrow \{1,5,9\} & P11 \leftarrow \{2,4,9\} \\
P4 \leftarrow \{1,4,7\} & P8 \leftarrow \{2,6,7\} & P12 \leftarrow \{3,5,7\}
\end{array}$$

To reconstruct the secret any two players can combine their shares, giving them at least five distinct sub-shares from the $(3,5,9)$-ramp scheme allowing them to recover the secret via the reconstruct algorithm.

Consider the scenario where player P_5 needs to repair their share. They can have assistance from players P_1, P_2 and P_3, who would each send the sub-shares 2, 5 and 8, respectively. We note that not all subset can repair a specific share, so this scheme provides restricted repairability.

2.7 Verifiable secret sharing

In the previous sections, we introduced secret sharing schemes. Which rely on trusting the dealer to distribute the shares correctly. However, in real-world scenarios, we cannot always assume the dealer's trustworthiness. Verifiable Secret Sharing (VSS), introduced in [6], addresses this challenge by providing a mechanism to verify the correctness of each share, even in the presence of malicious participants or a potentially corrupted dealer. VSS plays a critical role in various multi-party computation schemes, including Byzantine agreement and secure multi-party computation [5].

In a VSS scheme, the dealer shares the secret in a verifiable manner, even in the presence of malicious parties and a potentially corrupted dealer. The scheme consists of a Sharing phase, where the secret is distributed, and a Reconstruction phase, where the shared secret is reconstructed. For convenience, we can separate the part of the scheme where the validity of shares is verified into a separate Verification phase. Given a set of valid shares which make up an authorized set, there exists a unique secret which is output by the Reconstruct algorithm. Our main goal is that even if the dealer D is corrupt, in any execution of Sharing the Reconstruction of the honest parties defines some value s which is output by all honest parties at the end of Reconstruction.

A Verifiable Secret Sharing scheme (VSS) includes a verification algorithm where participants $P_1, \ldots, P_n$ can use their shares $x^1_{t+1}, \ldots, x^n_{t+1}$ to verify the validity of their shares.

One commonly used example of a VSS scheme is Feldman's scheme, which combines Shamir's secret sharing scheme with a homomorphic encryption scheme [9]. However, it is important to note that Feldman's scheme is only secure against computationally-bounded adversaries.

2.8 Proactive Secret Sharing

Proactive Secret Sharing (PSS) is an extension of secret sharing schemes that tackles the problem of adversaries who can corrupt parties over an extended period of time. In PSS an adversary is capable of corrupting all n parties over a long period of time, but controls no more than a threshold t at the same time. To mitigate this, PSS schemes periodically refreshes the shares of the secret and invalidate old shares. This also allows for the recovery of the current state held by a participant in case a share has been previously destroyed by the adversary [16, 7].

This means that a PSS scheme includes two additional algorithms. The refresh algorithm allows participants $P_1, \ldots, P_n$ to use their shares from phase ϕ to generate new random shares $x^1_{t+1}, \ldots, x^n_{t+1}$ for the same secret, which are then distributed to each respective participant. The recover algorithm is used when a corrupted node P_r needs to obtain a new share, and it involves contacting d uncorrupted nodes who combine their shares to compute a new share for P_r.

We can extend Shamir's secret sharing scheme to incorporate proactive features. The proactive (t, n) Shamir secret sharing scheme over a finite field F with evaluation points $A = \alpha_1, \ldots, \alpha_n \subseteq F$ is defined as follows [8]:

- **Refresh:** Each participant P_i generates a random polynomial δ_i of degree k conditioned on $\delta_i(0) = 0$ and sends $\delta_i(\alpha_j)$ to P_j for all $j \neq i$. Then, each P_i updates their share as $x^i_{t+1} \leftarrow x^i_t + \sum_j \delta_j(\alpha_i)$ (and erases all intermediate values used to compute x^i_{t+1}).

- **Recover:** For each corrupted node $P_r \in B$, each $P_i \in D$, a set of non-corrupted nodes, does the following: Generate a uniformly random polynomial of degree k, ξ_i, such that $\xi_i(\alpha_r) = 0$. Then, send $\xi_i(\alpha_j)$ to P_j for all $P_j \in D$. Each $P_j \in D$ updates their share as $x^j_t \leftarrow x^j_t + \sum_{i, P_i \in D} \xi_i(\alpha_j)$. Finally, each $P_i \in D$ sends its updated share x^i_t to P_r and P_r interpolates them to get its original share x^r_t.

2.8.1 `DM-Recover` Construction

In [7], Dolev introduces a new Proactive Secret Sharing Scheme, which provides a scheme secure against $t < n$ passive adversaries and $t < \frac{n}{2}$ active adversaries. As mentioned earlier, PSS includes a recovery protocol that enables honest parties to recover their shares. This recovery feature shares similarities with repairable threshold schemes and so, in our evaluation, we will compare the `DM-Recover` with the proposed schemes in the upcoming chapters.

First we will look at `DM-Share` where we can see how the shares are send between players as this is quite different from approach seen in Shamir's Secret Sharing.

Algorithm 3 `DM-Share`: Secret Sharing for Dishonest Majorities

```
1: procedure ShareSecret($s, n, d$)    ▷ Dealer shares secret $s$ with $n$ parties using $d$ random summands
2:     Dealer $P_D$ chooses $d$ random values $s_1, \dots, s_d$ such that $\sum_{i=1}^{d} s_i = s$
3:     for $i = 1$ to $d$ do
4:         $P_D$ generates a random polynomial $f_i(x)$ of degree $i$ with $f_i(0) = s_i$
5:         $P_D$ computes and broadcasts to each $P_r$ commitments of the coefficients of $f_i(x)$
6:         for each share $sh_{i,r} = f_i(\alpha_r)$ do
7:             Each receiving party $P_r$ locally computes commitment $c_{i,r}$ based on the homomorphic commitment scheme
8:             $P_D$ sends the opening information $o_{i,r}$ to party $P_r$
9:             Party $P_r$ broadcasts a complaint bit indicating whether $o_{i,r}$ correctly opens $c_{i,r}$ to some value $sh_{i,r}$
10:        end for
11:        for each inconsistent share $sh_{i,j}$ do
12:            $P_D$ broadcasts the opening information $o_{i,j}$, and if $o_{i,j}$ opens $c_{i,j}$, party $P_j$ accepts $o_{i,j}$
13:            Otherwise, $P_D$ is disqualified and a default sharing of a default value is used
14:        end for
15:    end for
16:    for each receiving party $P_r$ do
17:        $P_r$ outputs its $d$ shares $(sh_{1,r}, o_{1,r}), \dots, (sh_{d,r}, o_{d,r})$ and all commitments
18:    end for
19: end procedure
```

`DM-Share` requires $O(n^2)$ communication to share a single secret s. The secret s is first split into $O(n)$ summands, and then each summand is split into $O(n)$ shares because $d = O(n)$.

Recovering Shares for Dishonest Majorities (DM-Recover)

The main idea behind `DM-Recover` is other parties generating and verifiably sharing random recovery polynomials that evaluate to zero at the rebooted party's evaluation point. They then add their local shares of the current sharing polynomials to the shares of these random recovery polynomials, resulting in new shared random recovery polynomials. These shares are sent to the rebooted party, which can then interpolate these polynomials to recover its share without learning anything about the secret. Using commitments the protocol ensures that as long as there is at least $\frac{n}{2}$ honest parties,

Algorithm 4 `DM-Recover` Algorithm: Recovering Shares for Dishonest Majorities

```
1: procedure SHARESECRET($s, n, d$)    ▷ Dealer shares secret $s$ with $n$ parties using $d$ random summands
2:     Party $P_{rc}$, sharing polynomials $f_i(x)$ for $i \in \{1, ..., d\}$ at $\alpha_{rc}$
3:     for $i \in \{1, ..., d\}$ do
4:         Each party $P_j$ generates a random polynomial $g_{j,i}(x)$ of degree $i$ with $g_{j,i}(\alpha_{rc}) = 0$
5:         Each party $P_j$ verifiably shares $g_{j,i}(x)$ with other $n - 2$ parties, excluding $P_{rc}$
6:         for each share $sh_{r,j,i} = g_{j,i}(\alpha_r)$ do
7:             Each receiving party $P_r$ computes commitment $cr_{j,i}$ and checks $g_{j,i}(\alpha_{rc}) = 0$
8:             $P_j$ sends opening information $or_{j,i}$ for commitment $cr_{j,i}$ to party $P_r$
9:             $P_r$ broadcasts complaint bit indicating correctness of opening $or_{j,i}$
10:        end for
11:        for each inconsistent share $sh_{r,j,i}$ do
12:            $P_j$ broadcasts opening information $o_{j,i}$
13:            if $o_{j,i}$ opens $c_{j,i}$ then
14:                $P_r$ accepts $o_{j,i}$
15:            else
16:                $P_j$ is disqualified and added to set $B$ of parties with incorrect shares
17:            end if
18:        end for
19:        Each party $P_r$ adds received shares $sh_{r,i,j}$ to its share of $f_i(x)$: $z_{r,i} = f_i(\alpha_r) + \sum_{j=1}^{n-2} sh_{r,i,j}$
20:        Each party $P_r$ sends $z_{r,i}$ to $P_{rc}$
21:        $P_{rc}$ interpolates recovery polynomial $z_i(x)$ and obtains current share $z_i(\alpha_{rc}) = f_i(\alpha_{rc})$
22:    end for
23:    Current share $z_i(\alpha_{rc}) = f_i(\alpha_{rc})$ for $i \in \{1, ..., d\}$
24: end procedure
```

the secret information remains protected.

The complexity of `DM-Recover` when handling a single secret is $O(n^4)$, while for managing multiple secrets, the complexity is reduced to $O(n^3)$.

Chapter 3

Definitions and Threat Model

In this chapter, we provide definitions of correctness and security for threshold schemes. Additionally, we introduce new definitions specifically related to the `Repair` algorithm in Repairable threshold schemes. We also present a high-level overview of the setting we consider in this work. Furthermore, we discuss the specific threat model that will be utilized throughout the rest of this work.

3.1 Correctness and Security

As seen in Chapter 2.2, threshold schemes aim to distribute a secret among a group of participants in such a way that the secret can only be reconstructed if at least a certain number of participants collaborate. Recall, these schemes satisfy the following definitions:

- **Correctness**: The secret s can be reconstructed by any qualified set of participants, otherwise $\perp$ is output.
- **Privacy**: No unauthorized subset is able to gain any knowledge about the secret.

These definitions apply to threshold schemes in general, regardless of their specific implementation or parameters.

For repairable threshold schemes, these properties must still hold. Additionally, the introduction of a `Repair` algorithm requires us to define new properties. We provide slightly modified definitions of correctness and security of the `Repair` algorithm next.

Definition 3.1 (Correctness of `Repair`) *The `Repair` algorithm outputs a valid share equal to the repairing player's, P_r, initial share or a null value $\perp$.*

Definition 3.2 (Privacy of `Repair`) *No unauthorized subset of players learns any additional information about the secret during the repair protocol.*

These definitions are consistent with prior definitions in literature from robust secret sharing [19] which consider misbehaving participants.

The definition of correctness for the `Repair` algorithm in repairable threshold schemes differs from the traditional threshold schemes correctness due to the unique nature of RTS.

When using repairable threshold schemes, we can imagine a scenario involving share distribution, multiple rounds of repair, and final reconstruction with a qualified subset of participants.

We assume that share distribution and secret reconstruction have already been handled securely against malicious adversaries, as demonstrated in prior work on Verifiable Secret Sharing (VSS) [6, 15] and robust secret sharing [4]. Therefore, the repair protocol is considered the most vulnerable component of the scheme and the primary focus of our research.

Our threat model assumes that participants in the scheme can be actively malicious during the repair process. Specifically, participants may deviate from the protocol, submit incorrect information, or collude with other malicious participants. The goal is to identify and mitigate vulnerabilities in repairable threshold schemes under such adversarial conditions.

It is important to note that while the definition of correctness differs, the definition of privacy remains consistent with prior definitions in literature. The privacy of the repair process ensures that no unauthorized subset of participants gains any additional information about the secret during the repair protocol, maintaining the confidentiality of the secret throughout the repair process.

3.1.1 Security types

- **Information-theoretic security**: guarantees that an adversary with infinite computational resources learns no information about the secret.
- **Computational security**: guarantees that an adversary with "reasonable" computational resources learns no information about the secret.
- **Statistical security**: a small amount of information is potentially revealed about the secret, independent of the computing power of an adversary. [14]

For each of the proposed protocols we will explain the guarantee it manages to achieve.

3.2 Threat model

Here we will define the capabilities of the adversary for the rest of this work. We assume that during the initialization or share distribution phase, the adversary remains inactive, and does not learn any information about the secret. In particular, the adversary does not get to see the shares that the dealer sends to the players. We stress that the adversary cannot corrupt the dealer and so we assume that share distribution occurred correctly. VSS provides solutions for the case of a malicious dealer, offering various alternatives documented in Chapter 2.7 and related literature.

In the repair phase, we assume that the dealer is no longer active. Of course if the dealer is honest and active during the repairing phase, the solution is trivial as the dealer can simply reshare the lost or corrupted shares with the participants. This is pointed out in [2].

After the sharing phase, the adversary chooses up to t players to corrupt. Once a player P_i is corrupted, the adversary has full control over P_i, meaning that they learn P_i's share and control the information P_i may send. The corrupted player can either behave honestly, refrain from sending any messages, or intentionally send incorrect share or an invalid message.

3.2.1 Assumptions

We begin by stating several assumptions, which are used throughout the remainder of this work. These assumptions include:

- Each party is connected to all other parties through confidential and authentic communication channels, ensuring that messages are only visible to the intended recipients and verified to be from the sender.
- Connections are perfectly synchronous, meaning that messages arrive instantly at their destination.
- Parties can use a broadcast channel, which allows them to send the same message to all other parties.
- The dealer is honest and only active during the share distribution algorithm.

3.2.2 Adversaries

In the majority of previous work on repairable threshold schemes, the adversaries are assumed to be passive. Passive adversaries observe the shares held by participants under their control. They may use this knowledge to infer additional information about the secret; however, they are assumed to follow the protocol correctly. These adversaries are also sometimes known as honest-but-curious.

In practice, we might also be interested in modeling active adversaries, which are able to deviate from the protocol in an arbitrary fashion.

While modeling passive adversaries provides valuable insight, there is a need to consider scenarios involving active adversaries, as if we would like to implement these protocols in practice, active adversaries are much closer to the possible real-world threats. Active adversaries have the capability to deviate from the protocol in arbitrary ways, introducing new challenges and security concerns.

Chapter 4

Attacks

In this section we present several attacks that violate the definitions of correctness or privacy in the repairable threshold schemes as introduced in Chapter 2.2. As these attacks are possible in our adversarial model, they demonstrate that combinatorial, regenerating codes and enrollment RTS do not satisfy malicious security. These attacks motivate our work to create new maliciously secure repairable threshold schemes.

4.1 Combinatorial repairable threshold scheme attack

In this section, we delve into an adversarial scenario in the context of a combinatorial repairable threshold scheme. We reuse the example from [13] seen in detail in Chapter 2.6. Let's look at an example where an active adversary controls one participant and behaves dishonestly.

$P1 \leftarrow \{1,2,3\}$	$P5 \leftarrow \{2,5,8\}$	$P9 \leftarrow \{3,4,8\}$
$P2 \leftarrow \{4,5,6\}$	$P6 \leftarrow \{3,6,9\}$	$P10 \leftarrow \{1,6,8\}$
$P3 \leftarrow \{7,8,9\}$	$P7 \leftarrow \{1,5,9\}$	$P11 \leftarrow \{2,4,9\}$
$P4 \leftarrow \{1,4,7\}$	$P8 \leftarrow \{2,6,7\}$	$P12 \leftarrow \{3,5,7\}$

After the shares are created and distributed, suppose the active adversary corrupts player P_7.

Imagine that now player P_2 needs to repair their share and seeks assistance from players P_4, P_7, and P_8. However, now P_7 can simply send a wrong sub-share back to P_2. For example, instead of sending sub-share 5, P_7 might send sub-share 1. As a result, P_2 reconstructs their share as $4, 1, 6$.

Later, when pairs (P_2, P_{10}) or (P_2, P_4) attempt to reconstruct the secret, they combine their sub-shares finding $\{1, 4, 6, 8\}$ and $\{1, 4, 6, 7\}$ respectively.

We can formalize this attack as follows, suppose the scheme employs a base (m, l)-threshold scheme and the shares $(1, 2, ..., l)$ are created and distributed among n participants.

Let the shares of the participants be denoted as follows:

$$P_i \leftarrow \{a_{i1}, a_{i2}, ..., a_{ij}\}, \quad for \quad i = 1, 2, ..., n \text{ and } \quad j = 1, 2, ..., \tau$$

where a_{ij} is a sub-share of P_i and an active adversary takes control of a participant P_a. When P_x needs to repair their share and seeks assistance from a set of participants including P_a, the adversary-controlled P_j sends an incorrect sub-share to P_x. Instead of sending the correct sub-share a_{ay}, P_a sends an incorrect sub-share a_{az}, where $z \neq y$. As this is a valid sub-share P_x reconstructs their share as:

$$P_x = \{a_x 1, a_x 2, ..., a_a y, ..., a_x t\} \neq \{a_x 1, a_x 2, ..., a_a z, ..., a_x t\}$$

breaking the definition of Correctness of `Repair` defined in Chapter 3.1 and showing that the combinatorial repairable threshold scheme is not secure against actively malicious adversary.

4.2 Regenerating codes repairable threshold scheme attack

In this section, we investigate a scenario involving an active adversary during the repair process in a regenerating codes repairable threshold scheme, as detailed in [13] and explained in Appendix 2.4. Just like the attack discussed in Chapter 4.1, the adversary can disrupt the repair process of the regenerating codes RTS by providing a sub-share that differs from the one requested, resulting in an incorrect repair of the lost share.

Consider a (t, n, d)-regenerating code RTS, where t is the threshold, n is the number of participants, and d is the number of participants required to repair a share. Suppose that an adversary controls a participant P_z and behaves dishonestly. When another participant P_j needs to repair their share, they request assistance from d other participants, which includes the corrupted participant P_z.

Using the regeneration algorithm outlined in [13], we can see how the repair process can be tampered with by the dishonest participant. If P_z sends an incorrect value instead of the correct one during the repair process, P_j would inadvertently reconstruct an incorrect value for their share. Int this example, let all computations be performed in the field $\mathbb{Z}_{11}$.

For example, in the correct scenario, a node P_r should reconstruct the following:

$$(\Psi_{\text{repair}})^{-1} \times \begin{bmatrix} 3 \\ 2 \\ 9 \end{bmatrix} = \begin{bmatrix} 9 & 9 & 8 \\ 4 & 1 & 1 \\ 10 & 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 9 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 6 \end{bmatrix}$$

But if P_z behaves dishonestly and sends a wrong value (say 1 instead of 9), the node P_r reconstructs this instead:

$$(\Psi_{\text{repair}})^{-1} \times \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 & 9 & 8 \\ 4 & 1 & 1 \\ 10 & 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 4 \\ 1 \end{bmatrix}$$

This leads to the generation of an incorrect share, breaking the definition of Correctness of `Repair` defined in Chapter 3.1.

To generalize, when P_j seeks to repair their share, they construct a repair matrix, Ψ_{repair}, from consisting of the rows of Ψ related to the helper nodes (i.e. for helping node P_i add row i to the Ψ_{repair}), and then calculates the inverse.

For the correct scenario, the node P_j should reconstruct their share as follows:

$$(\Psi_{\text{repair}})^{-1} \times \begin{bmatrix} \alpha \\ \vdots \\ \alpha_d \end{bmatrix}$$

But if P_z behaves dishonestly and the repair matrix to $\alpha' = \alpha + \gamma$, the node P_j reconstructs an incorrect share:

$$(\Psi_{\text{repair}})^{-1} \times \begin{bmatrix} \alpha \\ \vdots \\ \alpha' \end{bmatrix} = \Psi_{\text{repair}}^{-1} \times \begin{bmatrix} \alpha \\ \vdots \\ \alpha \end{bmatrix} + \Psi_{\text{repair}}^{-1} \times \begin{bmatrix} 0 \\ \vdots \\ \gamma \end{bmatrix}$$

Which leads to an incorrect share being reconstructed and breaking the definition of Correctness of `Repair` defined in Chapter 3.1.

4.3 Enrollment repairable threshold scheme attack

In this section, we analyze a possible attack on the enrollment scheme discussed in Chapter 2.3. We consider a situation where an adversary controls one participant and behaves dishonestly during the repair process. The repair process involves each helping node P_i transmitting values $\delta_{j,i}$ to other players and subsequently transmitting values $\sigma_{j,i}$.

Assume that the adversary controlling P_i submits incorrect values of $\delta'_{j,i} = \delta_{j,i} + \beta$, this leads to incorrect computation of the σ_j value in the subsequent steps of the repair algorithm.

Initially, each player P_j computes σ_j as the sum of $\delta_{j,i}$ values:

$$\sigma_j = \sum_{i=1}^{t} \delta_{j,i} \tag{4.1}$$

However, if the adversary modifies the value of $\delta_{j,i}$ for a particular helping node P_j, the resulting σ_t becomes:

$$\begin{aligned} \sigma'_j =& \sum_{i=1}^{t} \delta_{j,i} + \beta \\ \sigma'_j =& \sigma_j + \beta \end{aligned} \tag{4.2}$$

During the repair process, the repairing player P_r computes their share ϕ_r using the equation:

$$\phi_r = \sum_{i=1}^{t} \sigma_i \tag{4.3}$$

But if for example the helping node P_t submits an incorrect value σ'_t, it would lead to incorrect computation of the share ϕ_r. Instead the repairing player P_r receives:

$$\phi'_r = \sum_{i=1}^{t-1} \sigma_i + \sigma'_t \tag{4.4}$$

As σ'_t can be chosen to be completely random P_r now has a random value ϕ'_r which gives them no information about their original share. Breaking the Correctness of `Repair` property for Repairable Threshold Schemes we have defined in Chapter 3.1.

Even more if in the future P_r and P_t are a part of a group who want recover the secret, then P_t can recover the valid share of P_r using the following equation:

$$\phi_r = \phi'_r + \sigma_t + \sigma'_t \tag{4.5}$$

Consequently, P_t becomes the only player with the correct value of the secret s, while all other players have an invalid value s'.

Chapter 5

Maliciously Secure Schemes

After looking at the repairable threshold schemes presented in Chapter 2, and looking at the possible attacks in Chapter 4 we observe that these schemes do not offer malicious security. To address this, we will propose three modified schemes in this chapter.

5.1 Maliciously Secure Combinatorial Repairability using Redundancy

In this section, we present a modification of the combinatorial repairable threshold scheme discussed in Chapter 2.6. Our goal is to provide malicious security by introducing redundancy into the scheme. Instead of requesting a single player to send their sub-share, the repairing party now solicits all players with the relevant sub-shares to send their information. The sub-shares are then checked for consistency.

To illustrate this scheme, let us consider an example from [13] with a $(2, 12, 3)$-RTS. We allocate sub-shares from the ramp scheme to the twelve players defined by the design, as follows:

$$\begin{array}{lll} P1 \leftarrow \{1,2,3\} & P5 \leftarrow \{2,5,8\} & P9 \leftarrow \{3,4,8\} \\ P2 \leftarrow \{4,5,6\} & P6 \leftarrow \{3,6,9\} & P10 \leftarrow \{1,6,8\} \\ P3 \leftarrow \{7,8,9\} & P7 \leftarrow \{1,5,9\} & P11 \leftarrow \{2,4,9\} \\ P4 \leftarrow \{1,4,7\} & P8 \leftarrow \{2,6,7\} & P12 \leftarrow \{3,5,7\} \end{array}$$

Suppose that P_5 wishes to repair their share. To obtain their first sub-share they contact P_1, P_8 and P_{11}. For their second sub-share, they contact P_2, P_7 and P_{11} and for third sub-share P_3, P_9 and P_{10}. To reconstruct their share, player P_5 checks that all the values for each sub-share the mode is calculated and this value is output.

This approach ensures that if one player submits an incorrect value for each sub-share it will be detected.

5.1.1 Formal Description

Let us provide a formal description of the redundancy repair scheme, which we refer to as Redundancy Repair.

The shares belonging to a player P_i are written as $(s_{i1}, s_{i2}, s_{i3}, ..., s_{in})$ where n is the number of sub-shares and the players who have a sub-share s_{ij} belong to the set S_{ij}, which is of size τ. The size of the set of the set S_{ij} depends on the distribution design used in the combinatorial

As mentioned before the repairing player knows the members of each group S_{ij} and helping nodes know what sub-share they need to send to P_r.

Algorithm 5 Redundancy Repair

```
Require: The dealer is honest and has previously sent each player their shares.
Require: Each player knows who to contact to repair their shares.
1: The repairing player $P_r$ contacts sets $S_{r1}$, $S_{r2}$ ... $S_{rn}$ for his sub-shares $(s_{r1}, s_{r2}, s_{r3}, ..., s_{rn})$
2: for $1 \leq i \leq n$ do
3:     Players in $S_{ri}$ send their share $s_{ri}$ to $P_r$
4: end for
5: return For each sub-share set $s_{ri}$ to be the mode of received sub-shares, if $P_r$ has several values as a mode output $\perp$ .
6: return Set $(s_{r1}, s_{r2}, s_{r3}, ..., s_{rn})$ as the new share of $P_r$.
```

The confidentiality of the secret is maintained as long as the threshold is not violated against passive adversaries. This scheme provides resilience against a few malicious players. However, if the majority of the participants in any of the sub-share sets S_{ri} are malicious, they can manipulate the mode, leading to the incorrect reconstruction of the share. This scheme is secure against active adversaries, up to a size of $\frac{|S_{ri}|}{2}$. As all S_{ij} are of size τ, this scheme maintains its correctness property in the presence of up to $\frac{\tau}{2}$ adversaries.

The proof of security will be presented in the next section, where we analyze the security properties of the scheme.

5.1.2 Correctness and Privacy

In this section, we will analyze whether the modified scheme fulfils the properties of correctness and privacy, as defined in Chapter 3.1.

Proof of Correctness

We assume that at most $\frac{\tau}{2}$ players are malicious. In order to achieve correctness, the repairing player must either receive their initial share P_r or output a null value $\perp$. Let us consider the scenario where the player receives a reconstructed share P_r' that is not equal to their original share P_r. This can only happen if, for at least one sub-share s_{ri}, the repairing player P_r has received at least $\frac{\tau}{2} + 1$ s_{ri}' values from repairing nodes in S_{ri} such that $s_{ri}' \neq s_{ri}$.

However, for this to occur, it would require at least $\frac{\tau}{2} + 1$ nodes in the set S_{ri} to be corrupted. Since our scheme has a malicious security threshold of $\frac{\tau}{2} - 1$, there is at least $\frac{\tau}{2} + 1$ honest players in S_{ri} sending a correct value s_{ri}. This gives a contradiction, as P_r must have received $s_{ri}' \neq s_{ri}$ from every helping node. It is impossible for all nodes in S_{ri} to be corrupted. Therefore, the correctness property is fulfilled in our scheme.

Proof of Privacy

To evaluate the privacy of the repair process, we need to ensure that no unauthorized group of players gains any knowledge about the secret during the repair protocol.

To gain knowledge about the secret, a repairing player would need to receive information that they did not have knowledge of before. However, as mentioned in our algorithm, the indexes of different sub-shares are public, and the only way a repairing player can receive a different sub-share is if a corrupted player provides it. In this case, the adversary already had knowledge of this sub-share, and the repairing player gains no new information.

Furthermore, the helping player are only contacted by the repairing player using the index of the sub-share they want to receive from the group S_{ri}. The only way helping nodes can receive new information is if the repairing player sends them their own sub-share instead. However, this can only happen if the repairing player is corrupted, which means that the malicious player already had knowledge of this sub-share and gains no additional knowledge. As all communication channels are secure against eavesdroppers, no unauthorized players can gain any knowledge about the secret. Therefore, the privacy property is preserved in the modified scheme.

In conclusion, the modified scheme satisfies the properties of correctness and privacy. It ensures that the repairing player receives the correct value of their sub-share, and it guarantees that unauthorized players cannot gain any knowledge about the secret during the repair protocol.

5.1.3 Complexity

It is evident that this redundancy has implications on the complexity of this modified combinatorial repairable threshold scheme. Specifically, to perform a repair, all players with the same sub-shares as the repairing player are now helping nodes and need to send their sub-shares to the repairing player.

For instance, in our previous example of a $(2, 12, 3)$-RTS, where each player owns three sub-shares, the repairing player, such as P_5, would need to contact the nine other players who possess the same sub-shares. This increases the number of messages exchanged between the players, changing the structure of the repair scheme from a $(2, 12, 3)$-RTS to a $(2, 12, 9)$-RTS.

To generalize this observation for all possible schemes, we can refer to certain properties of Balanced Incomplete Block Designs (BIBD) on which the combinatorial threshold scheme is based [13].

Definition 5.1 *In an (m, k, λ)-BIBD, every point occurs in exactly*

$$\tau = \frac{\lambda(m-1)}{k-1}$$

blocks.

and

Definition 5.2 *An (m, k, λ)-BIBD has exactly*

$$b = \frac{mr}{k} = \frac{m\lambda(m-1)}{k(k-1)}$$

blocks.

These definitions provide information about the structure of a BIBD and specify the number of blocks a BIBD must have. We also include a quick reminder on distribution designs, also from [13].

Definition 5.3 *A (t, l_1, l_2)-distribution design is a design that satisfies the following two properties:*

- *the union of any t blocks contain at least l_2 points,*
- *the union of any $t-1$ blocks contain at most l_2 points,*

From a $(m, k, 1)$-BIBD, we can construct a (t, l_1, l_2)-distribution design, and from that, a (t, b, l_1)-combinatorial RTS. The construction details can be found in [18].

In general, when modifying a (t, b, l_1)-combinatorial RTS to include redundancy, it becomes a $(t, b, \tau l_1)$-combinatorial RTS.

The communication complexity of a Repairable Threshold Scheme is defined as the total number messages transmitted in the protocol. In [13], it is mentioned that a (t, n, d)-RTS has a communication complexity of d. Therefore, the new communication complexity, after including redundancy, becomes $O(d\tau)$.

It is important to note that in practical scenarios, such as large distributed systems with numerous participants and BIBDs with a substantial number of elements, the increased communication complexity may render the scheme impractical for real-world settings. Additionally, the repair protocol stops whenever at least one inconsistent share is detected, requiring a restart of the repair process even if only one message was corrupted in transit. In the following sections, we will explore alternative approaches to constructing a maliciously secure repairable threshold scheme that aims to reduce the communication complexity while maintaining robustness against malicious players.

5.2 Construction via Bivariate Polynomials

Our previous construction applies a redundancy technique to any combinatorial RTS. This inflates the communication complexity of our repairing algorithm and we still require a larger proportion of honest participants over dishonest participants (more than half of the owners of the sub-shares are required to be honest).

In this section, we construct a protocol which is secure under the assumption that at least one of the helping nodes is honest. This relaxed assumption is the best we can hope to achieve our construction under, given that if all helping players were malicious, we could not hope to repair one's share.

The following construction is derived from bivariate polynomials. The dealer constructs a random polynomial which satisfies the following equation:

$$f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} f_{ij} x^i y^j, \text{ with } f_{00} = s$$

where s is the secret. In the original [3] construction of an actively-secure MPC protocol, they make use of such a polynomial, providing each participant P_i with a share $(h_i(x) = f(x, \alpha_i), v_i(y) = f(\alpha_i, y))$. Then, given a share from participant P_j, P_i can verify that $v_i(j) = h_j(i)$.

$$v_i(\alpha_j) = f(\alpha_i, \alpha_j)) = h_j(\alpha_i)$$

We will adapt this technique to our setting of repairing algorithms in combinatorial RTSs.

Let us demonstrate our construction with an example from [13] (and preceding section). Recall that sub-shares are allocated according to the distribution design below.

$$\begin{array}{lll} P1 \leftarrow \{1,2,3\} & P5 \leftarrow \{2,5,8\} & P9 \leftarrow \{3,4,8\} \\ P2 \leftarrow \{4,5,6\} & P6 \leftarrow \{3,6,9\} & P10 \leftarrow \{1,6,8\} \\ P3 \leftarrow \{7,8,9\} & P7 \leftarrow \{1,5,9\} & P11 \leftarrow \{2,4,9\} \\ P4 \leftarrow \{1,4,7\} & P8 \leftarrow \{2,6,7\} & P12 \leftarrow \{3,5,7\} \end{array}$$

Assuming that these sub-shares correspond to the values $h_i(x) = f(x, \alpha_i)$, $v_i(y) = f(\alpha_i, y)$, and players receive their shares in the following format:

$$\begin{array}{c} P1 \leftarrow \{(h_1(x), v_1(y)), (h_2(x), v_2(y)), (h_3(x), v_3(y))\} \\ P2 \leftarrow \{(h_4(x), v_4(y)), (h_5(x), v_5(y)), (h_6(x), v_6(y))\} \\ P3 \leftarrow \{(h_7(x), v_7(y)), (h_8(x), v_8(y)), (h_9(x), v_9(y))\} \\ etc.... \end{array}$$

Now, suppose P_5 wishes to repair their share. They can elicit the help from P_1, P_2, and P_3. P_1 sends $(h_2(x), v_2(\alpha_8))$, P_2 sends $(h_5(x), v_5(\alpha_2))$, and P_3 sends $(h_8(x), v_8(\alpha_5))$.

Then, P_5 verifies the following consistency checks:

$$\begin{aligned} h_2(\alpha_5) &= v_5(\alpha_2), \\ h_5(\alpha_8) &= v_8(\alpha_5), \text{ and} \\ h_8(\alpha_2) &= v_2(\alpha_8). \end{aligned}$$

These consistency checks ensure that if one or two players behave maliciously and submit incorrect values, the inconsistency will be detected. On the other hand, if all players behave honestly, the consistency checks will pass. It is worth noting that P_5 could have similarly requested their shares from (P_7, P_8, P_9) or (P_{10}, P_{11}, P_{12}). The helping players must know which values, v_i, to send, and this is determined by the participating players in the repairing algorithm. As long as there is some cyclic nature in the verifying values, v_i, provided, we can ensure that the repairing algorithm is secure in the presence of at least one honest helping participant.

The approach based on bivariate polynomials ensures security with a relaxed assumption, making it possible to repair one's share as long as there is at least one honest helping player. In the following subsections, we will provide a formal description of our approach and present proofs of correctness and privacy.

5.2.1 Formal description

Our construction begins with the dealer, who creates a bivariate polynomial satisfying a specific equation. The dealer then allocates shares to each participant. The process is formalized as follows:

Algorithm 6 Addition to Initialization and Share Distribution

```
Require: Degree of the random polynomial $t - 1$
Require: Secret $s$ to be distributed
Require: Participants $P_i$, where $1 \leq i \leq n$
1: Generate a random polynomial $f(x, y)$ of degree $t - 1$, where $f_{00} = s$
   where $s$ is the secret that we wish to distribute.
2: Choose $\alpha_j$, where $1 \leq j \leq m$
3: Allocate the sub-shares following the distribution design from [13]
4: for $i$ in 1 to $n$ do
5:     For each sub-share index $\alpha_k$ assigned to participant $P_i$
6:     Compute $h_k(x) = f(x, \alpha_k)$
7:     Compute $v_k(y) = f(\alpha_k, y)$
8:     Send $(h_k(x), v_k(y))$ as the share pair for participant $P_i$
9: end for
```

The initialization and share distribution process begins with the dealer generating a random polynomial $f(x, y)$ of degree t with the coefficient f_{00} equal to the secret s that needs to be distributed. Then, values α_j are chosen, where $1 \leq j \leq m$. The sub-shares allocation is determined based on the distribution design seen in [13]. For each participant P_i, the algorithm computes the share pairs $(h_k(x), v_k(y))$ by evaluating the polynomial $f(x, y)$ at the corresponding sub-share index α_k. Finally, the share pairs are sent to the respective participants.

When it comes to the repair process, we can formally define it as in Algorithm 7.

The algorithm begins with the repairing player P_r selecting the helping players according to the BIBD structure seen in Chapter 5.1. Then, P_r requests the appropriate values $(h_k(x), v_k(\alpha_r))$ from each helping player P_k. If the consistency checks are full-filled for all received values, the repairing player sets $(h_k(x), v_k(y)), (h_l(x), v_l(y)), ..., (h_z(x), v_z(y))$ as the new share. Otherwise, the output is $\perp$, indicating an inconsistent share.

We have provided formal descriptions of the initialization and share distribution as well as and the repair algorithm. For reconstruction we can simply follow the same algorithm as seen in [3] with some minor modifications.

Algorithm 7 Repair

```
Require: Repairing player $P_r$
Require: The dealer is honest and has previously shared the bivariate polynomial.
Require: Each player knows their corresponding $\alpha_i$ values
Require: Available repairing players $P_i$, where $1 \leq i \leq n$ with relevant share pairs $(h_i(x), v_i(y))$
 1: Request aid from one player for each $\alpha_k$ of $P_r$ according to previously seen structure
 2: for Each repairing node $P_k \in R_r$ where $R_r$ is the set of helping players for $P_r$ do
 3:     Send $(h_k(x), v_k(y))$ to $P_r$
 4: end for
 5: Perform consistency check on received shares
 6: for each received $(h_k(x), v_k(y))$,$(h_{k+1}(x), v_{k+1}(y))$ from players in $R_r$ do
 7:     Verify if $h_{k \mod |R_r|}(\alpha_{k+1 \mod |R_r|}) = v_{k+1 \mod |R_r|}(\alpha_{k \mod R})$
 8:     if Values do not match then
 9:         Output Null value $\perp$
10:     end if
11: end for
12: Reconstruct the initial share using the received shares as $(h_k(x), v_k(y)), (h_l(x), v_l(y)), ...., (h_z(x), v_z(y))$
```

5.2.2 Proofs of correctness and privacy

We will analyze whether the modified scheme fulfils the properties of correctness and privacy, as defined in Chapter 3.1 in the same way we did for Redundancy construction in Chapter 5.1.

Proof of Correctness

To prove the correctness of our scheme based on bivariate polynomials, we will assume the existence of a repairing player P_r who receives a reconstructed share $(h'_k(x), v'_k(y))$ that is different from the original share $(h_k(x), v_k(y))$. We will show that this assumption leads to a contradiction, proving that the repairing player receives their correct share.

If $(h'_k(x), v'_k(y)) \neq (h_k(x), v_k(y))$, it implies that at least one player has sent $(h'_k(x), v'_k(y))$ as their sub-share, and it has passed the consistency check. In other words, for some indices α_k, α_m, and α_l, we have:

$$h'_k(\alpha_l) = v_l(\alpha_k) \quad \text{and} \quad h_m(\alpha_k) = v'_k(\alpha_m)$$

The key observation here is that the polynomial $f(x,y)$ is a secret known only to the dealer. The malicious player attempting to construct a different polynomial to pass the consistency check must know the values for $h_m(\alpha_k)$ and $v_l(\alpha_k)$ as to create polynomials for which $h'_k(\alpha_l) = v_l(\alpha_k) \quad \text{and} \quad h_m(\alpha_k) = v'_k(\alpha_m)$. However, since $f(x,y)$ is only known to the dealer who is now inactive and the malicious player controls subset of the repairing nodes smaller than t , they cannot reconstruct neither the secret s nor the $f(x,y)$

Therefore, the malicious player is effectively guessing the values to satisfy the consistency check. The probability of guessing each value correctly is at most $1/q$, where q is the size of the finite field used for the polynomial evaluations. This probability is acceptably low and represents the best security we can hope to achieve.

Hence, we have shown that the assumption of receiving a reconstructed share $(h'_r(x), v'_r(y))$ different from the original share leads to a contradiction. Therefore, the repairing player P_r indeed receives their correct share with a probability at best $P = 1 - 1/q$, and the correctness of our scheme is proven.

Proof of Privacy

To establish the privacy property of our scheme based on bivariate polynomials, we aim to show that no unauthorized group of players can gain any knowledge about the secret during the repair process except for what they would learn from the inputs and outputs anyways.

Assume that there exists a malicious player who controls fewer than t nodes as this is a (n,k,t)-RTS. At the beginning, the adversary gets up to t pairs of polynomials $(h_i(x), k_i(y))$, the goal of the adversary is to use the shares it knows to find $f(x,0)$ or $f(0,x)$ and then calculate $f(0,0) = s$. But these shares give no information about the secret as the polynomial $f(x,y)$ is of degree t and to uniquely define it $t+1$ points are needed.

The only knowledge gained is when P_r outputs $\perp$ so they can check if their guess for $(h'_r(x), v'_r(y))$ was correct or not. With sufficiently big F_q this is a very minor advantage especially as we can consider starting banning nodes who fail checks several times as to avoid such situations.

Complexity

During the initialization and share distribution phase, the dealer needs to generate $f(x,y)$, and calculate $h_k(x)$ and $v_k(y)$ for all participants and for each sub-share index assigned to them. Each polynomial calculation involves operations of degree t. Assuming each participant has τ sub-shares, the complexity for each participant becomes $O(t\tau)$, and for all n participants, the complexity is $O(nt\tau)$.

During the repair phase, the repairing player receives $(h_k(x), v_k(y))$ from each helping player and performs consistency checks on the received shares. As P_r receives one double $(h_k(x), v_k(y))$ from d repairing players and $h_k(x)$ and $v_k(y)$ are polynomials of degree $t-1$ and we assume that the secret s is approximately of the same size as coefficients of $f(x,y)$, we find that the communication complexity is $O(2d(t-1))$.

5.3 Maliciously Secure Enrollment Scheme

Building upon the enrollment scheme detailed in Chapter 2.3, we introduce additional measures to ensure security in a malicious setting. We will incorporate commitments which allow the repairing participants to verify the integrity of each stage of the protocol.

5.3.1 Commitment Schemes

Commitment schemes serve as a cryptographic protocol to ensure reliable information exchange between parties who may not fully trust each other. Analogous to the function of signatures in physical contracts, commitment schemes ensure that once a commitment has been made to a particular message, it cannot be altered retrospectively.

A commitment scheme involves two phases: the commitment phase and the opening phase. In the commitment phase, the Sender holds a message m and chooses a random key K to "encode" the message and sends this to the Receiver. This what we call a commitment to the message. In the opening phase, the Sender reveals the key to the Receiver, allowing the Receiver to verify that commitment and the message match.

There are two main properties a commitment scheme should satisfy:

- **Hiding:** Receiving a commitment to a message m should give no information to the Receiver about m.
- **Biding:** After the Commit phase, there exists only one value m that will be accepted by the Receiver in the Open phase.

These properties can be either perfect or computational, i.e., against unbounded or computationally bounded adversaries, respectively. It is important to note that it is impossible to satisfy both properties simultaneously against adversaries with unlimited computational power. However it is possible, to have schemes in which the Hiding property holds against computationally unbounded adversaries and the Binding property holds for computationally bounded adversaries or the other way around.

The most simple commitment scheme is simply computing a cryptographic hash function of the message and sharing it alongside the original message.

The receiver can then verify the authenticity of the message by comparing the received hash with one computed from the received message. However, this form of commitment scheme does not allow for the addition of committed values without revealing them and may not hold up against computationally unbounded adversaries, who could potentially compromise it. To address these limitations, more advanced commitment schemes have been developed such as Pedersen Commitment Scheme.

Pedersen Commitment Scheme

In [15], Pedersen proposes an efficient non-interactive scheme for verifiable secret sharing used to securely commit to a value without revealing the value itself.

The commitments are perfect hiding, meaning that an adversary cannot learn any information about the committed value, and computationally binding, meaning that the committed value cannot be changed.

Algorithm 8 Pedersen Commitment Scheme

```
Require: A value $x$ to be committed, a random value $r$, and generators $g$
    and $h$ such that nobody knows $\log_g h$
1: Compute $C = g^x \cdot h^r$
2: return $C$
```

This commitment can be later revealed, along with x and r, to prove that the committed value was indeed x.

5.3.2 Construction via Pedersen Commitments

In this section we will expand the enrollment scheme seen in Chapter 2 using Pedersen's Commitment. To quickly explain the protocol we have:

A threshold scheme (t, n) with a secret s, where P_r is the repairing player and the helping players P_1,P_2....P_d are such that $d = t$.

Note that we can recover P_r's share using the following equation:

$$\phi_r = f(r) = \sum_{i=1}^{t} \zeta_i \phi_i,$$

where ζ_i is the Lagrange coefficients. Here we are using the same approach seen in Chapter 2.1.1, where Shamir's scheme is explained.

In this approach, the Initialize, Share, and Reconstruct algorithms are the same as the ones described in Chapter 2.3.

Algorithm 9 Proposed Verifiable Repairable Threshold Scheme Protocol

```
1:  procedure Repair Protocol($P_1$,$P_2$....$P_d$)
Require: $d \geq t$
2:      for $1 \leq i \leq d$ do
3:          $P_i \leftarrow$ compute random $\delta_{ji}$ $1 \leq j \leq d$ so that $\zeta_i\phi_i = \sum_{i=1}^{d} \delta_{ji}$
4:          for $1 \leq j \leq d$ do
5:              $P_i$ sends $\delta_{ji}$ to $P_j$
6:          end for
7:      end for
8:      for $1 \leq j \leq d$ do
9:          $P_j$ computes $\sigma_j = \sum_{j=1}^{d} \delta_{ji}$
10:     end for
11:     for $1 \leq i \leq d$ do
12:         $P_i$ sends $\sigma_i$ to $P_r$
13:     end for
14:     $P_r$ computes sum of commitments $\sum_{i=1}^{d} \sigma_i$
15:     if Commitments $\sum_{i=1}^{d} \sigma_i = \zeta_i\phi_i$ then
16:         $P_r$ sets its share as $\zeta_i\phi_i$
17:     else
18:         Outputs $\perp$
19:     end if
20: end procedure
```

Formal Description

Note that every item highlighted in red, such as $\zeta_i\phi_i$, has a commitment made using Pedersen's Commitment Scheme. During the initialization phase of the protocol, the dealer made public commitments to $\zeta_i\phi_i$ for all P_i.

Pedersen's commitments are computationally binding, ensuring that in the first step each P_i is sending values which sum up to $\zeta_i\phi_i$, meaning that malicious players cannot send different values as in the attack seen in Chapter 4.3.

Each P_i essentially re-shares their value, but splitting it between other participants means that less than d malicious actors cannot find out their value.

Now we will argue how the commits for the red values protects this scheme against the attacks shown earlier. Especially as every P_i has to commit every δ_{ji} it shares means that it can't send wrong values to P_r without being discovered by other players.

Correctness and Privacy

We will first establish the correctness of the protocol and then prove its privacy.

Proof of Correctness

The correctness of the scheme for honest-but-curious adversaries has already been shown in [18]. We will now demonstrate its correctness for malicious adversaries.

In the case of active adversaries, a malicious player may attempt to send incorrect values δ_{ji} in line 5 of the repair protocol or share incorrect σ_i with P_r. To address this, each participant commits to $\zeta_i \phi_i$ publicly, where ζ_i is the commitment to the correct δ_{ji} and ϕ_i is the commitment to the corresponding σ_i. Participants can verify the correctness of the commitments they receive by comparing them with the expected values $\zeta_i \phi_i = \sum_{i=1}^{d} \delta_{ji}$. By using the additive properties of Pedersen commitments, the correctness of all σ_i can also be verified as all participants already have δ_{ji} they can also verify the correctness of all σ_i using the same additive properties.

$$\begin{aligned}
\zeta_r \phi_r &= \sum_{i=1}^{t} \sigma_i \\
&= \sum_{j=1}^{t} \sum_{i=1}^{t} \delta_{ji} = \sum_{i=1}^{t} \sum_{j=1}^{t} \delta_{ji} \\
&= \sum_{i=1}^{t} \zeta_i \phi_i = \zeta_r \phi_r
\end{aligned} \tag{5.1}$$

Given that there is at least one honest participant, the protocol remains secure, even in the presence of malicious participants. The repairing player, P_r, verifies that $\sum_{i=1}^{t} \sigma_i = \zeta_i \phi_i$, outputting $\perp$ if the verification fails.

When commitment verification fails, at least one helping player P_i submitted a commitment for which $\sigma_i' \neq \sigma_i$. The repairing player, P_r, then outputs $\perp$ fulfilling the property of correctness.

Proof of Privacy

We employ a similar argument to [18] to prove the privacy property. If a coalition of $d-1$ players attempts to calculate the secret, they may have knowledge of $\sum_{i=1}^{d-1} \sigma_i$, but the missing value σ_k is completely random. Knowing this value is equivalent to knowing the value of the secret itself. Therefore, no unauthorized player can gain any knowledge about the secret during the repair protocol, satisfying the privacy property.

Complexity

Lastly, we look at the complexity of the proposed commitment scheme. The construction process of p and q involves an initial selection of q, followed by determining p as the smallest prime congruent to 1 modulo q. Heuristics show that $p \leq q \log q^2$ [15], which suggests that a commitment to q bits necessitates a maximum of $q + 2 \log q$ bits.

Looking at Chapter 2.3, the communication complexity of the repair algorithm the total complexity is $O(d^2)$ for the initial algorithm. Adding commitments means that for each message send there is a commitment send as well making the complexity of the improved scheme $O(d^2)$

Chapter 6

Evaluation

In this chapter, we will compare the three schemes we have discussed in the previous sections: The redundancy scheme as discussed in Chapter 5.1, the construction via Bivariate Polynomials as detailed in Chapter 5.2, and the Maliciously Secure Enrollment Scheme as described in Chapter 5.3. We will evaluate these schemes in terms of their communication complexity, security, and correctness as well as compare it with `DM-Recover` scheme from [7] a proactive secret sharing scheme which also offer repair algorithm to recover lost or corrupted shares.

To recap from Chapter 2, *communication complexity* is the sum of the sizes of all messages transmitted during the execution of an algorithm. Where n is the number of participants, d is the number of players needed to repair a share and τ is the number of times a sub-shares occurs in combinatorial RTS. Table 6.1 compares the communication complexity and malicious adversaries threshold of these schemes.

Name	**Communication Complexity**	**Threshold of actively malicious adversaries**
Redundancy Scheme	$O(d\tau)$	$\frac{\tau}{2}$
Bivariate Polynomials Scheme	$O(2d(t-1))$	$d-1$
Enrolment Commitment Scheme	$O(2d^2)$	$d-1$
`DM-Recover` Scheme	$O(n^3)$	$\frac{n}{2}$

Table 6.1: Comparison of Communication Complexity and Threshold

It is clear that different schemes offer different levels of resilience against actively malicious adversaries. The choice of scheme would depend on the specific requirements of the application. We also need to take into account computational complexity of these different protocols, but with computational abilities of modern devices, communication complexity is our focus.

The redundancy scheme is the least efficient as it supports the smallest number of actively malicious adversaries, even if in comparison its communication and computation complexity is quite small. Both bivariate polynomials construction and enrolment scheme with commitments offer significant improvements communication complexity compared to `DM-Recover` scheme. The choice between bivariate polynomials and enrollment commitment scheme will depend on specific values of n, d and τ.

In conclusion, if communication complexity is a priority and τ and supposed number of adverseries is low, redundancy scheme might be preferable. If number of adverseries is assumed to be higher, bivariate polynomials or enrollment commitments are preferable.

Notably, all of the modified schemes display an increased communication complexity compared to the original scheme. However, when considering the improved thresholds of against against actively malicious adversaries, this higher complexity can be seen as a valuable trade-off.

Chapter 7

Conclusion

This project undertook an intensive study of repairable threshold schemes and the influence of actively malicious participants on these models.

We introduced new definitions of correctness and privacy for repairable threshold schemes, which we used when evaluating attacks on existing RTS constructions. After presenting a survey of existing schemes, we introduced three modified schemes based on [13], subsequently proving their correctness and privacy, as well as evaluated their communication complexity. These three new constructions we proposed include the redundancy scheme, the commitments enrolment scheme and construction using bivariate polynomials.

Finally, we compared the three constructions against `DM-Recover` construction, which is a proactive secret sharing scheme and also secure against malicious players. Then we concluded that the most optimal solution is either commitments enrolment scheme or construction using bivariate polynomials.

This work can be extended in several directions, especially when it comes to expanding malicious security to regenerating codes RTS also introduced in [13] and briefly introduced in Appendix 2.4. There seems to be an interesting extension from Robust Threshold Schemes using Difference Sets but there was not enough time to come up with distribution designs needed.

Appendix A

Appendix

A.1 Lagrange Interpolation

As to understand Shamir's secret sharing scheme we need to define Lagrange interpolating polynomial.

Given a set of $n+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, where $x_0, x_1, \ldots, x_n$ are distinct. For this set of points the Lagrange interpolation polynomial $P(x)$ is given by:

$$L(x) = \sum_{i=0}^{n} y_i \ell_i(x)$$

where $\ell_i(x)$ are the Lagrange basis polynomials defined as:

$$\ell_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Where each $\ell_i(x_j)$ has the following property:

$$\ell_i(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

The Lagrange interpolation polynomial $L(x)$ is a sum of the y_i values multiplied by $\ell_i(x_j)$ evaluated at x and so the resulting polynomial has the following property $L(x_i) = y_i$ for all $i = 0, 1, \ldots, n$.

Most importantly this polynomial is unique. The proof of this statement can be found in [10].

A.2 Shamir's Secret Sharing Scheme Example

Here we have an example of a $(3, 5)$-threshold scheme in field $\mathbb{Z}_{101}$. Suppose we have a secret $s = 42$ that we want to distribute among 5 participants. To do this we need to choose 3 random coefficients a_0, a_1, a_2. In this case we choose $a_0 = 42$, $a_1 = 11$, and $a_2 = 8$ from $\mathbb{Z}_{1009}$.

We then compute the polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2$$

$$f(x) = 2 + x + 5x^2$$

We can find the shares are by evaluating the polynomial $f(x)$ at 5 different points $x_1, x_2, \ldots, x_5$. We choose $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 4$, and $x_5 = 5$ for convenience.

The shares for each participant P_i are:

$$\begin{aligned} P_1 &= (1, f(1)) = (1, 8) \\ P_2 &= (2, f(2)) = (2, 24) \\ P_3 &= (3, f(3)) = (3, 50) \\ P_4 &= (4, f(4)) = (4, 86) \\ P_5 &= (5, f(5)) = (5, 132) \end{aligned}$$

Suppose now that participants P_1, P_2, P_3 wish to reconstruct the secret. They share their shares with each other, and then compute the Lagrange coefficients λ_i as follows:

$$\begin{aligned} \lambda_1 &= \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{1}{2}x^2 - \frac{5}{2}x + 3 \\ \lambda_2 &= \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -x^2 + 4x - 3 \\ \lambda_3 &= \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{1}{2}x^2 - \frac{3}{2}x + 1 \end{aligned}$$

Then, we compute $f(x)$:

$$\begin{aligned}
f(x) &= \lambda_1 \cdot f(1) + \lambda_2 \cdot f(2) + \lambda_3 \cdot f(3) \\
&= 8 \cdot (\frac{1}{2}x^2 - \frac{5}{2}x + 3) + 24 \cdot (-x^2 + 4x - 3) + 50 \cdot (\frac{1}{2}x^2 - \frac{3}{2}x + 1) \\
&= 5x^2 - x + 2
\end{aligned}$$

So we find the original secret is $f(0) = 2 \mod 1009 = 2$. Note that any subset of 3 or more participants could also have reconstructed the secret, since a $(3, 5)$-threshold scheme was used.

Bibliography

[1] Anna Baydakova. Is ledger's new bitcoin key recovery feature safe? experts have doubts, 2023.

[2] 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. Cryptology ePrint Archive, Paper 2020/800, 2020. https://eprint.iacr.org/2020/800.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 1–10, New York, NY, USA, 1988. Association for Computing Machinery.

[4] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 195–208. Springer, 2012.

[5] Anirudh Chandramouli, Ashish Choudhury, and Arpita Patra. A survey on perfectly secure verifiable secret-sharing. *ACM Computing Surveys (CSUR)*, 54:1 – 36, 2021.

[6] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 383–395. IEEE, 1985.

[7] Shlomi Dolev, Karim Eldefrawy, Joshua Lampkins, Rafail Ostrovsky, and Moti Yung. Proactive secret sharing with a dishonest majority. In *Security and Cryptography for Networks: 10th International Conference, SCN 2016, Amalfi, Italy, August 31–September 2, 2016, Proceedings*, pages 529–548. Springer, 2016.

[8] Karim Eldefrawy, Nicholas Genise, Rutuja Kshirsagar, and Moti Yung. On regenerating codes and proactive secret sharing: Relationships and implications. In *Stabilization, Safety, and Security of Distributed Systems: 23rd International Symposium, SSS 2021, Virtual Event, November 17–20, 2021, Proceedings 23*, pages 350–364. Springer, 2021.

[9] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.

[10] Jeffrey Humpherys, Tyler J Jarvis, and Emily J Evans. Foundations of applied mathematics, volume 2: Algorithm design and optimization.

[11] Steve Jiekak, Anne-Marie Kermarrec, Nicolas Le Scouarnec, Gilles Straub, and Alexandre Van Kempen. Regenerating codes: A system perspective. *ACM SIGOPS Operating Systems Review*, 47(2):23–32, 2013.

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

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

[14] 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.

[15] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*, pages 129–140. Springer, 2001.

[16] David A Schultz, Barbara Liskov, and Moses Liskov. Mobile proactive secret sharing. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, pages 458–458, 2008.

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

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

[19] Martin Tompa and Heather Woll. How to share a secret with cheaters. *journal of Cryptology*, 1(3):133–138, 1989.